QUANTIFYING CANCER CELL MOTILITY IN AN *IN VITRO* SYSTEM

By

Walter Georgescu

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Biomedical Engineering

August, 2012

Nashville, Tennessee

Approved By:

John P. Wikswo, PhD

Vito Quaranta, PhD

Jeffrey M. Davidson, PhD

Adam Anderson, PhD

Shane Hutson, PhD

Frederick R. Haselton, PhD

To my wife Jen and my mom Marilena

ACKNOWLEDGEMENTS

I have been fortunate to have very supportive advisors for my Ph.D. So first of all I would like to thank Dr. John Wikswo and Dr. Vito Quaranta for their insight and suggestions, their praises and criticisms, their financial support and their fantastic ideas. Dr. Wikswo, even though you are one of the busiest people I have ever met, you have always taken the time to meet with me when I needed your help and always provided me with great advice on my experiments and my manuscripts. Vito, your enthusiasm and generosity, your open door policy and your biological expertise have made what could have been a very difficult transition from building microfluidic devices to working with cancer cells a very enjoyable and interesting process for me. I would also like to give special thanks to Dr. Jeff Davidson for always giving me great advice and for always having my best interests at heart. I'd like to thank my other committee members, Dr. Adam Anderson, Dr. Shane Hutson and Dr. Rick Haselton. Dr. Anderson, I enjoyed your MRI class very much, and it also helped me become more familiar with MATLAB at a time when I was just getting started with it. Dr. Hutson and Dr. Haselton, your comments and recommendations during the committee meetings and occasional informal talks have improved this work.

I would like to thank Jerome Jourquin for helping me when I was getting started with cell biology and Manisha Tripathi for her help and experiments with the prostate cancer cell lines, as well as for being great lab mates. Allison Price, thank you for your editorial assistance on manuscripts; your changes have always improved them. I would like to thank Ron Reiserer for helping me any time I had any sort of fabrication question – your expertise is unmatched. Phil Samson, thank you for your questions and suggestions for improvements during our morning

meetings. Thank you, Dave Schaffer, for teaching me clean room techniques. A big thanks to Jing Hao for helping me with any cell culture questions. Thank you, Darren Tyson, for being CellAnimation user number one. Your comments improved the software. Shawn Garbett and Sam Hooke, thank you for your work on CellAnimation. Your additions help make CellAnimation great. Shawn, thank you also for introducing me to novel mathematical techniques and sending me interesting papers. Cheryl Cosby, thank you for always finding a spot in Dr. Wikswo's calendar when I needed to talk to him. Lourdes Estrada, thanks for your suggestions and advice during our lab and informal meetings. I've always found them really helpful.

Thank you, all my friends in Nashville, for making life in Music City fun. Thank you, Jen, for your support, love and patience during the years we've been married. And finally, thanks, mom, for everything.

This is a long list, and I may have missed somebody who has helped make this work possible. If I omitted your name, know that it was an honest mistake, not ingratitude.

This work was supported by grants from the National Institutes of Health and the Whitaker Foundation, as acknowledged in the following chapters.

TABLE OF CONTENTS

Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Background

Cell migration plays an important part in embryonic development, wound healing and cancer invasion. In this work, we examine the role of hepsin overexpression in increased cancer cell motility and develop new technologies for the study of cell migration that improve on the current state-of-the-art. The techniques we develop allow us to extract individual dynamic cell properties and quantify not only motility, but morphology and ancestry parameters as well, at the single cell level. This type of data can be used to parameterize *in silico* cancer models, such as those being developed here at the Center for Cancer Systems Biology at Vanderbilt University, that capture the full dynamic range of heterogeneous cell populations.

Software for Automatic Analysis of Microscopy Images

Advances in microscope and camera technology have made it possible to image many fields of view using multiple wavelengths at rapid intervals over several days. Today's high-throughput microscopes can generate over the course of a few days the same amount of data that would have taken months to produce using a conventional fixed-stage microscope. Unfortunately, acquiring the images is only one step in the videomicroscopy workflow. The raw images have to be analyzed and all relevant data extracted before any conclusions can be drawn. Analyzing such a deluge of images manually is error-prone and unpractical. Only a computer or

a supercomputer can keep up with this rapid pace of image generation. If the goal is to analyze each image as a separate entity, commercial and free automated image processing software is available. However, if the goal is to identify each cell and track its trajectory across a set of time-lapse images, the current crop of software is not up to the task [1]. Cells will divide multiple times during a typical time-lapse experiment. It is important that software can detect these mitotic events. Combining data from multiple cells in a single track will affect the perceived heterogeneity of the population. In addition, it would not be possible to determine what percentage of the variation in motility is due to cell cycle.

Two of the most popular software packages for microscopy are MetaMorph™ from Molecular Devices and the open source application ImageJ™ which is available as a free download from the National Institutes of Health website (rsbweb.nih.gov/ij). MetaMorphcan be used to automatically segment images into individual cells. Once the cells are identified it can extract a number of parameters for each individual cell, such as area, intensity, shape, etc. The cell tracking feature, however, is manual and requires the user to click on each cell of interest at each time point. This is not feasible for analyzing more than a few hundred cells. Considering that a high-throughput data set consists of thousands of cells and thousands of images, manual tracking is not a viable option.

The ImageJ base package has a smaller number of features than MetaMorph. However, its open source architecture makes available hundreds of plugins for everything from image morphology to cell tracking. The quality of the plugins varies wildly. Many plugins are rudimentary, while some rival or even exceed commercial software capabilities. Of the several cell tracking plugins, only one is automated, and it requires cells that are not touching and are well isolated from the background, which is often not the case.

A number of commercial and free packages have been developed that use the MATLAB technical language from Mathworks. Two of the best are Imaris™ from Bitplane and the free package CellProfiler. For the purposes of cell tracking, Imaris offers the most advanced features of all the packages we evaluated. It can track cells in both 2D and 3D using several different tracking models, such as Brownian motion, autoregressive motion and connected components. For each object tracked, it can extract shape and intensity statistics and save these along with motion statistics to Excel files for further analysis. The main disadvantage is the high cost of ownership. The features listed above require users to purchase two packages, ImarisTrack™ and Imaris MeasurementPro™, in addition to the expensive base software. It is also not possible to set up Imaris™ on a computer cluster for true high-throughput processing.

CellProfiler was developed at the Broad Institute of MIT and Harvard [2]. It is a collection of MATLAB scripts that implement advanced algorithms for cell thresholding and segmentation. Unlike Imaris, it can be set up in a high-throughput mode on a cluster. Because the software is set up as a collection of MATLAB™ scripts, it can be easily interfaced with other MATLAB™ scripts. Similar to Imaris, CellProfiler can extract shape and intensity parameters for each object identified and export those parameters to Excel. However, CellProfiler has not been designed for studying cell migration, and its single loop architecture does not interface well with advanced tracking algorithms.

Recent interest in automated cell lineage generation has led to the development of several software packages with features that are useful to the cell migration community. The Roysam lab recently reported development of a cell lineage construction tool that includes a number of advanced features [3]. In addition to ancestry, the position, shape, cell-cell contacts and motility are stored for each cell at each time point.

The lineage tool is implemented in MATLAB. To identify individual cells, images are thresholded adaptively to segment the cells. Cells that are touching are split using a simple seeded watershed algorithm. Just as a watershed separates two drainage basins, the watershed algorithm identifies object boundaries in an image by flooding a topological surface [4]. One needs to define a topological surface where the drainage basins correspond to the objects of interest. One such surface is the gradient magnitude of the original image. Unfortunately, due to irregularities in the gradient, using the gradient image directly results in oversegmentation and numerous irrelevant boundary lines. In seeded watershed segmentation, a set of seed points is defined either manually or automatically. By allowing watershed ridges to be built only between catchment basins that are associated with a seed point, oversegmentation is controlled.

Every cell is then characterized using a number of features, such as centroid, area, eccentricity, major axis length and orientation. These characteristics are used to build a feature vector. Cells are tracked from frame to frame by calculating the probability that a cell with feature vector x has deformed, divided or split to become a cell with feature vector y in the next frame. Assignment of the cells from one frame to the next is made by picking the combination of probabilities that maximizes the total probability sum of the frame. The program was shown to perform robustly with a set of four cortical progenitor cells that grew to 78 cells over the course of three days, as observed using brightfield imaging. This type of approach is likely to work well as long as the number of cells is kept low; however, a simple seeded watershed algorithm will likely result in oversegmentation when cell densities become high. In addition, at high cell densities, the algorithm described above for cell assignment from one frame to the next becomes computationally very expensive.

Other automatic lineage tools have been developed but they are too specific to be useful to the general cell migration community. One such example is the Starrynight software package developed at the Waterston lab [5]. It works with very high accuracy for tracking and assigning ancestry to cells in the embryo of *C. elegans*, but it cannot be used with any other type of cells. A general purpose cell tracking software needs to be able to track different types of cells at high density under a variety of lighting conditions. It should also be capable of high-throughput processing and should export shape and ancestry data, as well as movies for visual inspection, in addition to position information. Finally, the software will preferably be open source and easy to interface with other programs. For our studies of cell migration we have developed a software package that meets these requirements.

Cell Migration

Cell migration plays a major role in many biological processes, including cancer invasion. Without it, the ability of cancer tumors to metastasize to other parts of the body would be largely eliminated [6]. For this work we have investigated cancer motility in two prostate cancer cell lines. According to the American Cancer Society, prostate cancer is the second leading cause of cancer death in North American men, with an expected 241,740 new cases and 28,170 deaths in 2012.

Proteases are enzymes that degrade proteins [7]. They have long been recognized to play a role in cancer progression and metastasis [8]. By degrading the extracellular matrix, proteases allow tumor cells to spread to other parts of the body. In addition, cleavage of extracellular matrix proteins by proteases creates protein fragments, some of which can in turn increase prostate cancer cell migration and motility. Laminin-332 is an extracellular matrix protein and an important component of the basement membrane. It is a heterotrimeric glycoprotein that consists

of α3, β3 and γ2 chains [9]. It has a cell scattering effect *in vitro* that is thought to play a role in both wound healing and tumor cell invasion. Laminin-332 is unique in both activity and structure, as it is the only laminin that contains the β3 and γ2 chains.

Cleavage of laminin-332 by various proteases has been shown to induce cell migration by a number of research groups. The Quaranta group has shown that cleavage of laminin-332 by matrix metalloprotease-2 (MMP2) induces cell migration in breast cancer epithelial cells [10]. The same group has shown that cleavage of laminin-332 by MT1-MMP also induces migration in different types of cancer cell lines (breast, colon, etc.) [11]. Remy et al. [12] have shown that MMP7 also cleaves laminin-332 and induces increased cell motility in the colonic carcinoma cell line HT29. Our lab has demonstrated that two different type II transmembrane serine proteases, matriptase and hepsin, cleave laminin-332, and that this cleavage leads to increased cell motility [13,14]. Klezovitch et al. [15] reported that hepsin overexpression causes basement membrane disorganization, promotes prostate cancer progression and causes metastasis to the lung, liver and bone in a mouse model of the disease. In contrast, Srikantan et al. [16] found that hepsin overexpression inhibits invasion in prostate cancer cells. In general, hepsin is overexpressed in over 90% of prostate cancers. In this work, we investigated how hepsin overexpression affects prostate cancer cell motility.

An important component of motility is directed or biased cell motility. There are multiple types of directed motility, such as chemotaxis, haptotaxis, durotaxis, electrotaxis, etc. Chemotaxis and haptotaxis seem very similar, and both have been hypothesized to play important roles in cancer cell invasion; however, chemotaxis is far better understood than haptotaxis. Of all the types of directed cell migration, chemotaxis has seen by far the most research activity. A recent search for the keyword “chemotaxis” on Pubmed returned 26,900

articles with 3,152 reviews, while a search for “haptotaxis” resulted in only 188 hits with 11 reviews.

At least two models of gradient sensing have been proposed for cells undergoing chemotactic migration. Cells that are specialized for migration, such as neutrophils and *Dictyostelium*, can sense gradients as shallow as 2% across the entire cell body [17]. For *Dictyostelium* this can mean as few as a difference of 5 receptors out of a total of 50,000 [18]. This remarkable feat is accomplished by internal amplification of the gradient. It is thought that this occurs through one or several local-excitation/global-inhibition mechanisms [19]. One of the main molecules involved in local excitation is the membrane lipid phosphatidylinositol-3,4,5-triphosphate (PI(3,4,5)P3). In neutrophils and *Dictyostelium*, PI(3,4,5) P3 forms gradients that are steeper than the external gradient. However, in other types of cells which are only temporary migrators such as fibroblasts and epithelial cells this internal amplification does not occur [20]. This might explain the poor ability of these cells to detect a chemotactic gradient.

Much less is known about haptotaxis, the migration of cells on a substrate-bound gradient of chemoattractant or cellular adhesion sites. The details of the gradient detection mechanism in haptotaxis are not yet clear. Aznavoorian et al. [21] showed that haptotaxis has at least one signaling pathway that is different from pathways involved in chemotaxis. Current research indicates that Rac/Cdc42 are involved in haptotactic migration [22]. One possible mechanism is creation of a gradient of integrin activity due to the distribution of surface binding sites leading to polarized activation of Rac and Cdc42 and protrusion of a leading edge. The activation of Rac and Cdc42 may also lead to a positive feedback loop and magnification of the external gradient [6]. Unlike chemotaxis, it is not known if there are differences between the gradient sensing mechanism of specialized migrating cells and the mechanism used by occasional migrators.

A part of the problem is due to the type of devices being used to study haptotaxis. Many chemotaxis or haptotaxis studies use a derivative of the Boyden chamber assay [23]. In the original assay, a cellulose ester filter membrane was used to separate an upper and lower chamber. The chemoatractant to be tested was placed in the lower chamber and the cell solution was placed in the top chamber. Migration was initially quantified as number of cells that traveled to the other side of the membrane. However, many cells did not make it across the membrane due to its 150 μm thickness. In a modified version of the assay, a shorter time was allowed for cells to migrate. In this version of the assay, instead of counting the numbers of cells on the bottom of the filter membrane, migration was quantified as the distance traveled by the leading cell front from the top of the membrane [17]. Modern versions of the Boyden chamber assay use disposable multi-well polycarbonate filters [24]. Because the filter membrane is only 10 μm thick, most cells drop off once they cross the membrane. The number of cells present in the bottom compartment at the end of the experiment is taken as a measure of migration. The problem with these types of assays is that they are end-of-experiment assays. They can only provide statistics for the behavior of the cells as a population at the end of the experiment, and the presence of subpopulations with different migration properties may lead to erroneous conclusions. To obtain insights into the mechanisms of gradient sensing, techniques that provide dynamic cell information about haptotaxis are needed.

There are a small number of studies that use methods which allow quantification of the adhesion gradients. In the experiments that coined the term “haptotaxis,” Carter [23] created a gradient of adhesion sites by depositing increasing amounts of evaporated palladium on a substrate of cellulose acetate. Mouse fibroblasts were observed to move towards the areas of higher adhesion.

Smith et al. [25] described migration of bovine aortic endothelial cells (bAECs) on gradients of alkanethiols functionalized with fibronectin. Cell movement was characterized using a Langevin equation:

$$\frac{d\vec{v}}{dt} = -\beta\vec{v} + \sigma\overrightarrow{W}(t) + \alpha\hat{x} \ ,$$

where $\vec{v} = v_x\hat{x} + v_y\hat{y}$, v_x and v_y are the velocity components parallel and perpendicular to the gradient direction, $-\beta\vec{v}$ is drag force, and $\overrightarrow{W}(t)$ is a function representing random impulses in the x and y directions, while σ represents the strength of those impulse and α is a parameter that quantifies the tendency of cells to drift up the gradient. Cells were observed to increase their drift speed, $S_d = \frac{\alpha}{\beta}$, on gradients of fibronectin in comparison to a control substrate containing a uniform density of fibronectin using two separate assays. In the first assay, cells were seeded at the low end of the gradient, which was blocked from the rest of the cells using a Teflon restraint. Once the cells reached confluence, the restraint was removed and cell positions were recorded. In the second assay, a small number of cells (3000-4000 cells/cm^2) were seeded randomly on the gradient and their positions were recorded until cells exited the field of view or touched another cell. No differences were observed between the drift speeds of cells grown in serum-free and serum-containing media.

The effect of gradient slope on haptotaxis is not yet clear. Several studies have looked at the effect of changing gradient slope on cells undergoing haptotactic migration on defined extracellular matrix gradients. However, results have been contradictory, and due to the low number of studies no definitive conclusion can be drawn at this time. In a subsequent study Smith et al. [26] examined the effect of changing the slope of fibronectin gradients on the

migration of human microvascular endothelial cells (hMEC). They observed a linear increase in drift speed with increasing slope. The minimum slope that was detected by the cells was 0.5 ng Fn/mm^3, and the maximum slope they were able to create was 1.2 ng Fn/mm^3. Maximum fibronectin density that could be deposited in the system was measured at 2 ng Fn/mm^3.

Liu et al. [27] studied migration of bAECs on alkanethiol gradients functionalized with fibronectin, as in the first Smith study. They tested cells on two different gradient slopes: a 1 mm wide gradient with a sharp slope of 1.44 ng/mm^3 and a 10 mm wide gradient with a shallow slope of 0.14 ng/ mm^3. Increased directional migration was observed for cells migrating on gradients when compared with a uniform density control. However, no statistically significant difference in speed was observed between the sharp and shallow slope gradients. In addition to fibronectin, Liu et al. generated VEGF gradients and combination VEGF-fibronectin gradients. BAECs exhibited a two-fold increase in directional migration on VEGF gradients compared to fibronectin and a further two-fold increase on combination VEGF-fibronectin gradients. However, again no significant change in speed was observed with changing gradient slope.

Chinese Hamster Ovary (CHO) cells show increased directional migration that is sensitive to the gradient slope and insensitive to the amount of fibronectin present. Rhoads et al. [28] used a microfluidic gradient generator to deposit fibronectin gradients of varying slopes and varying minimum and maximum densities. Change in slope from 0-25 µg/ml to 0-50 µg/ml and 0-100 µg/ml correlated with increased directional migration. However, the overall migration speed did not correlate with the change in slope. Maximum cell migration speed was recorded in the 0-50 gradient. In another experiment, the slope was kept constant and the range of fibronectin densities varied. In fibronectin gradients going from 0-50 µg/ml to 25-75 µg/ml and 50-100 µg/ml, no significant change in directional migration was observed.

It might seem surprising at first that cells exhibited the fastest migration speed on the intermediate slope gradient. However, cell speed does not depend on the slope of the gradient. The factor that determines overall migration speed is the total number of attachment sites available to a particular cell. It has been shown by Palecek et al. [29] that cell speed is a biphasic function with respect to the number of attachment sites, integrin levels and integrin-binding affinities. CHO cells migrating on different uniform concentrations of fibronectin exhibited a biphasic dependence on the surface density of fibronectin, regardless of integrin expression level or the integrin-ligand binding affinity.

In all conditions cells achieved the same maximum migration speed. The only difference between the different conditions was a shift in the position of the biphasic curve. These shifts could be predicted by the changes they caused in a single parameter, the short-term cell-substratum adhesion strength. It is known that cells can modify the cell-substratum adhesion strength by activating/inactivating integrins [30]. It has not yet been investigated if cells seek to maintain specific cell-substratum adhesiveness and what strategies they employ to achieve this goal. It is possible that cells in a migratory mode will show a preference for areas where the substratum-cell adhesiveness is optimal for migration. In this case, cells will migrate up or down the gradient to the optimal surface density of protein, and then they will migrate perpendicular to the gradient in order to maintain optimum adhesion.

This type of migration can be confused with contact guidance, especially in the case of steep gradients. Contact guidance is the ability of cells to follow changes in topography. The fact that cells tend to use the direction cues provided by topography was observed as early as 1912 by Harrison [31]. He cultured cells on spider webs and showed that the webs affected the cell's direction of movement. The term "contact guidance" was coined by Paul Weiss. In a study

published in 1945, Weiss examined 5,000 tissue cultures of Schwann cells and nerve fibers and observed that both cells and nerve fibers followed the orientations of the fibrils in the "ground mat" (ECM) [32].

These early studies suffered from the lack of well-defined substrates. Advances in technology have made it possible to create uniform patterns on a variety of substrates using photolithography [33]. In a series of studies, Clark et al. [34,35] used grooved substrates to show that contact guidance is strongly dependent on groove depth. Nevertheless, the effect of the topography is hard to separate from the effect of the medium and proteins adsorbing from the medium. In a recent study Teixeira et al. [36] showed that cells cultured on 400 nm pitch grooves aligned perpendicular to the grooves when grown in Epilife medium, but they aligned parallel to the grooves when DMEM/F12 medium was used. In our studies, we have used different methods to generate the patterns to eliminate any possible effects of contact guidance. In addition, the use of short second deposition steps followed by long blocking steps at high concentration of blocking protein should further eliminate any effects of topography directing cell migration paths.

Conclusion

To understand and prevent cancer metastasis we need to understand the mechanisms of cell motility. In this introduction we have given a brief overview of current status of software applications and methods used to study cell motility. We have also discussed the effects of proteases and laminin-332 on cell motility and looked at the similarity and differences between

chemotaxis and haptotaxis. In the following chapters we will show how our work adds to each of these areas.

References

[1] S.T.C. Wong, Informatics challenges of high-throughput microscopy, IEEE Signal Processing Magazine. 23 (2006) 63-72.

[2] A.E. Carpenter, T.R. Jones, M.R. Lamprecht, C. Clarke, I.H. Kang, O. Friman, et al., CellProfiler: image analysis software for identifying and quantifying cell phenotypes, Genome Biology. 7 (2006) 1-11.

[3] O. Al-Kofahi, R.J. Radke, S.K. Goderie, Q. Shen, S. Temple, B. Roysam, Automated Cell Lineage Construction, Cell Cycle. 5 (2006) 327-335.

[4] R.C. Gonzales, R.E. Woods, Digital Image Processing, Addison-Wesley, 1992.

[5] Z. Bao, J.I. Murray, T. Boyle, S.L. Ooi, M.J. Sandel, R.H. Waterston, Automated cell lineage tracing in Caenorhabditis elegans, Proceedings of the National Academy of Sciences of the United States of America. 103 (2006) 2707-12.

[6] S. Li, J.-L. Guan, S. Chien, Biochemistry and biomechanics of cell motility, Annual Review of Biomedical Engineering. 7 (2005) 105-50.

[7] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter, Molecular Biology of the Cell, Garland Science, 2007.

[8] C. Basbaum, Z. Werb, Focalized proteolysis: spatial and temporal regulation of extracellular matrix degradation at the cell surface, Current Opinion in Cell Biology. 8 (1996) 731-738.

[9] M.P. Marinkovich, Tumour microenvironment: laminin 332 in squamous-cell carcinoma, Nature Reviews. Cancer. 7 (2007) 370-80.

[10] G. Giannelli, J. Falk-Marilier, O. Schiraldi, W.G. Stetler-Stevenson, V. Quaranta, Induction of Cell Migration by Matrix Metalloprotease-2 Cleavage of Laminin-5, Science. 277 (1997) 225-228.

[11] N. Koshikawa, G. Giannelli, V. Cirulli, K. Miyazaki, V. Quaranta, Role of cell surface metalloprotease MT1-MMP in epithelial cell migration over laminin-5, The Journal of Cell Biology. 148 (2000) 615-24.

[12] L. Remy, C. Trespeuch, S. Bachy, J.-Y. Scoazec, P. Rousselle, Matrilysin 1 influences colon carcinoma cell migration by cleavage of the laminin-5 beta3 chain, Cancer Research. 66 (2006) 11228-37.

[13] M. Tripathi, S. Nandana, H. Yamashita, R. Ganesan, D. Kirchhofer, V. Quaranta, Laminin-332 is a substrate for hepsin, a protease associated with prostate cancer progression., The Journal of Biological Chemistry. 283 (2008) 30576-84.

[14] M. Tripathi, A.A. Potdar, H. Yamashita, B. Weidow, P.T. Cummings, D. Kirchhofer, et al., Laminin-332 cleavage by matriptase alters motility parameters of prostate cancer cells, The Prostate. 71 (2011) 184-96.

[15] O. Klezovitch, J. Chevillet, J. Mirosevich, R.L. Roberts, R.J. Matusik, V. Vasioukhin, Hepsin promotes prostate cancer progression and metastasis, Cancer Cell. 6 (2004) 185-195.

[16] V. Srikantan, M. Valladares, J.S. Rhim, HEPSIN Inhibits Cell Growth / Invasion in Prostate Cancer Cells, Cancer Research. 62 (2002) 6812-6816.

[17] S.H. Zigmond, Ability of polymorphonuclear leukocytes to orient in gradients of chemotactic factors, The Journal of Cell Biology. 75 (1977) 606-616.

[18] R.R. Kay, P. Langridge, D. Traynor, O. Hoeller, Changing directions in the study of chemotaxis., Nature Reviews. Molecular Cell Biology. 9 (2008) 455-63.

[19] C. Janetopoulos, L. Ma, P.N. Devreotes, P.A. Iglesias, Chemoattractant-induced phosphatidylinositol 3,4,5-trisphosphate accumulation is spatially amplified and adapts , independent of the actin cytoskeleton, Pnas. 101 (2004) 8951-8956.

[20] I.C. Schneider, J.M. Haugh, Quantitative elucidation of a distinct spatial gradient-sensing mechanism in fibroblasts, The Journal of Cell Biology. 171 (2005) 883-92.

[21] S. Aznavoorian, M.L. Stracke, H. Krutzsch, E. Schiffmann, L.A. Liotta, Signal transduction for chemotaxis and haptotaxis by matrix molecules in tumor cells, The Journal of Cell Biology. 110 (1990) 1427-38.

[22] M.A. Sells, J.T. Boyd, J. Chernoff, p21-Activated kinase 1 (Pak1) regulates cell motility in mammalian fibroblasts, The Journal of Cell Biology. 145 (1999) 837-49.

[23] S.B. Carter, Haptotaxis and the mechanism of cell motility, Nature. 213 (1967) 256-260.

[24] E.K. Frow, J. Reckless, D.J. Grainger, Tools for anti-inflammatory drug design: in vitro models of leukocyte migration, Medicinal Research Reviews. 24 (2004) 276-98.

[25] J.T. Smith, J.K. Tomfohr, M.C. Wells, T.P. Beebe, T.B. Kepler, W.M. Reichert, Measurement of cell migration on surface-bound fibronectin gradients, Langmuir. 20 (2004) 8279-86.

[26] J.T. Smith, J.T. Elkin, W.M. Reichert, Directed cell migration on fibronectin gradients: effect of gradient slope, Experimental Cell Research. 312 (2006) 2424-32.

[27] L. Liu, B.D. Ratner, E.H. Sage, S. Jiang, Endothelial cell migration on surface-density gradients of fibronectin, VEGF, or both proteins., Langmuir. 23 (2007) 11168-73.

[28] D.S. Rhoads, J.-L. Guan, Analysis of directional cell migration on defined FN gradients: role of intracellular signaling molecules, Experimental Cell Research. 313 (2007) 3859-67.

[29] S.P. Palecek, J.C. Loftus, M.H. Ginsberg, D.A. Lauffenburger, A.F. Horwitz, P.P. Deshpande, et al., Integrin – ligand binding properties govern cell migration speed through cell – substratum adhesiveness, Nature. 388 (1997) 537-540.

[30] S.P. Holly, M.K. Larson, L.V. Parise, Multiple roles of integrins in cell motility, Experimental Cell Research. 261 (2000) 69-74.

[31] R.G. Harrison, The cultivation of tissues in extraneous media as a method of morphogenetic study, The Anatomical Record. 6 (1912) 181-193.

[32] P. Weiss, Experiments on cell and axon orientation in vitro: The role of colloidal exudates in tissue organization, Journal of Experimental Zoology. 100 (1945) 353-386.

[33] A. Curtis, C. Wilkinson, Topographical control of cells, Biomaterials. 18 (1997) 1573-83.

[34] P. Clark, P. Connolly, A.S.G. Curtis, J.A.T. Dow, C.D.W. Wilkinson, Topographical control of cell behaviour. I. Simple step cues, Development. 99 (1987) 439-48.

[35] P. Clark, P. Connolly, A.S.G. Curtis, J.A.T. Dow, C.D.W. Wilkinson, Topographical control of cell behaviour: II. Multiple grooved substrata., Development. 108 (1990) 635-44.

[36] A.I. Teixeira, G.A. McKie, J.D. Foley, P.J. Bertics, P.F. Nealey, C.J. Murphy, The effect of environmental factors on the response of human corneal epithelial cells to nanoscale substrate topography, Biomaterials. 27 (2006) 3945-54.

CHAPTER II

SPECIFIC AIMS

The overall goal of this dissertation research was to understand and quantify cancer cell motility. Because at present there are no tools to measure some of the biological aspects of cell motility we needed to quantify, we have developed a set of methods and assays which may be used to extract dynamic measures of motility at the individual cell level. This novel toolset we have developed may be used to understand motility in any adherent cell line. However, for this dissertation our specific focus was to understand the effect of hepsin over-expression on prostate cancer cell migration. We have achieved this through three specific aims.

Specific Aim I. CellAnimation: An Open Source MATLAB Framework for Microscopy Assays

In this aim we developed and tested a microscopy framework that can be used for cellular tracking with support for morphology and motility data extraction from time-lapse microscopy images. The framework supports mitotic event detection, ancestry recording and tracking of intracellular objects. A high-throughput mode is also supported that allows analysis of multiple time-lapse image sequences, such as those generated by a high-throughput microscope in a short amount of time. We have also compared the performance of our tracking algorithm with the

performance of other modern or popular tracking algorithms and shown that our algorithm improves on the current state-of-the-art.

Specific Aim II. Effects of Hepsin Overexpression in the LNCAP-34 Prostate Cancer Cell Line

For this aim we studied the differences in motility between low hepsin-expressing (LNCaP-17) and hepsin-overexpressing (LNCaP-34) prostate cancer cells. Our hypothesis is that the differences between the two cell lines are due to differences in the interaction of the cells with laminin-332. The software developed in Aim I was used to extract motility and morphology data for the two cell lines. From the motility data we calculated mean square displacement, speed and directional persistence. We have shown that significant differences exist between the two cell lines in both motility and morphology measures. We then tested our main hypothesis and confirmed that the two cell lines interact differently with laminin-332.

Specific Aim III. A Method for Determining Haptotaxis Efficiency in Cancer Cell Lines

For the final aim we developed a method to quantify the ability of cells to sense differences in substrate-bound densities of extracellular matrix proteins. This method was developed so that differences in directed motility, specifically haptotaxis, between the cells in aim II as well as other cancer cells may be investigated. It is our hypothesis that cells which migrate only occasionally have a weak ability to detect haptotaxis gradients. We demonstrate our method using HT1080 fibrosarcoma cells. We test our hypothesis by measuring the ability of the cells to detect various differences in density of surface-bound extracellular matrix protein.

CHAPTER III

CELLANIMATION: AN OPEN SOURCE MATLAB FRAMEWORK FOR MICROSCOPY ASSAYS[*]

Abstract

Motivation: Advances in microscopy technology have led to the creation of high-throughput microscopes that are capable of generating several hundred gigabytes of images in a few days. Analyzing such wealth of data manually is nearly impossible and requires an automated approach. There are at present a number of open source and commercial software packages that allow the user to apply algorithms of different degrees of sophistication to the images and extract desired metrics. However, the types of metrics that can be extracted are severely limited by the specific image processing algorithms that the application implements, and by the expertise of the user. In most commercial software, code unavailability prevents implementation by the end user of newly developed algorithms better suited for a particular type of imaging assay. While it is possible to implement new algorithms in open source software, rewiring an image processing application requires a high degree of expertise. To obviate these limitations, we have developed an open source high-throughput application that allows implementation of different biological assays, such as cell tracking or ancestry recording, through the use of small, relatively simple image processing modules connected into sophisticated imaging pipelines. By connecting modules, non-expert users can apply the particular combination of well-established and novel algorithms developed by us and others that are best suited for each individual assay type. In addition, our data exploration and visualization

[*] Adapted from Georgescu W, Wikswo JP, Quaranta V. *Oxford Bioinformatics*. 28 (2012) 138-9.

modules make it easy to discover or select specific cell phenotypes from a heterogeneous population.

Contact: walter.georgescu@vanderbilt.edu

Introduction

At present there are a number of microscopy applications on the market, both open source and commercial, that address both very specific and broader microscopy needs. In general, commercial software packages such as Imaris® and MetaMorph® tend to include more complete sets of algorithms, covering different kinds of experimental setups, at the cost of ease of customization and integration with other software. Open source packages such as CellCognition [1], CellTracer [2], CellTracker [3], Cell-ID [4] or Starrynite [5] are easier to customize and integrate well with other software packages for a complete microscopy pipeline. However, these applications are designed for a specific task or set of tasks and cannot easily be made to run other assays. For example, CellCognition, an application designed for high throughput tracking of fluorescent cells, cannot easily be repurposed to perform a colony counting assay.

Two microscopy applications that offer a high degree of customization are CellProfiler [6] and ImageJ [7]. CellAnimation is similar to CellProfiler with respect to the modular architecture and high-throughput options. However, CellProfiler is an application whose main purpose is not tracking cells and as such it does not have modules to detect mitotic events, track cells under brightfield illumination or perform tracking correction, or virtual staining capabilities. ImageJ offers tracking functionality and other assays through a series of plugins, but is not designed for high-throughput cell tracking or cell ancestry recording.

We have chosen to implement CellAnimation in MATLAB because MATLAB is optimized to work with matrices, it includes a large number of built-in image processing functions and it has been used extensively in the image processing community. This has led to the availability of a large number of thresholding, segmentation and object tracking algorithms. CellAnimation is a framework that provides users with the tools to perform high-throughput cell tracking and ancestry recording, but is suitable for a number of other microscopy assays such as colony and cell counting, focal adhesion tracking, etc. We provide a set of modules (currently >60) and pipelines designed for high-throughput tracking of different types of cells under multiple types of illuminations and with the ability to quickly and easily replace parts of these pipelines with higher-performing modules from internal and external sources as they become available. In addition, we provide a set of segmentation and tracking correction modules that allow correction and validation of automated tracking assays. Finally, our data visualization module allows the selection of multiple cell sub-populations based on shape and motility parameters and the creation of multiple virtual stains that highlight specific sub-populations.

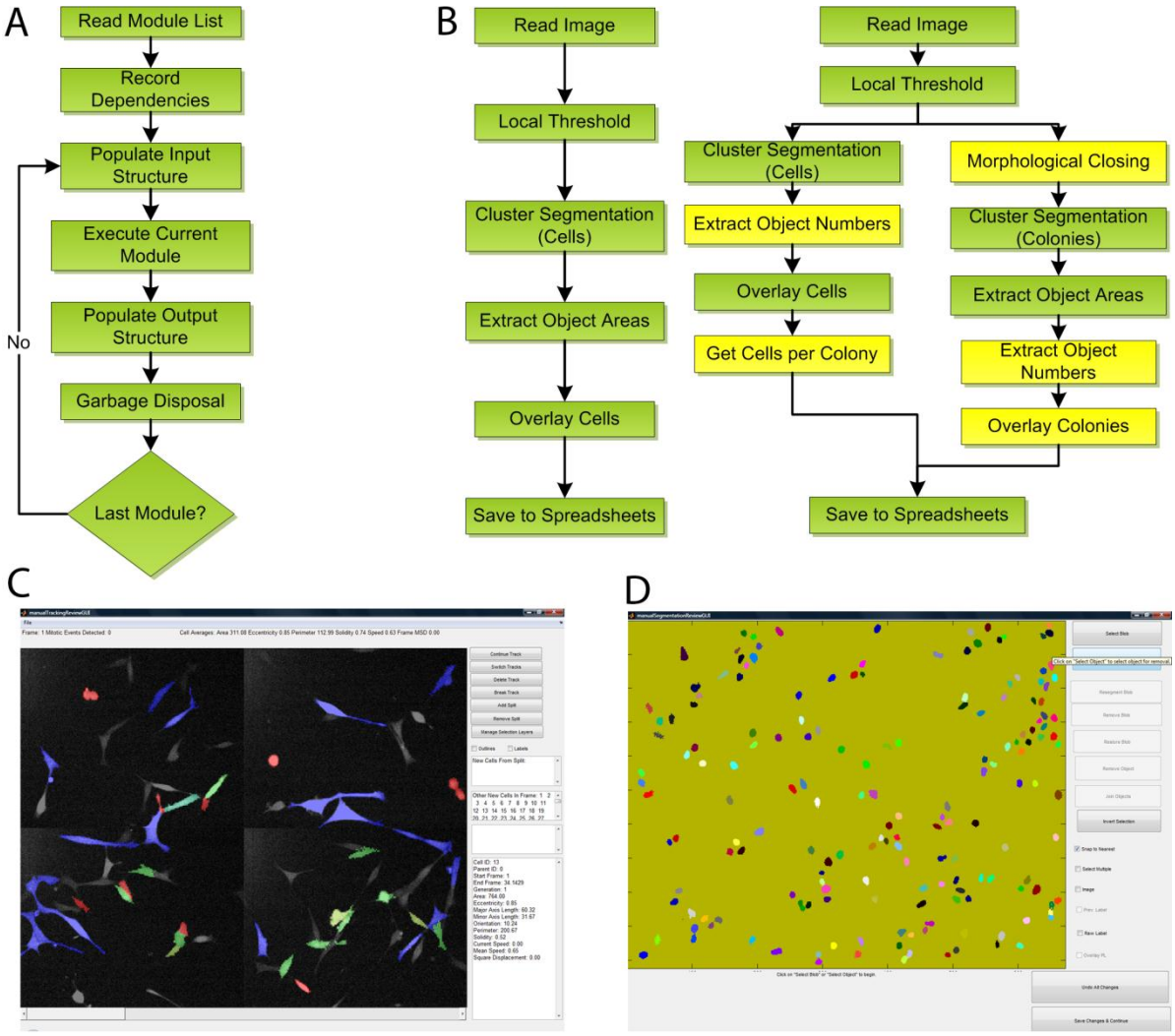

Figure 3.1 A) Assay processing by the CellAnimation core functions. The CellAnimation core functions are responsible for reading the module chain, creating a dependency tree, populating the input values, executing each module and saving those output values required by modules further downstream. B) The modular structure of CellAnimation assays allows for very efficient code reuse. The modules from one assay, such as a "cell spreading assay" (green), can be completely reused to create a "colony counting assay" by inserting a few new modules (yellow). C) Visualization module GUI. Three selection layers have been defined: Top 20% area (blue), bottom 20% area (red) and top 20% motility (green). Cell population statistics are presented at the top, single cell statistics on the right. Track review buttons are on the top right. D) Segmentation review module. Errors in segmentation or thresholding may be corrected and recorded using the GUI.

We have been using this toolbox in our lab to perform different types of imaging assays both in stand-alone and in a high-throughput mode [8,9].

Implementation

CellAnimation was developed using MATLAB R2007b and requires the Image Processing Toolbox. We have run our assays on version R2007b and newer versions (up to R2010b) on Windows, Linux and Mac OS. To allow for fast assay creation and code reuse, each assay is implemented as a modular pipeline. A CellAnimation assay is a chain of MATLAB structures. Each structure describes a module and its connectivity. The CellAnimation core functions are responsible for reading the module chain, creating a dependency tree, populating the input values, executing each module and saving those output values required by modules further downstream (Fig. 3.1A). A module is a reusable set of functions that has a narrow specific use (image input-output, thresholding, segmentation, annotation, etc.). We have also included a set of control modules. These are special modules that operate on other modules. Through the use of control modules we can implement looping and branching at the pipeline level. The benefits to this approach are two-fold. First, the assay logic becomes easier to follow and modify. Second, it allows us to use smaller, more reusable modules.

The modular structure of CellAnimation assays allows for very efficient code reuse. The modules from one assay, such as a cell spreading assay, can be completely reused to create a colony counting assay (Fig. 3.1B). Replacing a module in an assay can be done in a matter of minutes by an experienced user. Modifying existing assays or creating new assays from scratch can be easily done using a graphical user interface we provide as part of the framework.

Once an assay has been optimized, it can be distributed to a computer cluster for high-throughput processing. Using 60 nodes of the Vanderbilt Advanced Computing Center for Research and Education (ACCRE) cluster, in less than two hours we can analyze 60 movies captured over the course of several days and each consisting of hundreds of frames. Since at

present it takes about eight hours to transfer the ~100 GB of image data to and from the computing cluster, the speed with which we can analyze movies is limited by network transfer times instead of image processing times!

The performance of assays can be evaluated against manually validated images. To assist with manual validation we have included modules for segmentation and tracking review (Figs. 3.1C and 3.1D). The tracking review module is part of a larger cell-centered image exploration module. The module provides a set of tools that help users explore the patterns and clusters that may be present in the cell population, in addition to correcting errors in automatic tracking. Using the image exploration module a user can explore single cells and population shape and motility parameters or highlight a sub-population of cells using selection layers (Fig. 3.1C). A selection layer is a transparency overlaid on the original image that highlights certain cells based on a user-defined criterion. The criterion for comparison may be an exact value (all cells with an area larger than 500) or a percentage (cells with an area in the top 20%). Any combination of shape, motility and ancestry parameters may be used alone or in combination to define a layer.

Processing Steps in a Typical Assay

The MATLAB code shown here illustrates an assay designed to identify cells in a brightfield image. The result of the module processing on the input image is presented below each module. The resulting cell outlines are stored in a MATLAB label matrix. Cell shape statistics can be easily extracted from a label matrix using the regionprops function. In the code TrackStruct is a structure holding all the user-defined constants.

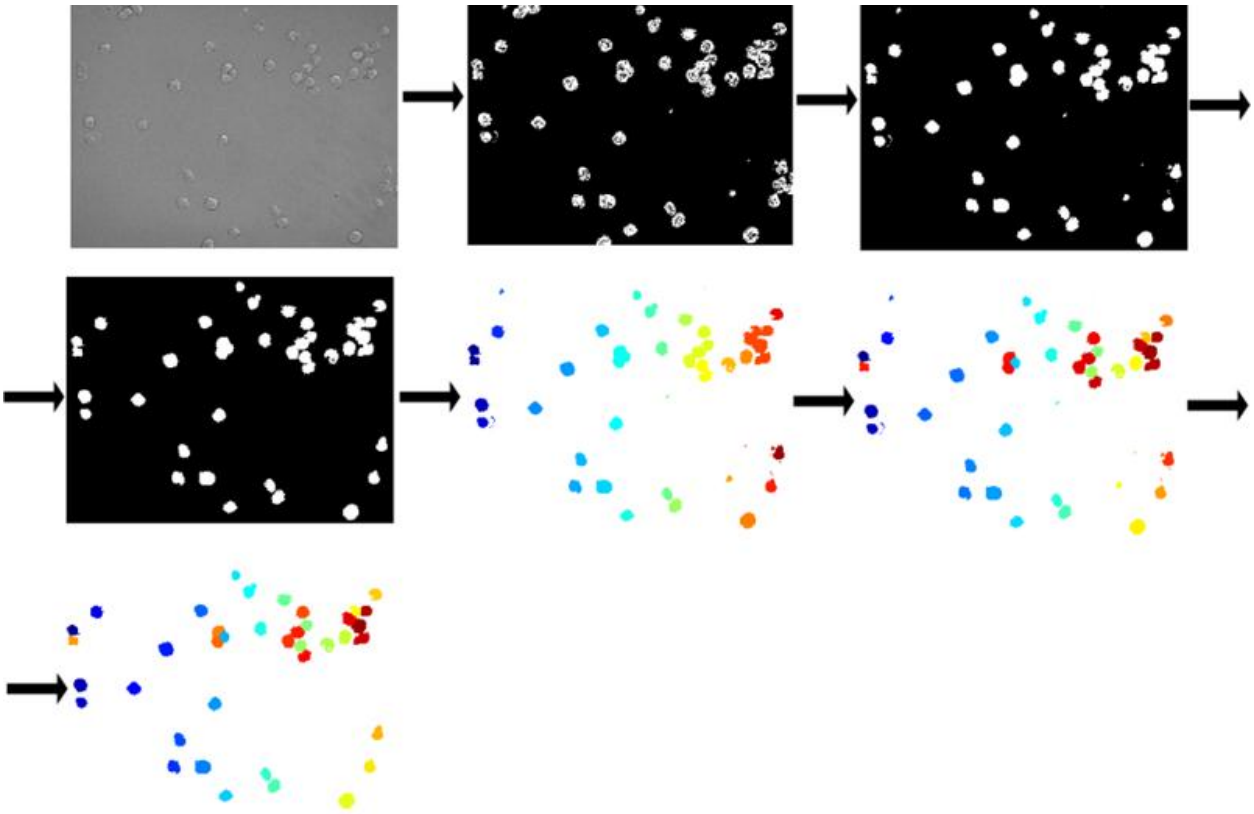

Figure 3.2. An overview of the assay sequence. The image file, in this case a grayscale image, is loaded into MATLAB. Subsequently, a gradient filter is used to identify the cells, the cell outlines are filled in, small artifacts are removed from the image, cell clusters are labeled and segmented into individual cells (which are given separate colors), and selected for size. Finally, at the end of the assay individual cell properties such as area, eccentricity, solidity, etc. can be extracted, in this example for size above a particular threshold

%read image module

read_image_function.InstanceName='ReadImage';

read_image_function.FunctionHandle=@readImage;

read_image_function.FunctionArgs.ImageName.Value=TrackStruct.ImageName;

read_image_function.FunctionArgs.ImageChannel.Value='';

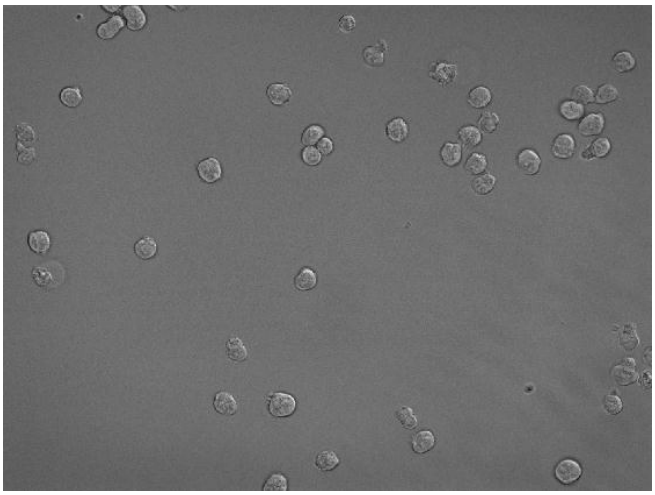

Figure 3.3. The readImage module can load images saved as tiff or jpeg into a MATLAB matrix. For this example we are using cells imaged with bright field microscopy.

%gradient filter module

grad_filter_function.InstanceName='CytoGradientLocalFilter';

grad_filter_function.FunctionHandle=@generateBinImgUsingGradient;

grad_filter_function.FunctionArgs.Image.FunctionInstance='ReadImage';

grad_filter_function.FunctionArgs.Image.OutputArg='Image';

grad_filter_function.FunctionArgs.GradientThreshold.Value=TrackStruct.GradThresh;

grad_filter_function.FunctionArgs.ClearBorder.Value=true;

grad_filter_function.FunctionArgs.ClearBorderDist.Value=TrackStruct.BorderDist;

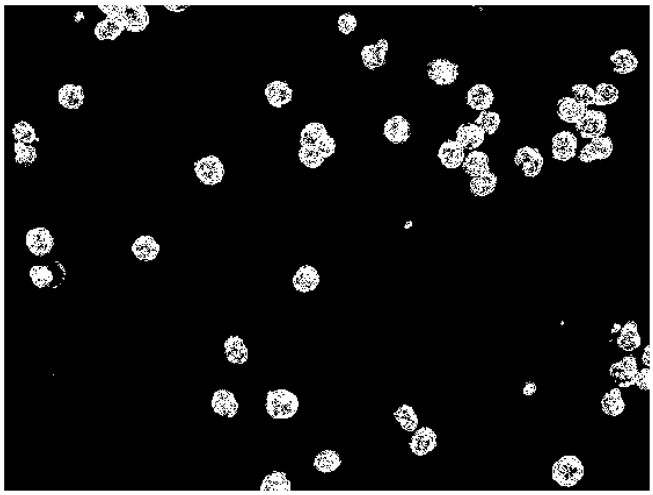

Figure 3.4. The generateBinImgUsingGradient module uses the image gradient to detect the cell outlines. A binary image is created by setting areas with high values of the gradient to 1 and everything else to 0.

%fill holes module

fill_holes_cyto_images_function.InstanceName='FillHolesCytoplasmImages';

fill_holes_cyto_images_function.FunctionHandle=@fillHoles;

fill_holes_cyto_images_function.FunctionArgs.Image.FunctionInstance='CytoGradientLocalFilt

er';

fill_holes_cyto_images_function.FunctionArgs.Image.OutputArg='Image';

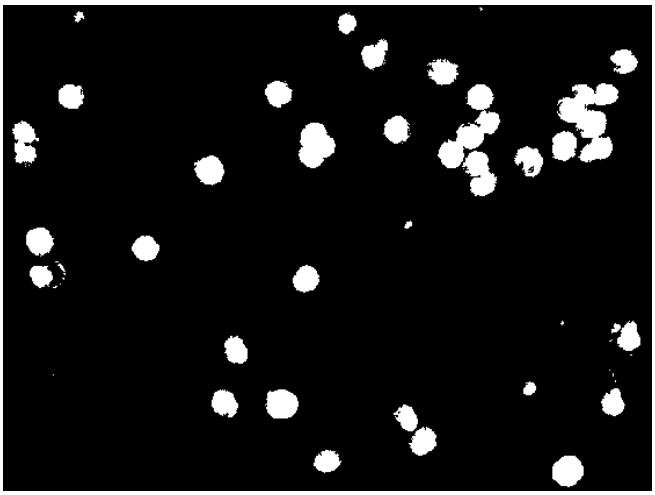

Figure 3.5. The fillHoles module replaces any background pixels inside the cell contours with foreground pixels.

%remove small objects module

clear_small_objects_function.InstanceName='ClearSmallObjects';

clear_small_objects_function.FunctionHandle=@clearSmallObjects;

clear_small_objects_function.FunctionArgs.Image.FunctionInstance='FillHolesCytoplasmImag

es';

clear_small_objectss_function.FunctionArgs.Image.OutputArg='Image';

clear_small_objects_function.FunctionArgs.MinObjectArea.Value=TrackStruct.MinCytoArea;

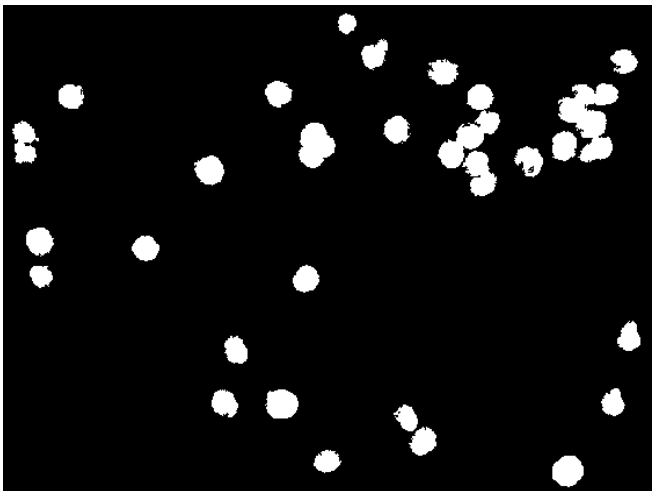

Figure 3.6. The clearSmallObjects module removes any objects with an area smaller than a specified threshold area.

%create label matrix module

label_cyto_function.InstanceName='LabelCytoplasm';

label_cyto_function.FunctionHandle=@labelObjects;

label_cyto_function.FunctionArgs.Image.FunctionInstance='FillHolesCytoplasmImages';

label_cyto_function.FunctionArgs.Image.OutputArg='Image';

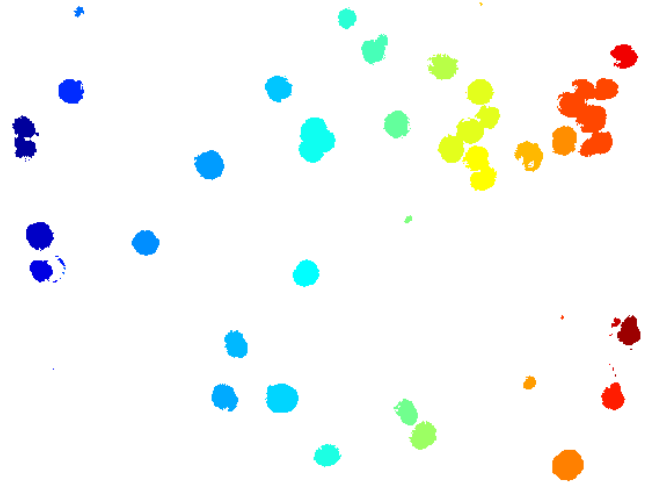

Figure 3.7. The labelObjects module generates a matrix where each cell cluster has a unique integer ID. Here each cluster has been displayed in a different color using the MATLAB label2rgb function.

%segment objects using clusters module

segment_objects_using_clusters_function.InstanceName='SegmentObjectsUsingClusters';

segment_objects_using_clusters_function.FunctionHandle=@segmentObjectsUsingClusters;

segment_objects_using_clusters_function.FunctionArgs.ObjectsLabel.FunctionInstance='Label Cytoplasm';

segment_objects_using_clusters_function.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';

segment_objects_using_clusters_function.FunctionArgs.ObjectReduce.Value=TrackStruct.ObjectReduce;

segment_objects_using_clusters_function.FunctionArgs.MinimumObjectArea.Value=TrackStruct.MinNuclArea;

segment_objects_using_clusters_function.FunctionArgs.ClusterDistance.Value=TrackStruct.ClusterDist;

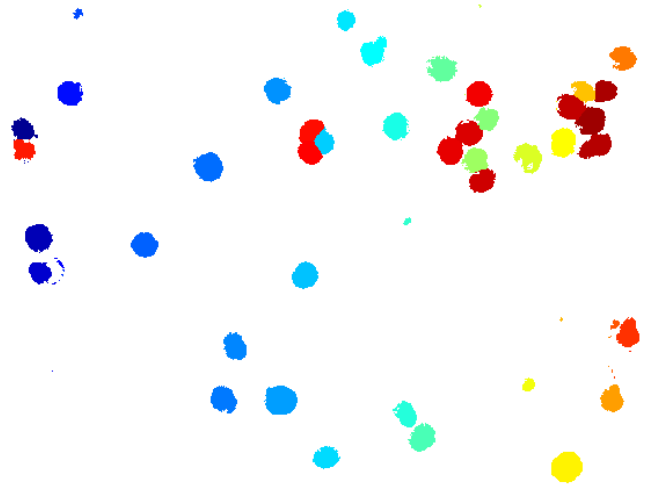

Figure 3.8. The segmentObjectsUsingClusters module uses hierarchical clustering to segment the cell clusters into individual cells. The label matrix data have been color coded using the label2rgb function.

%area filter module

area_filter_function.InstanceName='AreaFilter';

area_filter_function.FunctionHandle=@areaFilterLabel;

area_filter_function.FunctionArgs.ObjectsLabel.FunctionInstance='SegmentObjectsUsingCluste

rs';

area_filter_function.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';

area_filter_function.FunctionArgs.MinArea.Value=TrackStruct.MinCytoArea;

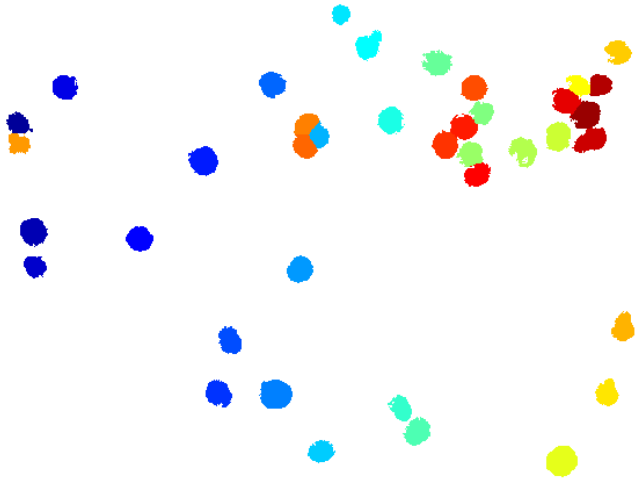

Figure 3.9. The areaFilterLabel module removes objects with an area smaller than a specified threshold area. The label matrix objects have been color coded using the label2rgb function.

%define module sequence

functions_list=[{read_image_function};

{ grad_filter_function};{fill_holes_cyto_images_function};

{clear_small_objects_function};{label_cyto_function};{segment_objects_using_clusters_functio

n};{area_filter_function}];

%define global variables

global dependencies_list;

global dependencies_index;

dependencies_list={};

dependencies_index=java.util.Hashtable;

%create dependency table

makeDependencies([]);

%run the pipeline

runFunctions();

Assay Editor

Usage

The assay editor is used to create new assays and edit existing assays. To start the assay editor make sure the current directory in MATLAB is set to the directory containing the CellAnimation source files and type *assayEditorGUI* at the MATLAB command prompt.

Main Window

The main window of the assay editor displays all the available CellAnimation modules, the modules in the current assay and a description of the currently selected assay. The available modules are displayed in the top left corner listbox. Clicking on a module in the listbox shows its description in the textbox below. The listbox on the right shows the modules in the current assay. Modules whose names are in a bold font are control modules and have their own submodules.

The submodule lists are indicated by an indented italic font immediately below the name of the

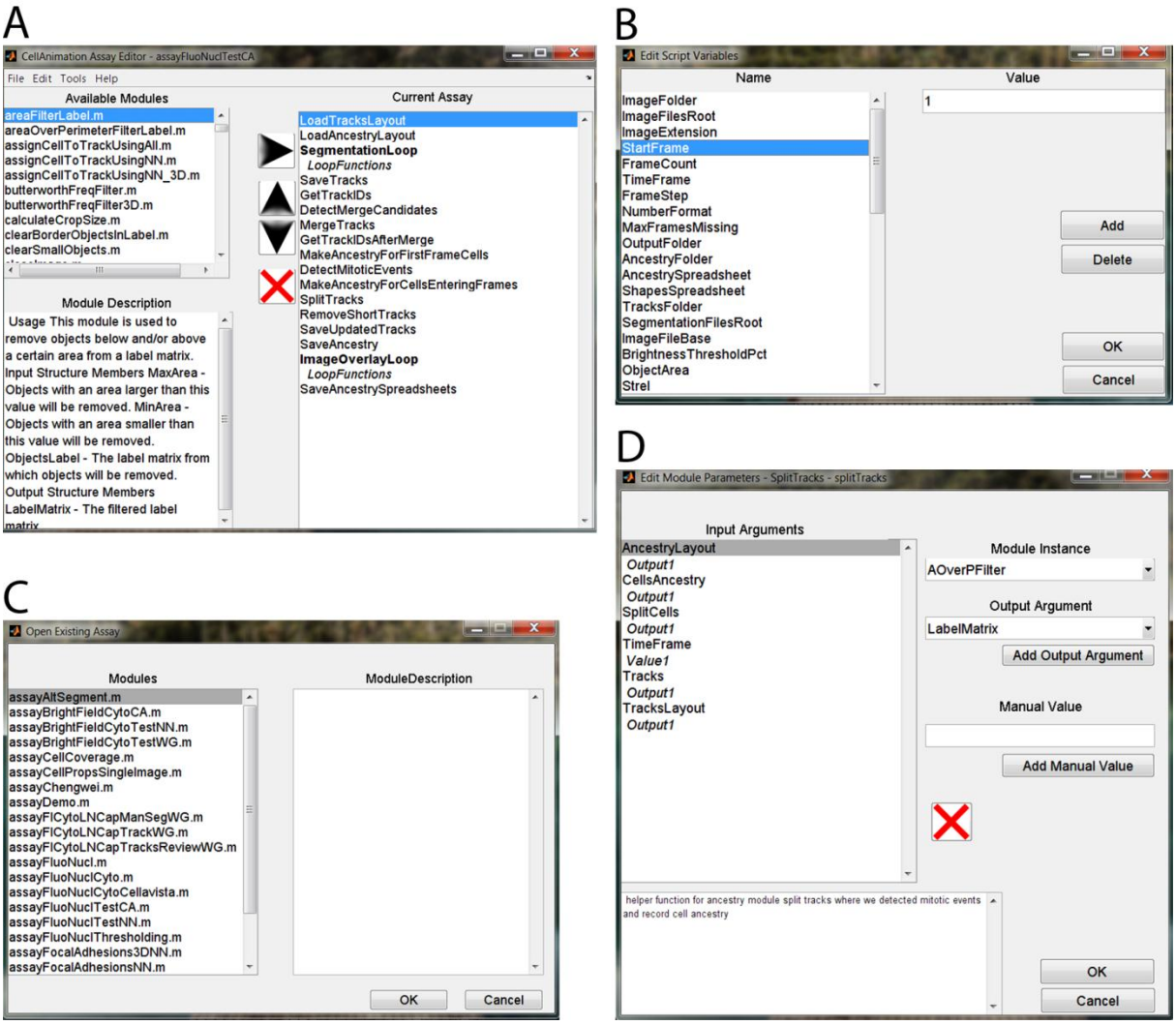

Figure 3.10. CellAnimation Assay Editor. A) Main window of CellAnimation assay editor. B) Modal dialog box to edit script variables. C) Modal dialog box to open an existing assay. D) Dialog box used for editing module parameters.

control module. Double-clicking on a submodule list expands it so the submodules it contains become visible. Double-clicking the submodule list again collapses it. Double-clicking a module name in the *Current Assay* listbox opens the *Edit Module Parameters* dialog box.

Editing an Assay

Open the assay to be edited using *File|Open.* In the *Open Existing Assay* dialog box the listbox on the left shows the available assays while the textbox on the right shows a description of the currently selected assay. To open an assay, double-click its name or select it and then click the *OK* button. Once an assay is opened its main modules become visible in the current assay listbox. There are three different levels of assay editing. Most users will only need to change script parameters of an existing assay to get the assay to work with their data set.

To modify a script variable, click on *Edit|Script Variables*. The *Edit Script Variables* dialog box displays the script variables listed in order of change frequency in the *Name* listbox. Variables whose values are likely to be changed often are displayed at the top of the list while variables that are unlikely to change are displayed towards the bottom of the list. A description of each variable can be found in the assay .m file or in this help file. To change the value of a variable, click on its name, then type a new value in the *Value* textbox on the right.

Advanced users can change values of variables at the individual module level. To change the value of module variables, double-click the desired module level in the *Current Assay* listbox in the main window. This opens the *Edit Module Parameters* dialog box, which displays the input values to the module along with the providers of those values. Each input argument name is displayed in regular font in the *Input Arguments* listbox. Below each argument name is a list of providers for that argument in indented italic font. Input arguments for which no provider has been yet specified are displayed in red. In general each input argument needs at least one provider with the exception of optional arguments. There are two types of providers and they are indicated by the prefix *Value* or *Output*.

Providers starting with the prefix *Value* are constant values that are entered manually by the user. To change a static value, click on a static provider name (usually *Value1*) and type a new value in the *Manual Value* textbox. The value is automatically saved when selecting something else in the *Input Arguments* box or clicking the *OK* button. String values such as file names must be surrounded by single quotation marks. Text entered without quotation marks is interpreted either as a script variable name or a number, if the text consists only of digits. To add a new static value provider, click on the input argument name, enter the value in the *Manual Value* textbox, then click on the *Add Manual Value* button.

Providers starting with the prefix *Output* are dynamic values that are provided at runtime by another module. To modify the provider for a dynamic value, click on a dynamic provider name (*Output1*, *Output2*, etc.) then select another module from the *Module Instance* popup box and/or another output argument from the *Output Argument* popup box. To add a new dynamic provider, click on the input argument name for which you wish to add the provider, select the appropriate module instance and output argument from the popup boxes on the right, then click on *Add Output Argument*.

It is not unusual for an input argument to have both a static and a dynamic provider, with the dynamic provider downstream from the module receiving the value. The static provider is used as an initial value, while subsequent values are obtained from the dynamic provider and change as the assay iterates. The values from the dynamic provider overwrite the static ones as soon as the module providing the value is called.

Creating a New Assay

To create a new assay, click on *File|New*. To add a module to the assay select it in the *Available Modules* listbox, then click on the button displaying an arrow pointing to the right. In

the dialog box that appears type an instance name for the current module. The instance name must be a unique name that is not used by any other modules in the assay. You may also select a parent module from the popup box. If the parent module is blank the module will be added as a main module. If the parent module is not blank the module will be added as a submodule belonging to a control module. The choice between a main module and a submodule depends on the intended purpose of the module. A main module will be called exactly once while a submodule may be called multiple times if it's part of a control module that iterates, such as a forLoop module. Once a module instance has been filled in, click the *OK* button. A module that has been added by mistake can be removed by selecting it in the *Current Assay* listbox, then clicking on the button displaying the red *X* shape.

After the module has been added, providers need to be specified for each of its input arguments. A provider is a value specified either statically when the assay is edited or provided dynamically by one of the modules when the assay is run. To specify providers, double-click on the module, then specify dynamic or static parameters as described in the "Editing an Assay" section.

Modules will be executed in the order they appear in the current assay listbox so you need to make sure that each module appears in its proper place. For example, a *readImage* module should appear above any modules that operate on that image. To change the position of a module in the assay, select it then click either the up arrow button or the down arrow button to move the module up or down respectively. As the module is moved it may become part of a submodule list or exit a submodule list and become a main module instead. This is indicated by the indent level of the module. Main modules are not indented from the left side of the *Current*

Assay listbox while submodules are indented one or more spaces from the left side depending on the level of their parent module.

If a parameter is common to several modules or it's likely to change every time an assay is run we recommend you add it as a script variable. To add a script variable, click on *Edit|Script Variables*. In the *Edit Script Variables* dialog box click on the *Add* button. Type the name of the new script variable in the *Name* textbox, then click the *OK* button. This name should be different from any existing script variables belonging to this assay. Type the value of the new script variable in the *Value* textbox, then click on the *OK* button or another script variable name to save it. In the same way as with module variables, a string literal must be enclosed within single quotation marks or it will be interpreted as a script variable name or a number if it consists only of digits. An example of a script variable may be an image file name that is used both by the *readImage* module and the *saveImage* module and is likely to change every time the assay is run. Instead of editing the file name every time the assay is run in both modules, one may add a script variable named *ImageFileName* and set its value to a string literal containing the image file name, for example *'C:/test/image1.jpg'*. Then in the modules, instead of entering the string literal one can simply enter *ImageFileName* and the assay will retrieve the value from the script variable. The value of the script variable can be easily changed when the assay is run on another image as described in "Editing an Assay" and both modules will receive the updated value.

Finally, we recommend adding an assay description so others can easily see the purpose of the assay when they click on it. An assay description can be added by clicking on *Edit|Assay Description* and typing the assay description in the *Assay Description* textbox. Once all the modules are in their proper positions and the input parameters have at least one provider, save

the assay using *File*|*Save*. A saved assay may be run using *File*|*Run* or by typing the assay name at the MATLAB command prompt.

Exporting Script Variables

Once an assay has been optimized for a particular set of images we recommend exporting the set of optimized script variables to the directory containing the images using *Tools*|*Export Script Variables*. This way if the analysis needs to be repeated at a later date it can be easily done by importing the optimized variables from the image directory using *Tools*|*Import Script Variables*.

Incorporating External MATLAB Functions

At times functionality may be required that is not available as a CellAnimation module but is available in some MATLAB function written by others. The assay editor makes it easy to access such functions via a CellAnimation wrapper module. The function which is to be made available to CellAnimation needs to be in a folder that is part of the MATLAB path. The MATLAB path can be modified using *File*|*Set Path* from the main MATLAB window. To wrap an external function in a CellAnimation module, click on *Tools*|*Wrap MATLAB Function*. Select the MATLAB function to wrap using the dialog box. A wrapper module will be automatically generated and a dialog box presented to the user that allows selection of the directory and file name where the wrapper module is to be saved. We recommend using the automatically generated file name as it makes it easy to recognize the module as a wrapper module. The wrapper module needs to be saved in the directory where the CellAnimation source files reside.

Popular Assays

Cell Spreading Assay

This assay is used to compute cell areas from individual microscopy images. Cell area is a proxy for the strength of interaction between cells and substrate; as an example, this assay can be used

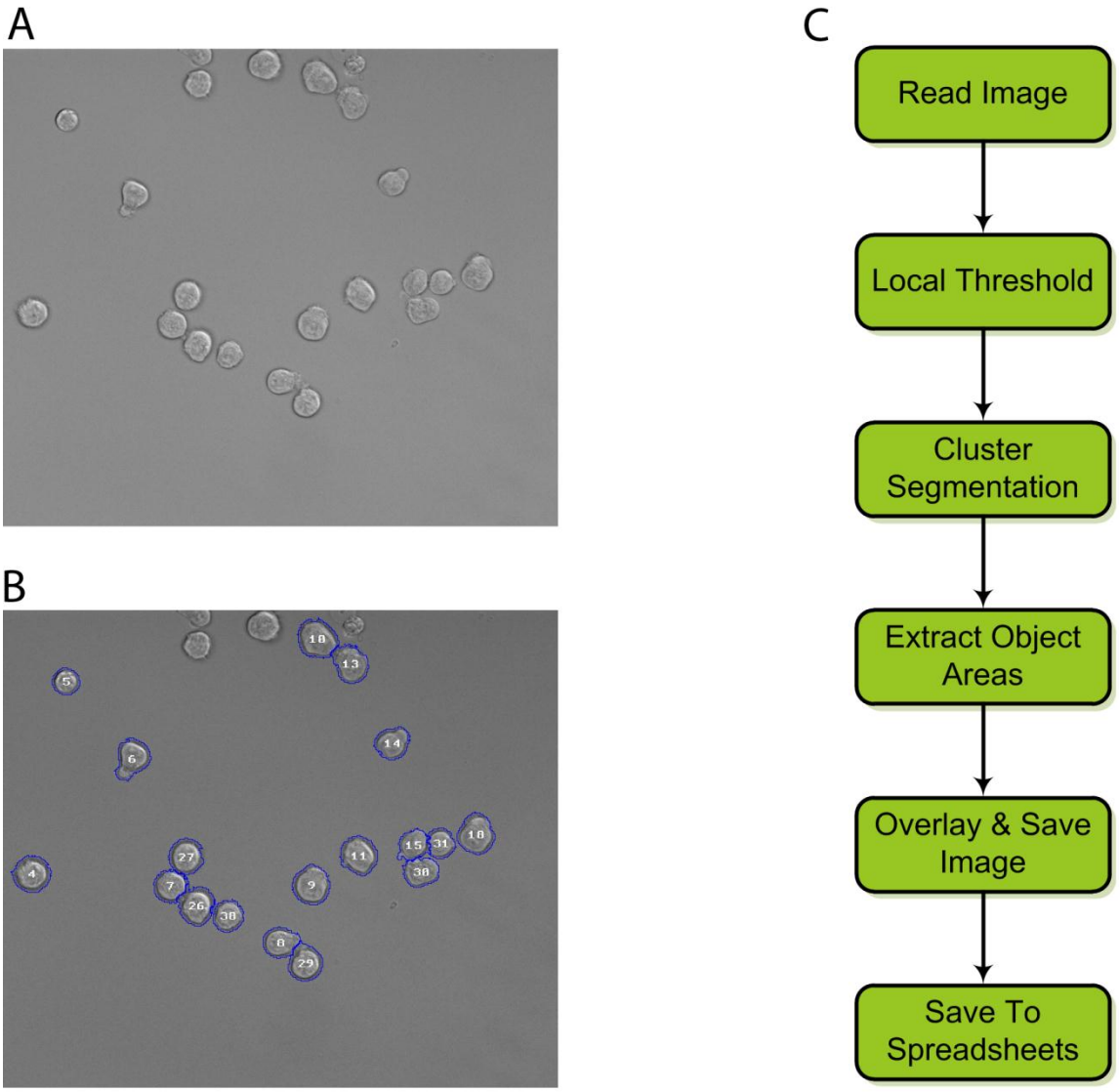

Figure 3.11. Cell spreading assay. A) Typical bright field input image (courtesy of Yoonseok Kam). B) Output image showing detected cell outlines and cell IDs. Cells at edges are excluded to prevent incorrect area measurements. C) Flowchart of major modules in this assay (see text for description of modules).

to screen for synthetic biomaterials in comparison to their natural counterpart. Images may be either bright field images (Figure 3.11A) or fluorescence images although we recommend fluorescence images for improved detection of the cytoplasm. To separate cells from the background the assay uses a local thresholding module. For bright field images we use the magnitude of the image gradient to detect foreground objects. For fluorescence images we compare pixel intensity with the local average of intensity in the region. Pixels with intensity values higher by a certain percentage than the local intensity average are assigned to the foreground while all other pixels are assigned to the background. By using local thresholding as opposed to global thresholding, the areas extracted are not distorted due to uneven illumination across the image. After thresholding we use a clustering module to detect individual cell outlines. Once individual cell contours have been detected, the object areas are calculated and the individual cell outlines along with cell IDs are overlayed on the original image and saved as a separate image for manual validation (Figure 3.11B). Finally computed areas, along with the unique cell IDs and centroid coordinates are exported to a comma-separated text file.

Fluorescent Nuclei Tracking Assay

This assay is used to track moving cells, detect mitotic events and record tracking data, cell shape parameters and ancestry information to spreadsheets. It can provide a complete history of every single cell in a culture, in high-throughput fashion; for instance, it can be used to accurately evaluate the effects of extracellular perturbations (e.g., drug addition) with respect to cell motility, proliferation and survival, by deriving cumulative cell population level effects from single cell data. Cells are labeled with a genetic or non-genetic nuclear stain (Figure 3.12A). Two parameters which can affect the results of cell tracking are cell density and cell displacement

between frames. For optimum performance we recommend imaging at a frame rate that allows most cells to only move a short distance between frames preferably less than a cell diameter from one frame to the next. Cell density can be a factor with cells that are dense and motile such that many cells are being displaced by other cells between frames. This can be fixed by maintaining an adequate frame rate so there is little cell displacement between frames.

To separate nuclei from the background the assay uses a local thresholding module. The intensity value of each pixel in the image is compared with the local average of intensity in a small region around the pixel. Pixels with intensity values higher by a certain percentage than the local intensity average are assigned to the foreground while all other pixels are assigned to the background. By using local thresholding as opposed to global thresholding, the areas extracted are not distorted due to uneven illumination across the image. The foreground objects are then segmented into individual nuclei using a convexity-restricted watershed module. This is described in detail in the tracking chapter. Briefly, nuclei are round objects with no significant convexities in their outlines. The presence of a significant convexity indicates multiple nuclei touching. We allow the watershed segmentation to only operate on objects with significant convexities to prevent oversegmentation. Once the nuclei have been segmented, the assay uses a set of shape filters to eliminate any objects that are too large, too small or too elongated to be nuclei. The filtered objects are then assigned to tracks using either a nearest-neigbor algorithm (faster, but less accurate) or our custom tracking algorithm described in the next chapter. Once cells have been tracked and mitotic events have been detected the tracking and ancestry data are written to comma-separated text files. For each tracked image, an overlay image is saved to disk that displays the detected cell outlines, track IDs and cell generation (Figure 3.12B). This set of images may be used to manually validate the automatically detected cell tracks.

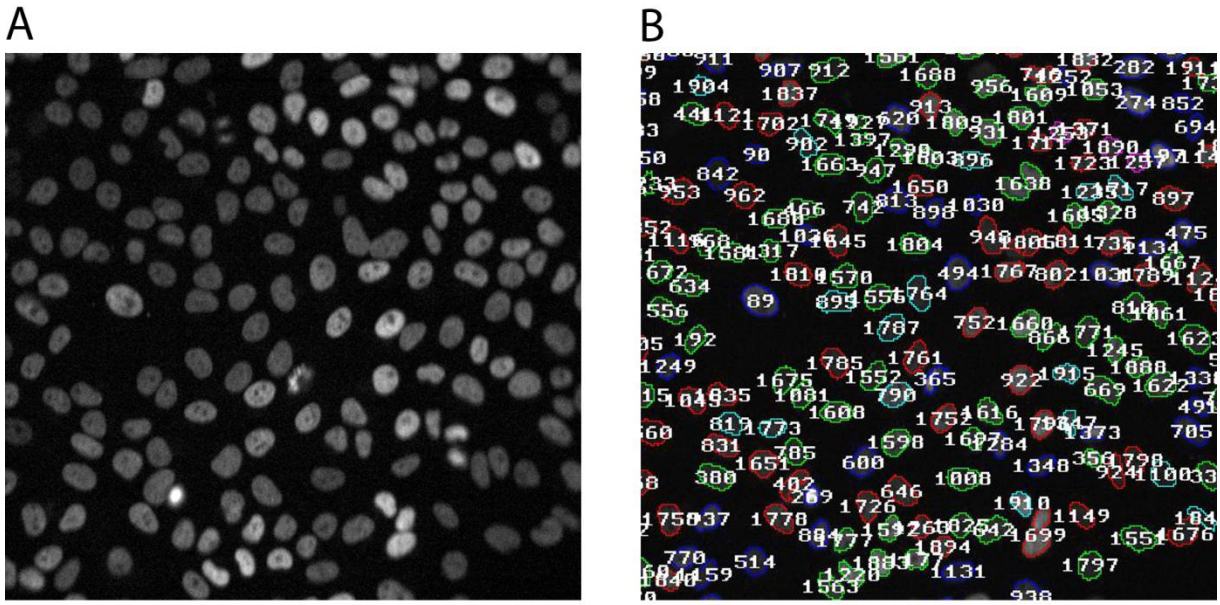

Figure 3.12. Fluorescent nuclei tracking assay. A) Typical input image. B) Output image showing detected nuclei outlines color-coded by cell generation and track IDs.

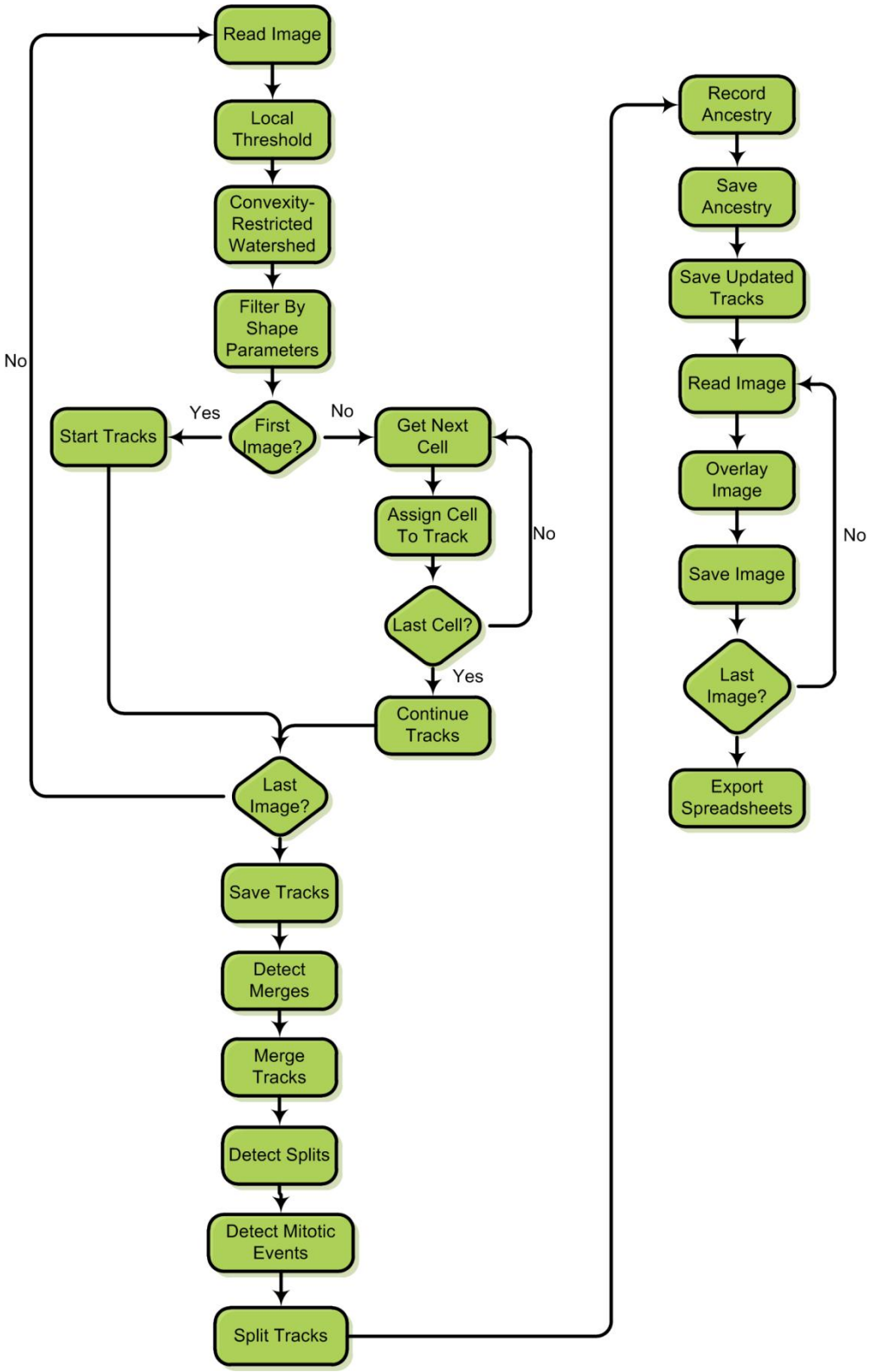

Figure 3.13. Flowchart of major modules in fluorescent nuclei tracking assay.

Focal Adhesion Turn-over Assay

This assay tracks focal adhesions of adherent cells in defined regions of interest, e.g., at the cell periphery. It is typically used to measure focal adhesion turn-over, which is a measure of cytoskeletal dynamics in response to perturbations. Hundreds of focal adhesions per cell may be tracked in both 2D and 3D. Focal adhesion structures must be labeled by expression of focal adhesion component proteins fused to naturally fluorescent proteins (e.g., green fluorescent protein). We recommend that cells be plated on plates with imaging quality plastic or on a glass substrate to ensure accurate quantification of fluorescence intensity. The frame rate should be high enough to prevent most focal adhesions from traveling more than a diameter from their positions in the previous frame.

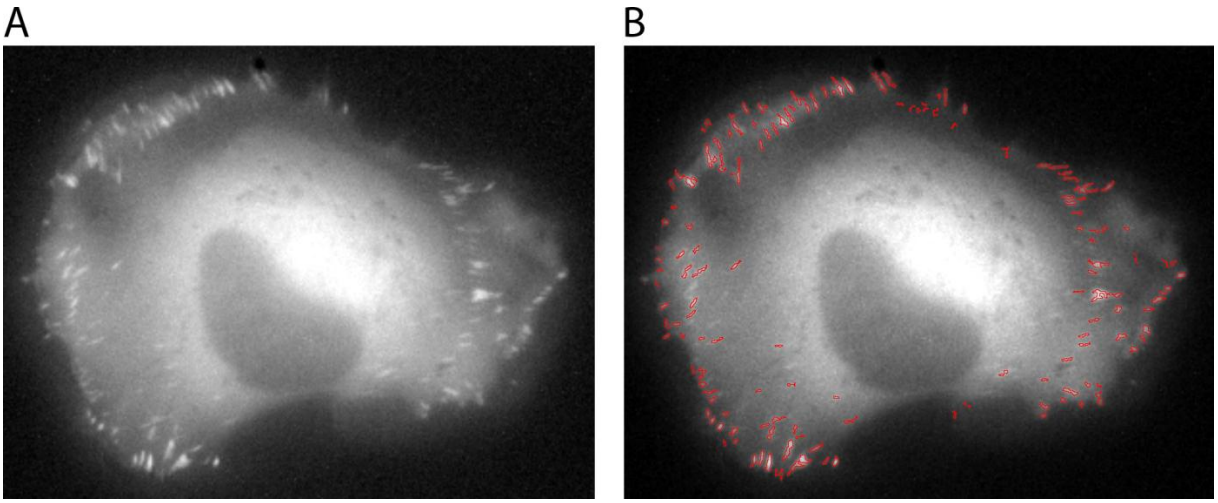

Figure 3.14. Focal adhesion assay. A) Typical input image showing fluorescent focal adhesions (courtesy of Donna Webb). B) Output image where detected focal adhesions have been outlined in red.

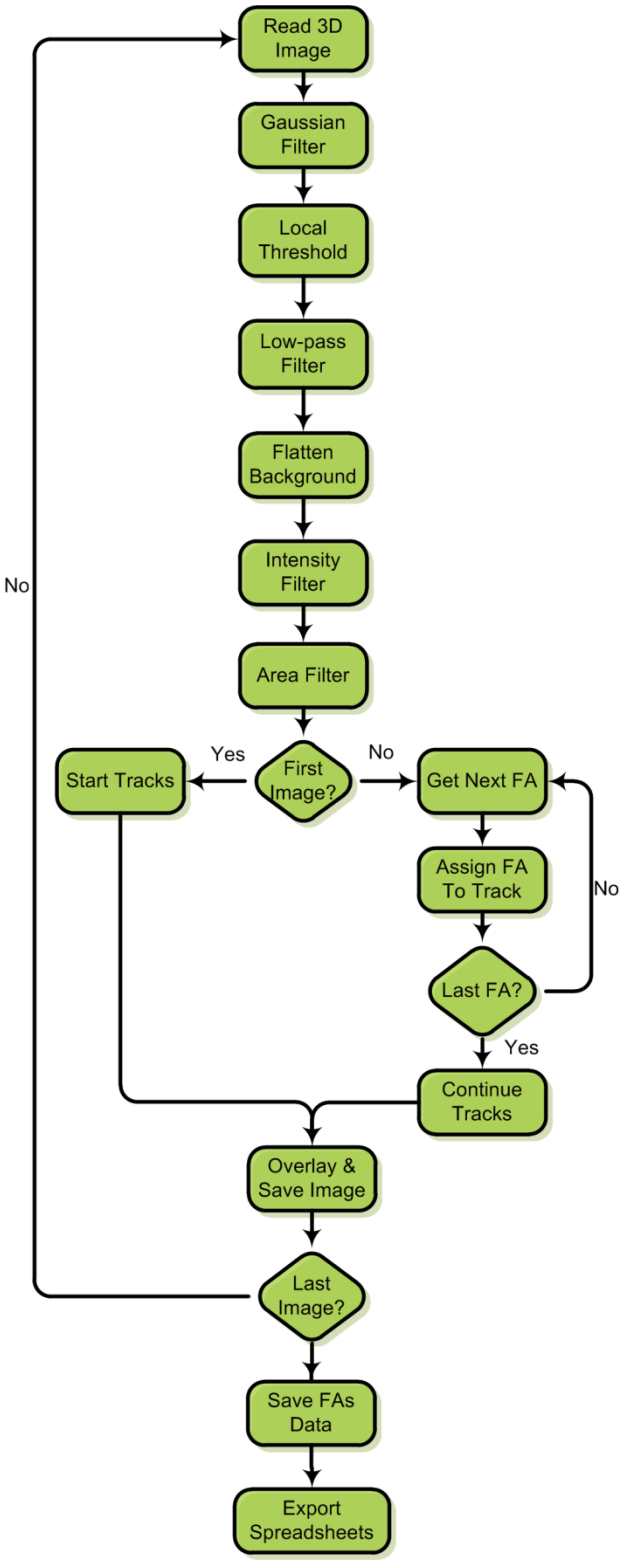

Figure 3.15. Flowchart of major modules in focal adhesion assay.

The assay applies a Gaussian filter to the input images to smooth out the background and prevent noise pixels from being detected as focal adhesions. A local thresholding of the fluorescence intensity as described previously is used to separate focal adhesions from the background. To prevent variations in light intensity from affecting the quantification of fluorescence intensity for each adhesion we use a low pass filter to obtain an image of the background intensity. The original image is then divided by the background intensity image to obtain an image with a flat background intensity. A series of filters further separate the focal adhesions from noise based on intensity and size criteria. The focal adhesions are then tracked along the time-lapse sequence using a nearest-neighbor algorithm. At each time point the integrated intensity for each detected focal adhesion is calculated. An image showing the outline of the detected adhesions overlayed on the original image is generated. Optionally, focal adhesions track IDs may be overlayed on the generated image as well. The centroid coordinates, along with integrated intensity values and adhesion IDs are saved to a comma-separated text file.

Cytoplasm Segmentation Assay

This assay is used to evaluate staining of cell cytoplasm. It is typically used to quantify detection of cytoplasmic components, e.g., signal transduction molecules, structural proteins, biomarkers, etc., via chromophore-conjugated antibodies. The difference between this assay and an assay such as the cell spreading assay, is that cell cytoplasm is detected and segmented based on the location of fluorescently labeled nuclei. This provides a substantial advantage since nuclei have more regular shapes facilitating detection. Stains for cytoplasm and nuclei need to emit in non-overlapping detection channels. We recommend that cells are seeded on plates with imaging quality plastic or on a glass substrate to ensure accurate quantification of fluorescence intensity. It is also preferable to have a fluorescence lamp with even illumination. Proper testing and

adjustment of the fluorescence light source should be performed before experiments. The assay uses local thresholding as explained previously to separate objects from background in both images. The positions of the nuclei are used as internal markers to determine to each cytoplasm a pixel outside the nuclear zone should be assigned. After the cytoplasm image has

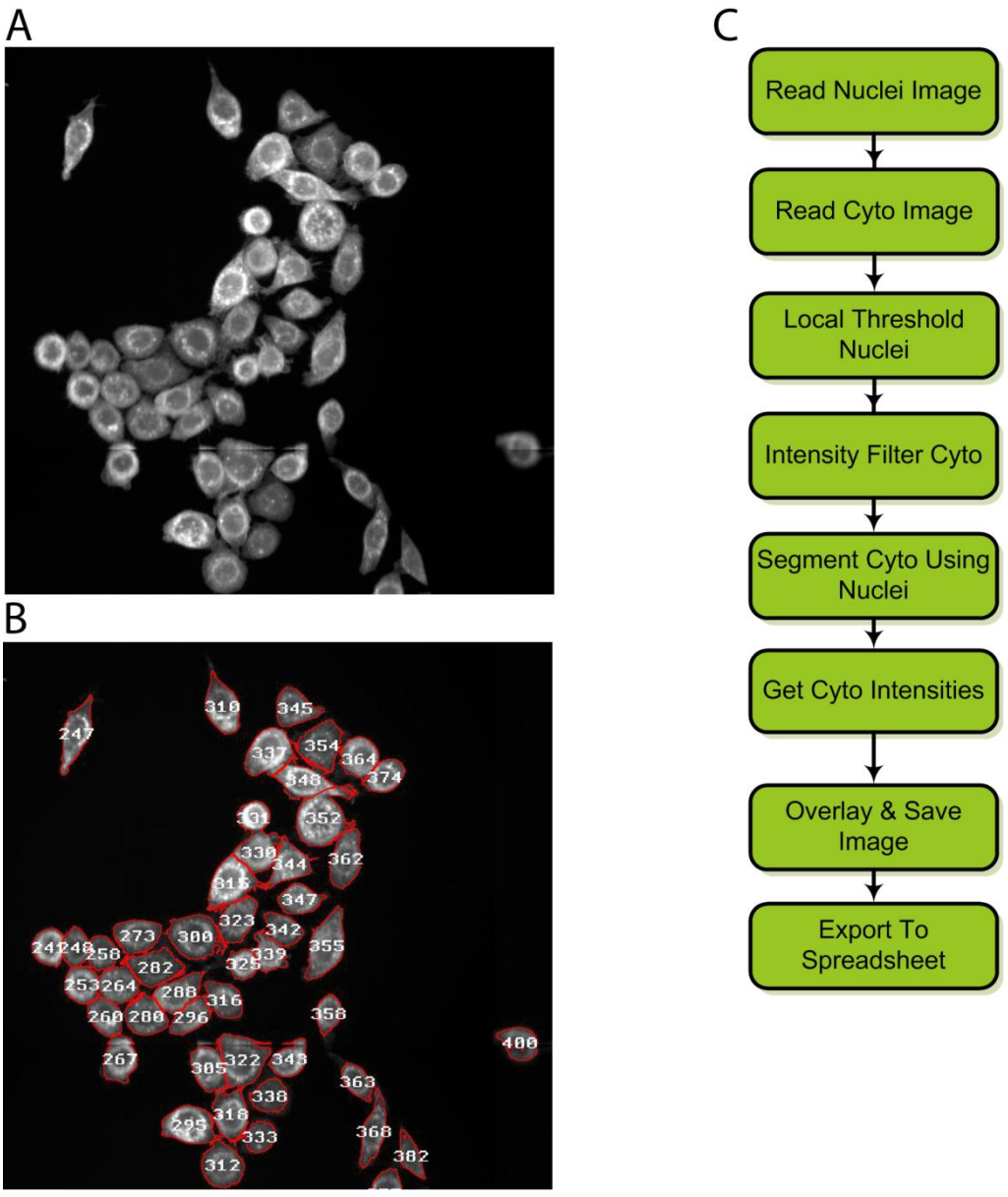

Figure 3.16. Cytoplasm segmentation assay. A) Typical input image showing labeled cytoplasm (courtesy of Peter Lee Frick). B) Detected cell outlines based on nuclear markers (not shown). C) Flowchart of major modules in this assay.

been segmented, the assay calculates integrated fluorescence intensity for the cytoplasm area alone. The integrated intensities along with unique cell IDs and centroid position are exported to a comma-separated text file. An image showing the outlines of the detected cytoplasms and the cell IDs is generated. This image may be used to evaluate the quality of the automatic segmentation.

Colony Counting Assay

This assay detects cell colonies, extracts colony shape metrics (area, solidity, etc.) and calculates the number of cells per colony. It is typically used in clonogenic assays (e.g., radiation response), anchorage-independent growth assays, stem-cell counting assays. The assay requires fluorescent labeling of cell nuclei. Every detected colony is assigned a unique ID. The number of cells in that colony along with the colony ID and colony shape metrics are saved to comma-separated text files.The images need to be acquired with high enough magnification that individual nuclei may be detected. This limits the number of colonies which may be displayed in the field of view. To alleviate this problem we recommend acquiring image montages using a microscope equipped with an automatic stage (Figure 3.17A).

The assay thresholds the montaged image using local thresholding techniques described previously. The outline of the well which may be present in montaged images is automatically removed using an area filter. To generate colony outlines the nuclei image is processed using a morphological closing. The resulting outlines are segmented into individual colonies using a cluster segmentation algorithm.The number of nuclei in each colony is calculated from the nuclear image. This information along with colony IDs and shape metrics of each colony is exported to comma-separated text files. An output image showing the detected nuclei in different

colors is generated. A second image showing the outlines of the detected colonies and the unique colony IDs is also generated for manual validation of the segmentation.

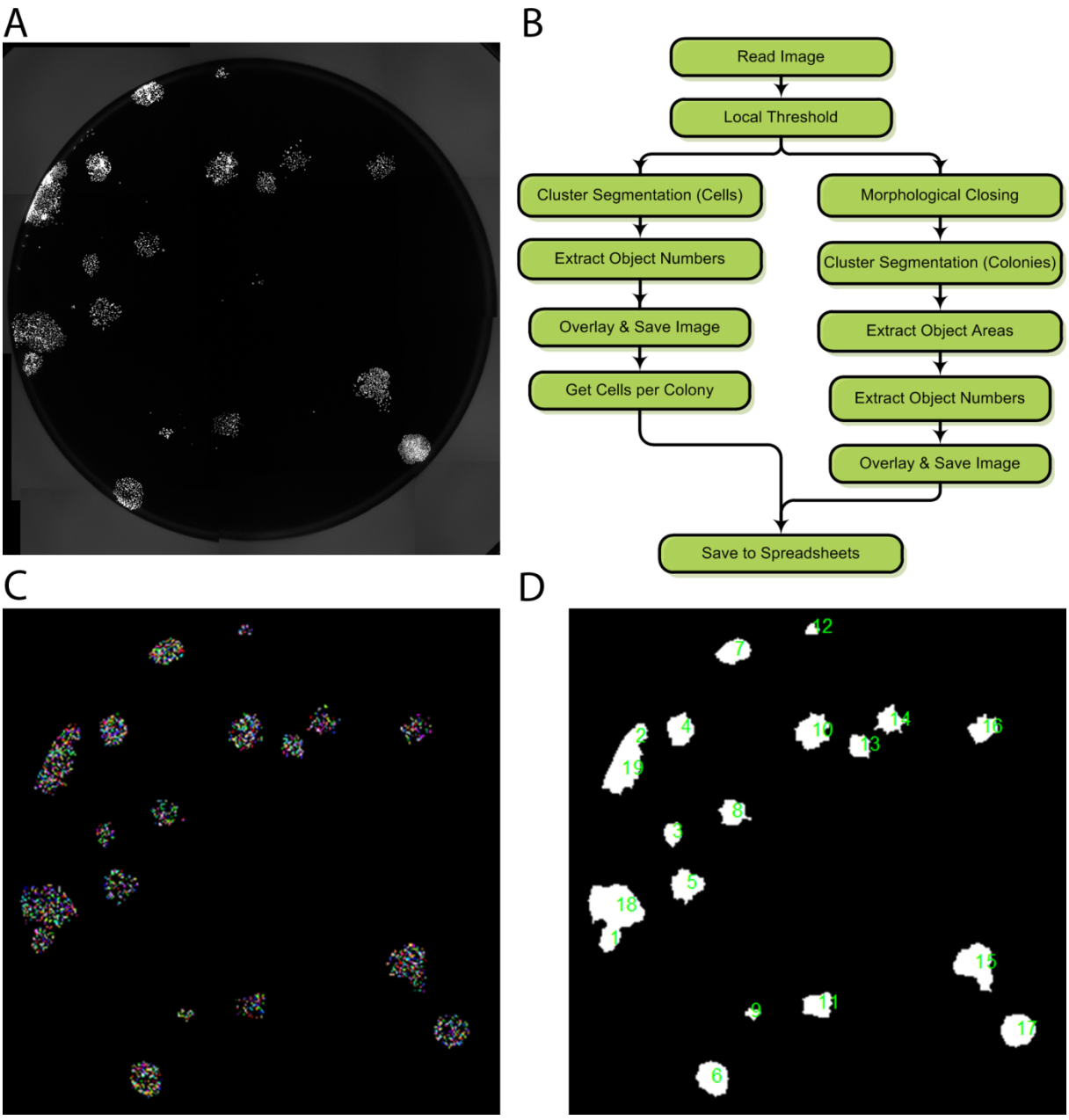

Figure 3.17. Colony counting assay. A) Typical input image.(courtesy of Yoonseok Kam) B) Flowchart showing interaction of major modules for this assay. C) Output image showing detected nuclei. D) Output image showing detected colonies and colony IDs.

Acknowledgements

The authors would like to thank Sam Hooke, Shawn Garbett, Yoonseok Kam, Phil Samson and Darren Tyson for their support and suggestions, and Allison Price for her editorial assistance.

Funding: This research is funded in part by National Institutes of Health/National Cancer Institute grants U54CA113007 and R01CA47858 to VQ, and by the Vanderbilt Institute for Integrative Biosystems Research and Education.

References

[1] M. Held, M.H.A. Schmitz, B. Fischer, T. Walter, B. Neumann, M.H. Olma, et al., CellCognition: time-resolved phenotype annotation in high-throughput live cell imaging, Nature Methods. 7 (2010) 747-54.

[2] Q. Wang, J. Niemi, C.-M. Tan, L. You, M. West, Image segmentation and dynamic lineage analysis in single-cell fluorescence microscopy, Cytometry. Part A : the Journal of the International Society for Analytical Cytology. 77 (2010) 101-10.

[3] A. Sacan, H. Ferhatosmanoglu, H. Coskun, CellTrack: an open-source software for cell tracking and motility analysis, Bioinformatics. 24 (2008) 1647-9.

[4] A. Gordon, A. Colman-Lerner, T.E. Chin, K.R. Benjamin, R.C. Yu, R. Brent, Single-cell quantification of molecules and rates using open-source microscope-based cytometry, Nature Methods. 4 (2007) 175-181.

[5] Z. Bao, J.I. Murray, T. Boyle, S.L. Ooi, M.J. Sandel, R.H. Waterston, Automated cell lineage tracing in Caenorhabditis elegans, Proceedings of the National Academy of Sciences of the United States of America. 103 (2006) 2707-12.

[6] A.E. Carpenter, T.R. Jones, M.R. Lamprecht, C. Clarke, I.H. Kang, O. Friman, et al., CellProfiler: image analysis software for identifying and quantifying cell phenotypes, Genome Biology. 7 (2006) 1-11.

[7] M.D. Abràmoff, P.J. Magalhães, S.J. Ram, Image Processing with ImageJ, Biophotonics International. 11 (2004) 36-43.

[8] H. Yamashita, M. Shang, M. Tripathi, J. Jourquin, W. Georgescu, S. Liu, et al., Epitope mapping of function-blocking monoclonal antibody CM6 suggests a “weak” integrin binding site on the laminin-332 LG2 domain, Journal of Cellular Physiology. 223 (2010) 541-8.

[9] V. Quaranta, D.R. Tyson, S.P. Garbett, B. Weidow, M.P. Harris, W. Georgescu, Trait variability of cancer cells quantified by high-content automated microscopy of single cells, Methods in Enzymology. 467 (2009) 23-57.

CHAPTER IV

IMPROVING CELL TRACKING WITH DYNAMIC PARAMETER GROUPS[*]

Abstract

High-throughput microscopy has made it possible to collect very large numbers of time-lapse images in a short amount of time. Manual tracking, the gold standard of cell tracking, is not an option for processing this deluge of data. To extract dynamic cell properties from large numbers of time-lapse images at a rate that keeps up with data generation, one must use automatic cell tracking. We have developed a novel cell tracking algorithm that uses multiple sets of dynamically determined matching criteria to track cells with very high accuracy and that compares favorably to other modern cell tracking algorithms.

Introduction

The introduction of high-throughput (HT) microscopy has led to a need for software tools designed to cope with the amount of data being generated. The software community has responded, and over the years we have seen the introduction of numerous applications designed to handle various aspects of the HT microscopy process. For the design of a HT microscopy pipeline, a number of applications are available, from open source data storing and annotation environments such as OMERO [1] and Bisque [2], to image processing applications that may be

[*] Georgescu W, Wikswo JP, Quaranta V

either open source, such as CellProfiler [3], or commercial, like Image-Pro. For those users in need of automatic cell tracking, there are currently two major options. They may opt to buy a commercial image processing application, which provides the functionality required as part of the package or as an optional add-on. In general, commercial applications provide better user interfaces at the expense of customization. Alternatively, users may opt for open source software, in which access to the underlying code allows deep customization, but with a less user-friendly interface. Regardless of the type of application used, it is important that it provide the user with as accurate a set of data as possible, since manually validating the results of data sets consisting of thousands of images is impractical. We have developed a cell tracking algorithm that uses local, temporal and nuclear morphology features to track fluorescently labeled cells. This algorithm has been extensively used for various tracking assays developed using our open source microscopy framework, CellAnimation [4]. We show that our algorithm outperforms the current state-of-the-art by comparing it to the tracking algorithms provided in two very popular image processing applications. We also compare it with a classical nearest-neighbor algorithm because this type of algorithm has been shown to outperform more complex tracking algorithms [5]. We have selected Image-Pro by MediaCybernetics and the open source application CellProfiler, developed at the Broad Institute, for our comparisons as they are both very popular microscopy applications with cell tracking capabilities.

In general, there are two phases each object tracking algorithm must perform. In the first phase objects must be separated from the background and from each other. This process is called segmentation. There are numerous types of segmentation algorithms. Two very popular algorithms are the watershed [6] and level sets [7]. To segment an image, the watershed algorithm considers it or a transformation of the image as a topographical relief. This relief is

then “flooded” from the local minima and “dams” are built to prevent the merging of “catchment basins.” These “dams” are called watershed lines, and they indicate the contours of the detected objects. In general, if the watershed algorithm is applied without any optimizations it will oversegment objects. To prevent this tendency one can pre-process the image to remove extraneous local minima or have user-provided segmentation correction [8]. Another popular approach to segmentation is the level sets method. Level sets are a type of deformable model. In a deformable model the contour of an object evolves over time. Matching the contour to the data is done using an energy minimization framework. For level sets, the contour is represented by the zero level set of a scalar function defined on the image domain. The level set approach can account for objects splitting automatically; however, ancestry assignments would still have to be determined by subsequent algorithms. An overview of level sets use in cell tracking is provided by Dufour et al. [9].

The second phase of cell tracking involves linking the cell bodies in each frame to their corresponding positions in subsequent frames, generating trajectories, and again, different types of algorithms are available. The nearest-neighbor approach links cells in the current frame to the cells nearest to their present position in the previous frame. This technique is fairly robust and simple to implement, which makes it a popular technique of many tracking applications. Some have even shown that it outperforms more complex tracking algorithms [5]. However, despite its robustness, nearest-neighbor tracking can provide very poor results in certain conditions. If the cells being observed travel long distances between frames, the core assumption of nearest-neighbor tracking, namely that cells are closest to their previous recorded position, will fail. To deal with this difficulty, additional constraints have been added to the nearest-neighbor approach or completely different techniques have been implemented. Some of these methods use shape in

addition to position information to determine whether a cell has moved, divided or died in the next frame [10]. Jaqaman et al. [11] have taken a novel approach that the linear assignment problem (LAP) framework to deal with the problems of particle or cell merging or splitting or with temporary object disappearances. In a LAP, a number of agents need to be assigned to an equal number of tasks in the best possible way [12]. Multiple algorithms have been designed to solve LAPs [13,14]. The tracking solution as implemented by Jaqaman et al. [11] uses two separate steps to establish an optimized set of tracks. Both steps are implemented as LAPs. In the first step, the objects are linked in a one-to-one fashion subject to the restriction that an object can link to one and only one object in the next frame. In the second step, the initial set of tracks is re-linked to capture temporal disappearances, merging and splitting objects. The algorithm has been shown to perform well even in high density conditions, and it has been adapted for use in CellProfiler. Image-Pro uses a proprietary cell tracking algorithm that supports different motion types: chaotic, directional and straight. We used the chaotic motion model for our comparison since this is the predominant motion type in our test movie.

Results and Discussion

Object Segmentation

For the first phase of cell tracking, namely cell segmentation, we employ two different strategies. For most image types, we use a modified version of the watershed algorithm. However, we have found that for certain situations, such as cells similar in shape that have been imaged using brightfield (Fig. 4.1A), a hierarchical clustering algorithm [15] will outperform the watershed. To prevent the oversegmentation characteristic of the watershed, one must either

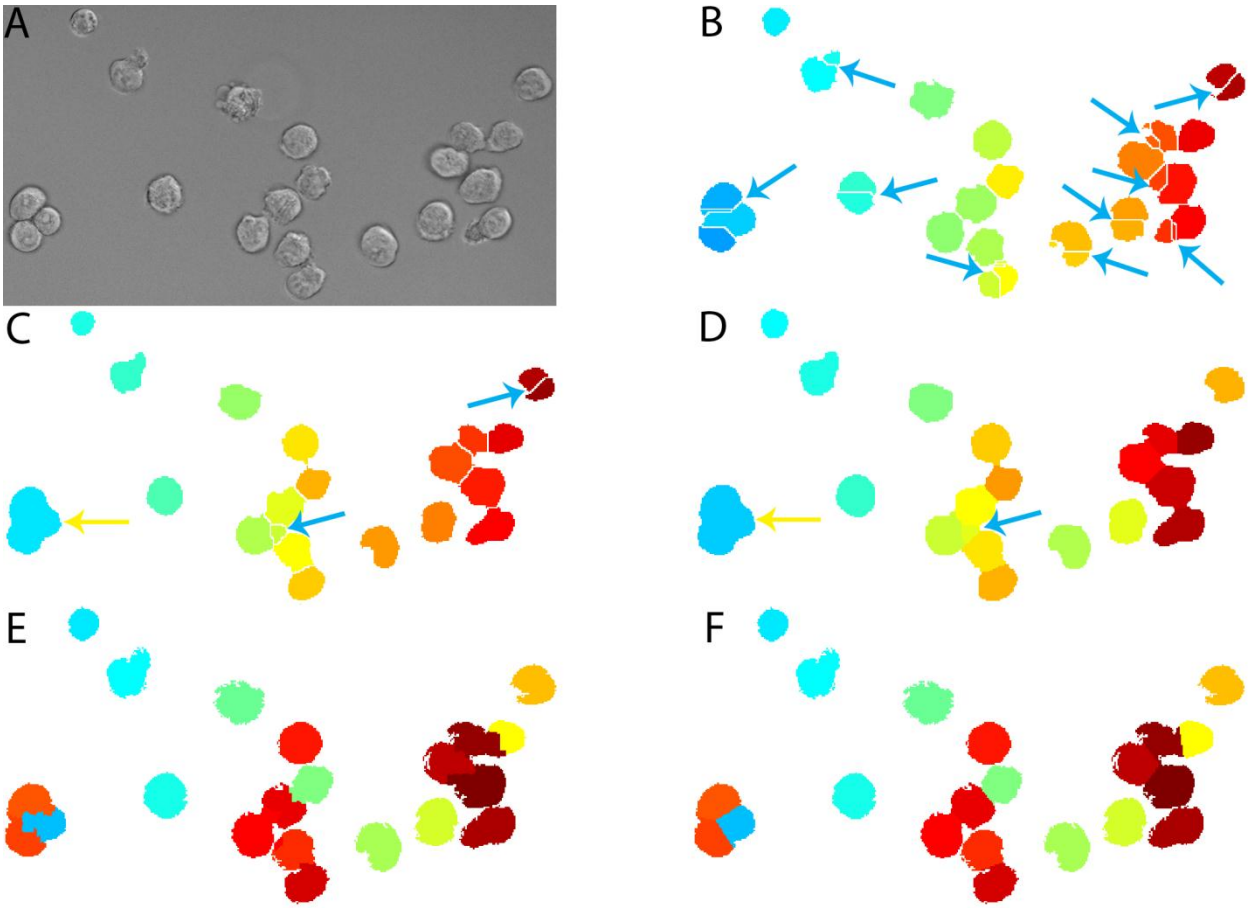

Figure 4.1. Effects of pre-processing and post-processing steps on the watershed segmentation. A) Original image. B) Watershed. C) Median-filtered watershed. D) Convexity restricted median-filtered watershed. E) Cluster segmentation. F) Cluster segmentation after pixel classification.

pre-process the image or post-process the segmentation results or both. Our approach includes both pre-processing and post-processing steps. To remove some of the extraneous local minima, we first apply a median filter to the distance transform of the binary image containing the foreground objects. While this succeeds in removing most of the oversegmentation, it does not remove it all (Fig. 4.1C). When dealing with nuclei or cells that are round in shape it, is valid to assume that clusters which have no significant concavities represent single objects, while clusters with significant concavities represent multiple objects [16,17]. We therefore add a post-processing step that prevents the watershed algorithm from splitting any convex objects. To detect only significant concavities we approximate the original contour with polygons with a small number of sides by using an external Ramer-Douglas-Peucker line simplification algorithm [18]. This works very well for nuclear stains; however, it may still fail in the case of cytoplasmic stains or brightfield images (Fig. 4.1D). To address this problem we have implemented another algorithm which uses hierarchical clustering to assign pixels in a blob to different cell bodies. Hierarchical clustering is a hard clustering method. In hard clustering (as opposed to fuzzy clustering), objects are assigned to one and only one cluster [19]. There are two types of hierarchical clustering. Agglomerative hierarchical clustering assigns each object to its own cluster. The nearest clusters are then merged based on a distance measure and the resulting clusters are merged again and again until finally all objects are assigned to a single cluster. In divisive hierarchical clustering, all objects are assigned to a cluster first and the initial cluster is then split into smaller clusters according to a distance measure. In our code, we use hierarchical agglomerative clustering to assign each pixel to one and only one cell. We use a height threshold to cut the resulting cluster tree at the appropriate height. This determines the number of cell bodies each blob contains. The appropriate height threshold for a particular cell line is

proportional to the average cell size. This approach works best if the cell sizes are not too dissimilar. In that case, the resulting numbers correctly represent the number of cell bodies and their approximate positions. However, the distribution of pixels between the different cell bodies in a cell cluster is not accurate (Fig. 4.1E). This problem can be solved by reassigning the pixels using a nearest-neighbor algorithm based on the locations of the cell body centroids (Fig. 4.1F).

Trajectory Linking

The second phase of cell tracking consists of linking the positions of cells at different times into complete trajectories. In cell tracking (as opposed to general purpose object tracking), one must account for cell splitting and apoptosis. To address these issues we have divided our trajectory algorithm into two separate parts.

First, cell trajectories are generated based on local properties of the cells, such as morphology and motility parameters. As others have done [10], we employ distance, direction and seven shape parameters (area, eccentricity, major axis length, minor axis length, orientation, perimeter and solidity) to compare cells between frames. Our algorithm links a cell in the current frame to its position in a previous frame based on which cell in the previous frame it matches best in *significant* parameters. Unlike other algorithms that rely on a fixed set of parameters to establish cell trajectories, we use dynamic groups of parameters to match specific cell or nuclear phenotypes. These dynamic groups are automatically determined by the software based on cells in the movie that can be linked with a high level of confidence. Cells whose matching parameters are ordered in the same fashion are assigned to the same group. So for example all the cells in which distance is the best matching parameters, area is the second best, and so on, will be part of the same group. We can make a match with high confidence when the cell is either the sole

uncontested match for a cell in the previous frame or when it matches a previous cell best in all parameters. We then use the information extracted from these links to rank matching parameters based on their predictive power.

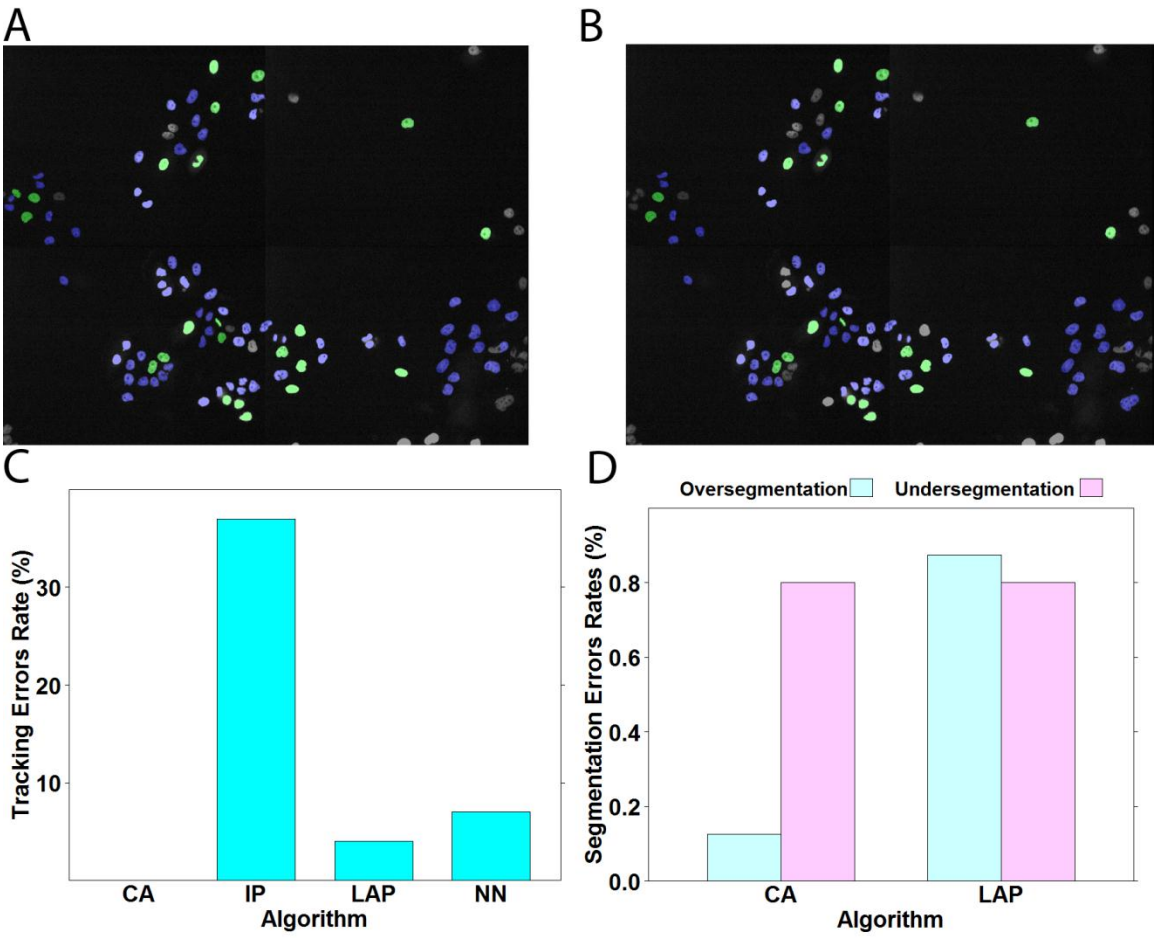

Figure 4.2. Tracking algorithms comparison. Compared to LAP (B), the CA algorithm (A) tracks more cells from the initial frame to the last frame (green nuclei) and correctly detects more mitotic events. All cells were tracked using fluorescently labeled nuclei. Daughter cells that were the result of a mitotic event correctly identified by the software are marked in blue. C) Tracking error rates for CellAnimation, Image-Pro, LAP and nearest-neighbor algorithms (N=69 cells over 72 frames, 3448 decision points). CellAnimation loses the fewest tracks. D) Correction of base undersegmentation/oversegmentation errors. CA corrects more oversegmentation errors.

A parameter is considered to have high predictive power if it varies little between the two instances of the cell, yet it varies significantly between different cells. To quantify how much a

parameter varies in the population we compute its coefficient of variation. Parameters with high predictive power are assigned more weight when establishing a trajectory.

We find that this approach offers significant advantages compared to using a fixed set of parameters. A set of parameters that is unchanging fails to capture the heterogeneity in phenotypes that may be present in the cell population or even the changes in the phenotype of a single cell as it progresses through the cell cycle. In numerous cases, we have observed cells that were matched incorrectly due to parameters not relevant to the cell's phenotype (such as direction for an immobile cell) skewing the final decision. We determine what group of parameters to use in matching a specific cell by comparing its morphology and motility values with the average values of parameters for cells in that group. The group that is the closest match to the cell is then used to determine which parameters are significant when linking it to cells in the previous frame.

The temporal progression of this preliminary set of tracks is examined for possible merging and splitting opportunities. Tracks that are parallel to each other for their entire duration and never further apart then a maximum distance *D* are merged. Tracks that split and merge only to split again are split at the first split point. Finally, we use nuclear phenotype and cell lifespan to determine if a mitotic event has occurred. When a mitotic event is detected the parent track is ended and new daughter tracks are created. It is important to be able to detect mitotic events as most cells will split at least once during a typical time-lapse experiment.

To compare our tracking results to those of other tracking algorithms, we have used time-lapse images of cells labeled with a fluorescent nuclear stain. Cells that were present in the first frame were manually tracked through the time-lapse until the cell either split, migrated out of the

imaging area, died or the end of the time-lapse was reached. We then used our Cell Animation software, Image-Pro, CellProfiler and a nearest-neighbor algorithm to automatically track the same movie that had been previously tracked manually. We assigned a tracking error every time the object ID for a cell changed. We also assigned a tracking error if a cell was not assigned any ID for more than five frames. For cells that split, we recorded whether the software correctly detected the mitotic event (if such functionality was available) and assigned an error in split detection if a mitotic event occurred that was not detected. A split error was also assigned for false positives when no mitotic event occurred but the software detected an event.

To eliminate the effects of segmentation in our tracking evaluation, we used the same set of segmentation data for CellAnimation, CellProfiler and the nearest-neighbor algorithm. Since this approach was not possible with Image-Pro, we excluded those tracks in Image-Pro affected by a segmentation mistake.

When compared to manually tracked data, our custom CellAnimation algorithm had the lowest error rate out of the four algorithms used in the test (Fig. 4.2C). It also was able to identify correctly more mitotic events with half the error rate of LAP. Because both our algorithm and LAP support track merging and splitting, we looked at how well these two algorithms corrected errors in object segmentation introduced by the first tracking phase. We compared the number of oversegmented and undersegmented objects present after the initial tracking phase to the numbers present at the end of the secondary phase. We observed that both algorithms corrected some errors and introduced no new ones when compared with manually tracked data. CellAnimation had much better oversegmentation correction and only slightly worse undersegmentation correction when compared to LAP (Fig. 4.2D).

Materials and Methods

Cell Culture

MCF10A cells, a human cell line derived from epithelial breast cells, were cultured in GIBCO DMEM/F-12 media (Invitrogen, Carlsbad, CA) supplemented with 5% horse serum (Invitrogen), 10 µg/ml insulin (Invitrogen), 0.5 µg/ml hydrocortisone (Sigma, St. Louis, MO), 20 ng/ml epidermal growth factor (Invitrogen) and 0.1 µg/ml cholera toxin (Calbiochem, Gibbstown, NJ). Cells were kept in culture in an incubator at 5% CO_2 and 37 °C.

Image Acquisition

For fluorescence imaging, cells were labeled with histone H2B conjugated to monomeric red fluorescent protein (H2BmRFP; Addgene Plasmid 18982). Labeled cells were plated in 96 wells BD Imaging plates (BD, Franklin Lakes, NJ) and allowed to adhere overnight in an incubator. The next day, cells were placed in a BD Pathway 855 high-throughput confocal imager (BD, Franklin Lakes, NJ) equipped with a temperature and CO_2 controlled hood. Two-by-two fluorescence image montages were acquired every 15 minutes for 6 hours.

Software and Hardware

For image processing we used a Dell XPS 17 laptop computer (Dell, Round Rock, TX) equipped with an Intel Core i7-2630QM CPU at 2 GHz and 6 GB of RAM. The CellAnimation framework was written in MATLAB (Mathworks, Natick, MA) and requires the Image Processing Toolbox. We have also used CellProfiler version 11710 (Broad Institute, Cambridge, MA) and Image-Pro Plus 3D version 7.0.1.658 for Windows XP/Vista.

Conclusions

We have shown a novel approach to cell tracking that uses information extracted from objects which can be assigned to a particular track with high confidence to establish groups of matching parameters. These dynamic groups are used to determine cell trajectories in low confidence situations. This approach allows us to use temporal information not only to establish the most likely trajectories, but also to improve the predictive power of local shape parameters. Because the algorithm learns what parameters are reliable from the movie itself, it is self-training, combining the accuracy of algorithms that require manual training with the ease of use of a fully automatic algorithm. Finally, the presence of multiple ranking groups allows the software to naturally accommodate heterogeneous or mixed cell populations.

Acknowledgements

The authors would like to thank Sam Hooke, Shawn Garbett, Phil Samson and Darren Tyson for their support and suggestions, and Allison Price for her editorial assistance.

Funding: This research is funded in part by National Institutes of Health/National Cancer Institute grants U54CA113007 and R01CA47858 to VQ, and by the Vanderbilt Institute for Integrative Biosystems Research and Education.

References

[1] C. Allan, J.-M. Burel, J. Moore, C. Blackburn, M. Linkert, S. Loynton, et al., OMERO: flexible, model-driven data management for experimental biology, Nature Methods. 9 (2012) 245-53.

[2] K. Kvilekval, D. Fedorov, B. Obara, A. Singh, B.S. Manjunath, Bisque: a platform for bioimage analysis and management, Bioinformatics. 26 (2010) 544-52.

[3] A.E. Carpenter, T.R. Jones, M.R. Lamprecht, C. Clarke, I.H. Kang, O. Friman, et al., CellProfiler: image analysis software for identifying and quantifying cell phenotypes, Genome Biology. 7 (2006) 1-11.

[4] W. Georgescu, J.P. Wikswo, V. Quaranta, CellAnimation: an open source MATLAB framework for microscopy assays, Bioinformatics. 28 (2012) 138-9.

[5] S. Jaeger, Q. Song, S.-S. Chen, DYNAMIK : a software environment for cell DYNAmics , Motility , and Information tracKing , with an application to Ras pathways, Bioinformatics. 25 (2009) 2383-2388.

[6] S. Beucher, C. Lantuejoul, Use of Watersheds in Contour Detection, in: International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation, Rennes, France, 1979.

[7] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations, Journal of Computational Physics. 79 (1988) 12-49.

[8] D.N. Mashburn, H.E. Lynch, X. Ma, M.S. Hutson, Enabling user-guided segmentation and tracking of surface-labeled cells in time-lapse image sets of living tissues, Cytometry. Part A : the Journal of the International Society for Analytical Cytology. 81A (2012) 409-418.

[9] A. Dufour, A. Thébaud, S. Berlemont, V. Meas-Yedid, J.-C.O. Marin, On the Digital Trail of Mobile Cells, IEEE Signal Processing Magazine. 54 (2006) 54-62.

[10] O. Al-Kofahi, R.J. Radke, S.K. Goderie, Q. Shen, S. Temple, B. Roysam, Automated Cell Lineage Construction, Cell Cycle. 5 (2006) 327-335.

[11] K. Jaqaman, D. Loerke, M. Mettlen, H. Kuwata, Robust single-particle tracking in live-cell time-lapse sequences, Nature Methods. 5 (2008) 695-702.

[12] R. Burkard, Linear assignment problems and extensions, in: Handbook of Combinatorial Optimization, 1999: pp. 1-54.

[13] H. Kuhn, The Hungarian method for the assignment problem, Naval Research Logistics Quarterly. (1955).

[14] A. Volgenant, A shortest augmenting path algorithm for dense and sparse linear assignment problems, Computing. 340 (1987) 325-340.

[15] P.H. Sneath, The application of computers to taxonomy, Journal of General Microbiology. 17 (1957) 201-26.

[16] W.X. Wang, Binary Image Segmentation of Aggregates Based on Polygonal Approximation and Classification of Concavities, Pattern Recognition. 31 (1998) 1503-1524.

[17] G. Fernandez, M. Kunt, J.-P. Zryd, A New Plant Cell Image Segmentation Algorithm, Lecture Notes in Computer Science. 974 (1995) 229-234.

[18] U. Ramer, An Iterative Procedure for the Polygonal Approximation of Plane Curves, Computer Graphics and Image Processing. 1 (1972) 244-256.

[19] A.K. Jain, M.N. Murty, P.J. Flynn, Data Clustering : A Review, ACM Computing Surveys. 31 (2000).

CHAPTER V

EFFECTS OF HEPSIN OVEREXPRESSION IN THE LNCAP-34 PROSTATE CANCER CELL LINE*

Abstract

In this study we investigate the interaction between laminin-332 and hepsin in the LNCaP prostate cancer cell line. Hepsin, a type II transmembrane serine protease, is overexpressed in prostate cancer. Hepsin overexpression has been linked with tumor progression and metastasis in a mouse model of prostate cancer. LNCaP-17 cells are low hepsin-expressing clones, and LNCaP-34 cells are hepsin-overexpressing clones. We show that LNCaP-17 cells use a different set of integrins to attach to laminin-332. We also demonstrate that LNCaP-34 cells have a more mesenchymal morphology and higher migration speeds than LNCaP-17 cells.

Introduction

Prostate cancer is the most diagnosed cancer in men and the second leading cause of cancer death in North American men [1]. Preventing prostate cancer metastasis would be of great benefit to a large number of men suffering from this disease.The expression of a number of proteins increases in prostate cancer. Hepsin, a type II transmembrane protease, is the most consistently overexpressed in a majority of prostate cancers [2]. *In vitro* studies of hepsin have been inconclusive with regard to its role in prostate cancer [3–5]. The strongest evidence of a

* Georgescu W, Tripathi M, Wikswo JP, Quaranta V

role for hepsin in prostate cancer has come from mouse models of the disease, where it has been shown that hepsin drives tumor growth and metastasis [2].

The trimeric glycoprotein laminin-332 (formerly laminin-5) is composed of three polypeptide chains: α3, β3 and γ2 [6]. It is an important component of the basement membrane [7]. A mutation in any of the three chains of laminin-332 leads to the lethal skin blistering disorder, junctional epidermolysis bullosa [8]. Laminin-332 is overexpressed in a number of cancers, such as cervical, colon, cutaneous, esophageal, laryngeal, oral and tracheal [9]. Contrary to this pattern there is loss of laminin-332 in prostate cancer [10]. Cleavage of laminin-332 has been shown to cause an increase in cell migration [11–15]. Vito Quaranta's group has previously demonstrated that cleavage of the γ2 chain by matrix metalloprotease-2 (MMP2) or by human membrane type 1 MMP (MT1-MMP) elicits migration of a number of different cell lines [11,12]. Other MMPs have been shown to cleave the γ2 chain of laminin-332 and induce epithelial cell migration [13]. In addition to MMPs, other enzymes have been reported to cleave the γ2 chain, although the effect on cell migration was not reported for all enzymes [16–19]. The α3 chain of laminin-332 is cleaved by a variety of proteases [17,18,20]. Unlike cleavage of the γ2 chain, which tends to promote motility, cleavage of the α3 chain tends to promote adhesion [17].

The β3 chain of laminin-332 is more resistant to proteolytic processing, and only a few proteases have been shown to cleave it. Udayakumar et al. [15] showed that MT1-MMP cleaves the β3 chain and that cleavage increased migration of DU-145 prostate cancer cells. Subsequently, it was shown by Remy et al. [14] that matrilysin-1 also cleaves the β3 chain and that cleavage of the chain results in increased cell motility. Our lab has shown that hepsin will cleave the β3 chain as well and that DU145 prostate cancer cells exhibit increased motility on

hepsin cleaved laminin-332 [21]. We have also shown that LNCaP cells overexpressing hepsin exhibit increased migration compared to low hepsin-expressing cells [21].

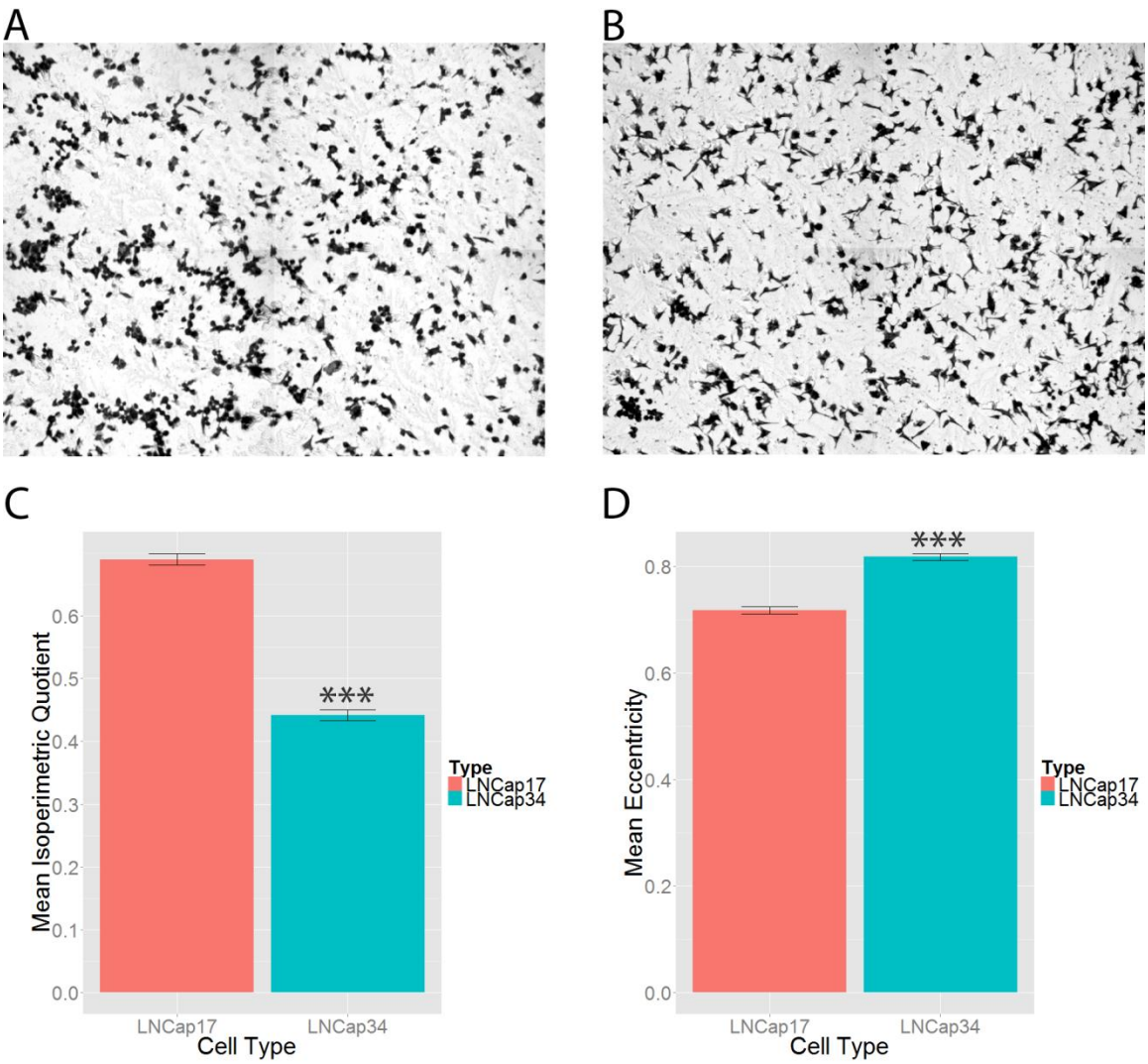

Figure 5.1. Morphological differences. LNCaP-34 and LNCaP-17 plated on laminin-332. LNCaP-34 cells (N=3, 590 total cells) (B) are more elongated compared to LNCaP-17 cells (N=3, 464 total cells) (A). These differences are quantified as significantly lower mean isoperimetric quotient ($p<0.001$, two sided Mann-Whitney test) (C) and significantly higher mean eccentricity ($p<0.001$, two sided Mann-Whitney test) for the LNCaP-34 cell line (D).

Results and Discussion

LNCaP-34 Cells are Morphologically Different from LNCaP-17 Cells

It has been previously shown that LNCaP-34 cells are more motile *in vitro* and more invasive *in vivo* [21]. We wanted to know if this is accompanied by a change in morphology, such as elongation, which is known to accompany epithelial-to-mesenchymal (EMT) transition [22]. To test for changes in morphology, we used our software, CellAnimation [23] , to extract shape data from brightfield images of cells stained with crystal violet. There are several measures that may be used to compare the difference in elongation between two objects. For our tests we used the isoperimetric quotient and eccentricity. The isoperimetric quotient is defined as $Q = \frac{4\pi A}{P^2}$ for closed curves, where P is the perimeter of the shape and A is the area. The maximum value of Q is one and it is attained only if the shape is a circle, the most compact shape. Eccentricity is a value that indicates how much a conic section deviates from a circular shape. To calculate the eccentricity we approximated each cell shape with an ellipse that had the same second-moments. Using this ellipse we calculated the eccentricity as $= \frac{C}{M}$, where C is the distance from the center of the ellipse to the focus, and M is the semimajor axis. The eccentricity ranges from zero to one, where zero indicates the shape is a circle and one indicates a line segment. We observed that LNCaP-17 cells had a significantly lower eccentricity (Fig. 5.1D) and a significantly higher isoperimetric quotient (Fig. 5.1C). These values confirm that LNCaP-34 cells are more elongated than LNCaP-17 cells.

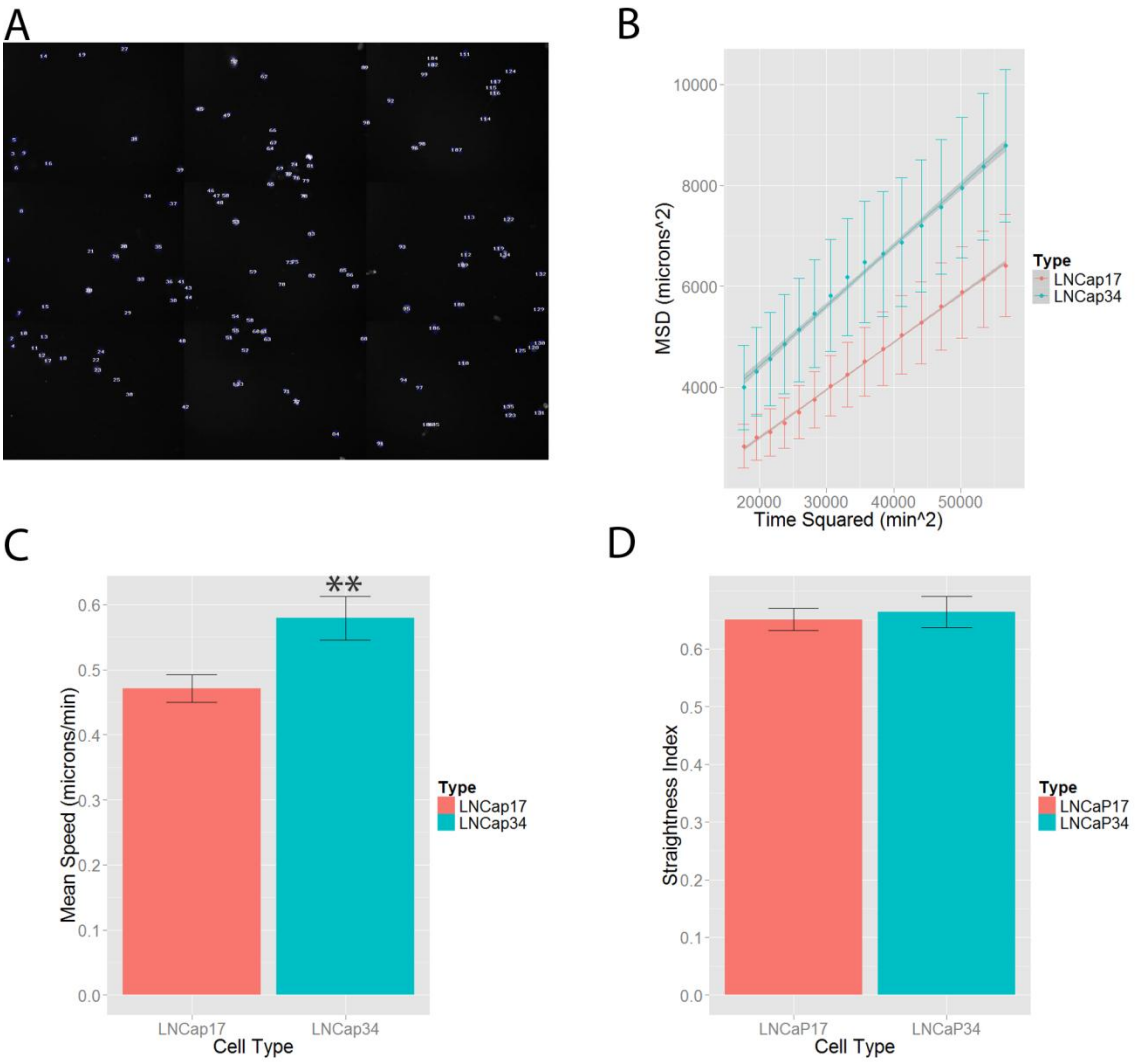

Figure 5.2. Motility differences. LNCaP-34 cells have higher MSD than LNCaP-17 cells. A) Output frame from CellAnimation showing outlines and track IDs of detected fluorescent nuclei. B) MSD of LNCaP-17 (N=3, 207 total cells) and LNCaP-34 cells (N=3, 116 total cells). The increased displacement is due to increased speed ($p<0.01$, two sided Mann-Whitney test) (C), not to reduced path tortuosity (D).

Motility Differences

Our lab has previously shown that LNCaP-34 cells are more motile on laminin-332 than LNCaP-17 cells. However, at the time we were unable to test what components of motility were altered by hepsin. Others have shown that both velocity and directional persistence increase in more invasive cells. To extract motility data from time-lapse images of fluorescently labeled cell

nuclei, we used our imaging framework, CellAnimation [23]. Once the images were processed, a set of overlayed images was available for manual validation of cell tracking data (Fig. 5.2A). We calculated speed and displacement from the cell centroid positions extracted from the time-lapse movies. We observed that LNCaP-34 cells had a significantly higher average speed than LNCaP-17 cells ($p<0.01$). Average speed for LNCaP-34 cells was 0.58 μm/min, while the slower LNCaP-17 cells moved at an average speed of 0.47 μm/min (Fig. 5.2C). The increased migration speed was accompanied by a larger mean square displacement (MSD) for the LNCaP-34 cells (Fig. 5.2B). We fitted the MSD data using a persistent random walk (PRW) model according to the following equation: $\left\langle D^2 \right\rangle = 2S^2P^2\left(\frac{t}{P} - 1 + e^{-\frac{t}{P}}\right)$, where $\left\langle D^2 \right\rangle$ is the MSD, S is speed, P is persistence time and t is time. After fitting the model, we obtained a best fit P=84 min and S=0.49 μm/min for the LNCaP-17 cells. MSD data for the LNCaP-34 cells were best fitted with P=50 min and S=0.68 μm/min. We observed that the increase in speed of the LNCaP-34 cells was associated with a decrease in persistence time. To determine if the observed decrease in persistence time was due to a change in the average path tortuosity of the LNCaP-34 cells or due solely to the increase in speed, we calculated the straightness index for the two populations. The straightness index is calculated as the ratio $S = \frac{D}{L}$, where D is the shortest distance between the starting point and the end point of the path and L is the length of the path the cell traveled between the two points. To calculate this measure we had to split the population into a motile fraction and an immobile fraction. Cells which traveled less than two cell diameters from their starting position were assigned to the immobile fraction. For S to be a relevant measure, all cells have to be measured after they have traveled an equal distance. However, time-lapse sequences capture cells at equal time intervals. We resampled the motility data for equal length intervals by

fitting smooth splines through the positions of each cell using the *smooth.spline* function in R [24]. Paths of equal length were extracted from the resulting smooth splines. We then calculated S with $D = D_{min}$, where D_{min} is the shortest final path length traveled by any cell in the motile fraction. We observed no significant difference between the straightness index for the two populations, indicating that hepsin does not cause a change in directional persistence in 2D.

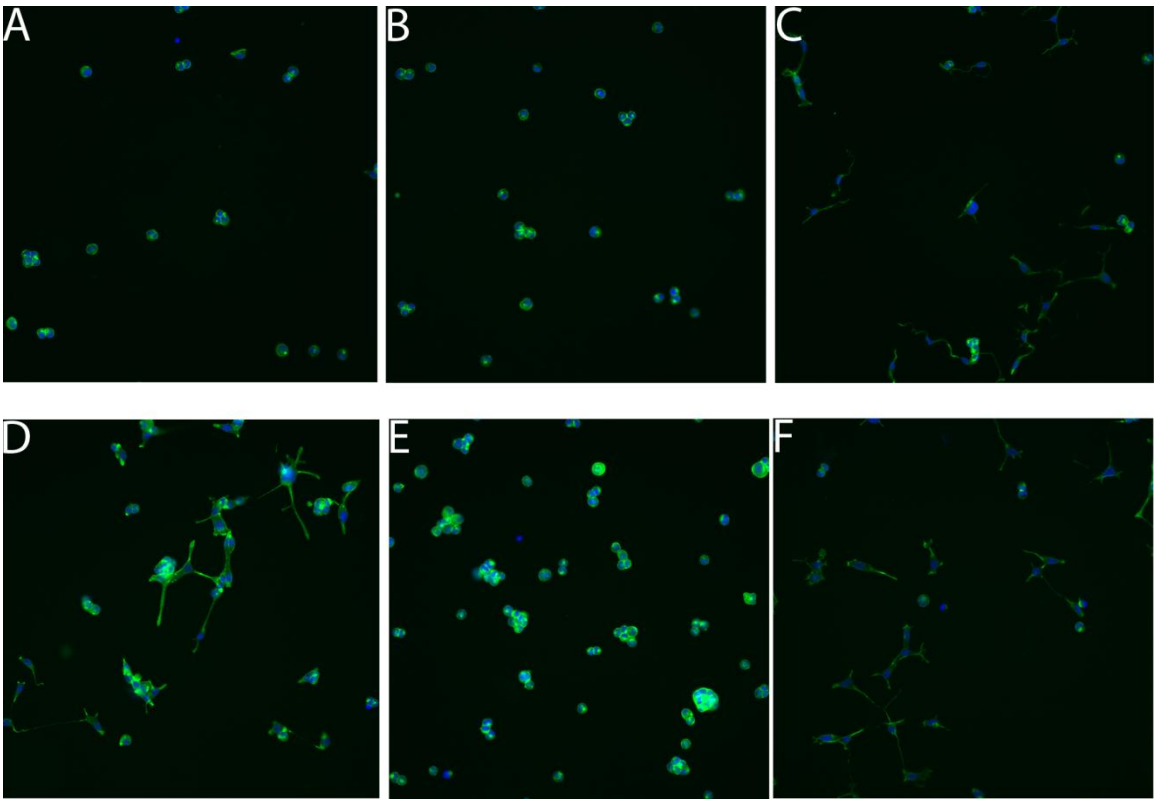

Figure 5.3. Integrin blocking response. We tested the effect of blocking integrin function using integrin α6 (A,D), β1 (B,E) and β4 (C,F). Blocking of β1 resulted in blocking of adhesion in both LNCaP-17 (E) and LnCap-34 (B) cells. α6 blocking had the same effect in LNCaP-34 cells but did not affect adhesion in LNCaP-17 cells. Blocking the β4 integrin had no significant effect on the adhesion of either cell line.

Differences in Attachment to Laminin-332

To determine whether the observed differences in motility and morphology are caused by a difference in integrin binding to the extracellular matrix, we used function blocking antibodies

to prevent cell attachment to the extracellular matrix via specific integrins. LNCaP cells may attach to laminin-332 via α6β4, α3β1, or α6β1 integrins [25]. We blocked each of these adhesions in turn by using antibodies to α6, β1and β4 (Fig. 5.3). LNCaP-17 adhesion was blocked by the β1 antibody but not by α6 or β4 antibodies, indicating adhesion through the α3β1 integrin. LNCaP-34 adhesion was blocked by both α6 and β1 antibodies but not by the β4 antibody, indicating that LNCaP-34 cells use a different integrin to attach to laminin-332, namely α6β1.

Possible Mechanisms

We believe the use of integrin α6 in the hepsin-overexpressing cells is responsible for the difference in motility between these cells and the low hepsin-expressing cells. In a previous study, Rabinovitz et al. have shown that DU-145 prostate carcinoma cells selected for high integrin α6 expression have been shown to be three times more motile *in vitro* than DU-145 cells selected for low α6 expression [26]. Intraperitoneal injection of the cells into mice showed greater invasion for the DU-145 cells expressing high levels of integrin α6. A study by Schmelz et al. using prostate carcinoma biopsies from 61 men found three different types of prostate carcinoma phenotypes [27]. Two phenotypes contained integrin α6 and integrin α3, while the last phenotype contained integrin α3 only. In differentiated tumors invading into the seminal vesicles only the first two phenotypes were found indicating that α6 expression leads to a more invasive phenotype.

Hepsin can affect the activation of the α6 integrin in multiple ways. One possibility is that the known cleavage of laminin-332 by hepsin exposes new binding sites that make the substrate more suitable for attachment with the α6β1 integrin. Another possibility we are now considering is a signaling pathway which involves the urokinase-type plasminogen activator

(uPA). It has been shown previously that uPA is a substrate for hepsin [28]. Hepsin was demonstrated to activate pro-uPA at a rate which was six times higher than matriptase. The rate of activation of pro-uPA by hepsin was similar to plasmin which is the most potent pro-uPA activator. It has also been shown that uPA cleaves integrin α6 leading to the formation of a truncated form called integrin α6p. GOH3, the function blocking antibody we have used can block α6p as well as α6.The α6p integrin is absent in normal prostate tissue but present in invasive prostate cancer tissue [29]. *In vitro* the clipped version of integrin α6 has been demonstrated to increase migration on laminin substrates [30]. We think that hepsin activation of pro-uPA leads to production of the integrin α6p which in turn leads to the observed changes in motility and shape. Activation of integrin α6 has been shown to cause increased activity of FAK and MAP kinase [31]. It is likely that integrin α6p can also activate FAK. FAK activation in turn is well known to lead to increased cell motility in both normal and cancer cells [32] and an elongated cell shape [33].

Materials and Methods

Nuclei of LNCaP cells were labeled using histone2B mRFP (Addgene Plasmid 18982). The labeled cells were cultured in Gibco RPMI 1640 medium (Invitrogen, Carlsbad, CA) supplemented with 10% fetal bovine serum (FBS), 500 µg/ml Geneticin (Invitrogen), 0.5 µg/ml puromycin (Sigma-Aldrich, St. Louis, MO), and 1% glutamine/penicillin/streptomycin at 5% CO_2 and 37 °C.

For the shape measurement experiments, 96 well tissue culture plates (TPP, Trasadingen, Switzerland) were coated with laminin-332 overnight at 4 °C. Afterwards, the wells were

blocked with 1% BSA in DMEM for one hour at 37 °C to prevent further adhesion. Cells were then stained with crystal violet using standard protocol. Briefly, cells were allowed to adhere for one hour, washed with cold PBS, fixed with methanol, stained with 0.5% crystal violet in 25% methanol for 10 minutes, washed with water after 30 minutes and dried overnight at room temperature.

For motility experiments, cells were plated on 96 well imaging plates (BD, Franklin Lakes, NJ) coated overnight with laminin-332. Cells were plated in RPMI 1640 supplemented as before and allowed to adhere overnight at 5% CO_2 and 37 °C. The cells were then imaged in a BD Pathway 855 high-throughput imager (BD, Franklin Lakes, NJ). Motility data were extracted from time-lapse sequences using our custom imaging framework CellAnimation [23]. Comma-separated motility values were imported into R [24], which was used to fit motility data to the persistence random walk model and test differences in shape and motility parameters for statistical significance. The ggplot2 module was used to produce the graphs.

In the antibody blocking experiments, we used GOH3 an antibody that has been shown to be specific to the α6 integrin [34], AIIB2 an antibody specific to the β1 integrin [35] and ASC-8 an antibody specific to the β4 integrin [36]. Cells were plated in RPMI 1640 supplemented as before, to which one of the three antibodies was added. Cells were stained with Alexa Fluor 488 phalloidin (Invitrogen) and Hoechst 33342 (Invitrogen). To stain the cells, we first fixed them using a 4% paraformaldehyde solution in PBS for 15 minutes. We rinsed the wells with PBS and then added a solution of 0.1% Triton X-100 in PBS to permeabilize the cells for five minutes. Afterwards, we rinsed the wells again with PBS, added a 1:100 dilution of stock Alexa Fluor 488 phalloidin (Invitrogen) and allowed the plate to sit for 45 minutes protected from light. After rinsing the plates again with PBS, we added a 1:10000 dilution of Hoechst 33342 stock solution

for 15 minutes. To image the wells, we used a Roche Cellavista analyzer (Roche, Basel, Switzerland).

To test if the observed differences in motility and morphology parameters were significant we used a non-parametric Mann-Whitney U test. We used a non-parametric test to account for the fact that the data did not have a normal distribution.

Conclusions

We have shown that hepsin-overexpressing LNCaP-34 cells use a different set of integrins to attach to laminin-332 than low hepsin-expressing LNCaP-17 cells. We have also shown that LNCaP-34 cells attaching via the α6β1or α6pβ1 integrin have a more elongated morphology and are more motile, which is consistent with a hypothetical activation by α6 or α6p of the FAK signaling pathway. This increased motility is due solely to an increase in migration speed. The path tortuosity is not significantly different between the two cell populations. This indicates there is no change in the intrinsic directional persistence between the two cell populations. The signaling pathway we are proposing may help explain the increase in α6 integrin that is observed in prostate cancer and it offers a plausible explanation of how hepsin over-expression can lead to increased cancer motility *in vitro*.

Acknowledgements

The authors would like to thank Allison Price for her editorial assistance.

Funding: This research is funded in part by National Institutes of Health/National Cancer Institute grants R01CA47858 to VQ, and by the Vanderbilt Institute for Integrative Biosystems Research and Education.

References

[1] Cancer Facts & Figures, Atlanta, 2012.

[2] O. Klezovitch, J. Chevillet, J. Mirosevich, R.L. Roberts, R.J. Matusik, V. Vasioukhin, Hepsin promotes prostate cancer progression and metastasis, Cancer Cell. 6 (2004) 185-195.

[3] J.-A. Xuan, D. Schneider, P. Toy, R. Lin, A. Newton, Y. Zhu, et al., Antibodies Neutralizing Hepsin Protease Activity Do Not Impact Cell Growth but Inhibit Invasion of Prostate and Ovarian Tumor Cells in Culture, Cancer Research. 66 (2006) 3611-3619.

[4] V. Srikantan, M. Valladares, J.S. Rhim, HEPSIN Inhibits Cell Growth / Invasion in Prostate Cancer Cells, Cancer Research. 62 (2002) 6812-6816.

[5] A. Torres-Rosado, K. O'Shea, A. Tsuji, S.-H. Chou, K. Kurachi, Hepsin, a putative cell-surface serine protease, is required for mammalian cell growth, Pnas. 90 (1993) 7181-7185.

[6] P. Rousselle, G.P. Lunstrum, D.R. Keene, R.E. Burgeson, Kalinin : An Epithelium-Specific Basement Membrane Adhesion Molecule That Is a Component of Anchoring Filaments, Journal Of Cell Biology. 114 (1991) 567-576.

[7] M.Y.C. Ryan, A.M. Christiano, E. Engvall, U.M. Wewer, J.H. Miner, J.R. Sanes, et al., The functions of laminins: lessons from in vivo studies., Matrix Biology. 15 (1996) 369-382.

[8] S. Kivirikko, J.A. Mcgrath, C. Baudoln, D. Aberdam, S. Ciatti, M.G.S. Dunnill, et al., A homozygous nonsense mutation in the α3 chain gene of laminin 5 (LAMA3) in lethal (Herlitz) junctional epidermolysis bullosa, Human Molecular Genetics. 4 (1995) 959-962.

[9] M.P. Marinkovich, Tumour microenvironment: laminin 332 in squamous-cell carcinoma, Nature Reviews. Cancer. 7 (2007) 370-80.

[10] T.L. Davis, A.E. Cress, B.L. Dalkin, R.B. Nagle, Unique Expression Pattern of the α6ß4 Integrin and Laminin-5 in Human Prostate Carcinoma, The Prostate. 46 (2001) 240-248.

[11] G. Giannelli, J. Falk-Marilier, O. Schiraldi, W.G. Stetler-Stevenson, V. Quaranta, Induction of Cell Migration by Matrix Metalloprotease-2 Cleavage of Laminin-5, Science. 277 (1997) 225-228.

[12] N. Koshikawa, G. Giannelli, V. Cirulli, K. Miyazaki, V. Quaranta, Role of cell surface metalloprotease MT1-MMP in epithelial cell migration over laminin-5, The Journal of Cell Biology. 148 (2000) 615-24.

[13] E. Pirilä, A. Sharabi, T. Salo, V. Quaranta, H. Tu, R. Heljasvaara, et al., Matrix metalloproteinases process the laminin-5 2-chain and regulate epithelial cell migration, Biochemical and Biophysical Research Communications. 303 (2003) 1012-1017.

[14] L. Remy, C. Trespeuch, S. Bachy, J.-Y. Scoazec, P. Rousselle, Matrilysin 1 influences colon carcinoma cell migration by cleavage of the laminin-5 beta3 chain, Cancer Research. 66 (2006) 11228-37.

[15] T.S. Udayakumar, M.L. Chen, E.L. Bair, D.C. von Bredow, A.E. Cress, R.B. Nagle, et al., Membrane Type-1-Matrix Metalloproteinase Expressed by Prostate Carcinoma Cells Cleaves Human Laminin-5 β 3 Chain and Induces Cell Migration Membrane Type-1-Matrix Metalloproteinase Expressed by Prostate Carcinoma, Cancer Research. 63 (2003) 2292-2299.

[16] P. Mydel, J.M. Shipley, T.L. Adair-Kirk, D.G. Kelley, T.J. Broekelmann, R.P. Mecham, et al., Neutrophil elastase cleaves laminin-332 (laminin-5) generating peptides that are chemotactic for neutrophils, The Journal of Biological Chemistry. 283 (2008) 9513-22.

[17] S. Amano, I.C. Scott, K. Takahara, M. Koch, M.-F. Champliaud, D.R. Gerecke, et al., Bone Morphogenetic Protein 1 Is an Extracellular Processing Enzyme of the Laminin 5 γ2 Chain, Journal of Biological Chemistry. 275 (2000) 22728-22735.

[18] D.P. Veitch, P. Nokelainen, K. a McGowan, T.-T. Nguyen, N.E. Nguyen, R. Stephenson, et al., Mammalian tolloid metalloproteinase, and not matrix metalloprotease 2 or membrane type 1 metalloprotease, processes laminin-5 in keratinocytes and skin, The Journal of Biological Chemistry. 278 (2003) 15661-8.

[19] B. Wang, J. Sun, S. Kitamoto, M. Yang, A. Grubb, H. a Chapman, et al., Cathepsin S controls angiogenesis and tumor growth via matrix-derived angiogenic factors, The Journal of Biological Chemistry. 281 (2006) 6020-9.

[20] L.E. Goldfinger, M.S. Stack, J.C.R. Jones, Processing of laminin-5 and its functional consequences: role of plasmin and tissue-type plasminogen activator, The Journal of Cell Biology. 141 (1998) 255-265.

[21] M. Tripathi, S. Nandana, H. Yamashita, R. Ganesan, D. Kirchhofer, V. Quaranta, Laminin-332 Is a Substrate for Hepsin , a Protease Associated with Prostate Cancer Progression, Journal of Biological Chemistry. 283 (2008) 30576-30584.

[22] P. Friedl, K. Wolf, Tumour-cell invasion and migration: diversity and escape mechanisms, Nature Reviews. Cancer. 3 (2003) 362-74.

[23] W. Georgescu, J.P. Wikswo, V. Quaranta, CellAnimation: an open source MATLAB framework for microscopy assays, Bioinformatics. 28 (2012) 138-9.

[24] R.D.C. Team, R: A language and evironment for statistical computing, Vienna, 2010.

[25] R. Calaluce, D.J. Bearss, J. Barrera, Y. Zhao, H. Han, S.K. Beck, et al., Laminin-5 beta3A expression in LNCaP human prostate carcinoma cells increases cell migration and tumorigenicity, Neoplasia (New York, N.Y.). 6 (2004) 468-79.

[26] I. Rabinovitz, R.B. Nagle, a E. Cress, Integrin alpha 6 expression in human prostate carcinoma cells is associated with a migratory and invasive phenotype in vitro and in vivo., Clinical & Experimental Metastasis. 13 (1995) 481-91.

[27] M. Schmelz, A.E. Cress, K.M. Scott, F. Bürger, H. Cui, K. Sallam, et al., Different phenotypes in human prostate cancer: alpha6 or alpha3 integrin in cell-extracellular adhesion sites., Neoplasia (New York, N.Y.). 4 (2002) 243-54.

[28] D. Kirchhofer, M. Peek, M.T. Lipari, K. Billeci, B. Fan, P. Moran, Hepsin activates pro-hepatocyte growth factor and is inhibited by hepatocyte growth factor activator inhibitor-1B (HAI-1B) and HAI-2., FEBS Letters. 579 (2005) 1945-50.

[29] M.C. Demetriou, M.E. Pennington, R.B. Nagle, A.E. Cress, Extracellular alpha 6 integrin cleavage by urokinase-type plasminogen activator in human prostate cancer., Experimental Cell Research. 294 (2004) 550-8.

[30] S.C. Pawar, M.C. Demetriou, R.B. Nagle, G.T. Bowden, A.E. Cress, Integrin alpha6 cleavage: a novel modification to modulate cell migration., Experimental Cell Research. 313 (2007) 1080-9.

[31] V. Carloni, a Mazzocca, P. Pantaleo, C. Cordella, G. Laffi, P. Gentilini, The integrin, alpha6beta1, is necessary for the matrix-dependent activation of FAK and MAP kinase and the migration of human hepatocarcinoma cells., Hepatology (Baltimore, Md.). 34 (2001) 42-9.

[32] S.K. Mitra, D. a Hanson, D.D. Schlaepfer, Focal adhesion kinase: in command and control of cell motility., Nature Reviews. Molecular Cell Biology. 6 (2005) 56-68.

[33] D.J. Sieg, C.R. Hauck, D.D. Schlaepfer, Required role of focal adhesion kinase (FAK) for integrin-stimulated cell migration., Journal of Cell Science. 112 (1999) 2677-91.

[34] A. Sonnenberg, H. Janssen, F. Hogervorst, J. Calafat, J. Hilgers, A Complex of Platelet Glycoproteins IC and IIa Identified by a Rat Monoclonal Antibody, Journal of Biological Chemistry. 262 (1987) 10376-10383.

[35] D.E. Hall, L.F. Reichardt, E. Crowley, B. Holley, H. Moezzi, A. Sonnenberg, et al., The α1/β1 and α6/β1 Integrin Heterodimers Mediate Cell Attachment to Distinct Sites on Laminin, Journal Of Cell Biology. 110 (1990) 2175-2184.

[36] C. Egles, H.A. Huet, F. Dogan, S. Cho, S. Dong, A. Smith, et al., Integrin-blocking antibodies delay keratinocyte re-epithelialization in a human three-dimensional wound healing model., PloS One. 5 (2010) e10528.

CHAPTER VI

A METHOD FOR DETERMINING HAPTOTAXIS EFFICIENCY IN CANCER CELL LINES*

Abstract

In this study we describe a microfluidic method to determine haptotaxis efficiency of a cell line by measuring the minimum difference in the density of a surface-bound protein which will result in a bias in the migration of the sample cells. We illustrate how the method may be applied using the HT1080 fibrosarcoma cell line and demonstrate that this cell line does not bias its migration for fairly large differences in the surface-bound concentration of type IV collagen, indicating a weak haptotaxis response.

Introduction

Directed cancer cell migration is important in cancer progression [1]. There are different types of directed cell migration. The best known type of directed cell migration is chemotaxis, which is directed cell migration up a gradient of chemokines in solution. There have been numerous studies on the role of chemotaxis in cancer [2,3], but there is less information about that of haptotaxis (the directed migration of cells on gradients of substrate-bound chemoattractant [4]).

*Georgescu W, Quaranta V, Wikswo JP

It is known that in chemotaxis there are differences in the ability of different cell lines to detect a gradient. Professional migrating cells, such as neutrophils and macrophages, have mechanisms that can amplify even very small differences in the concentration of chemoattractants so that even very shallow gradients may be detected [5]. In contrast, cells that migrate only on occasion, such as epithelial cells and fibroblasts, lack these internal amplification mechanisms and can only detect strong gradients [6]. Quantifying how well cells can detect differences in substrate-bound gradients is important not only to determine if these differences in gradient amplification mechanisms hold true for haptotaxis, as they do for chemotaxis, but also to establish the likely relevance of haptotaxis for different cell lines.

To date, studies of haptotaxis have focused on the effect of the slope of the gradients on the drift speed of the cells. The drift speed is the speed with which the cells migrate in the direction of the gradient. These studies have so far been contradictory. Smith et al. [7] examined the effect of changing the slope of fibronectin gradients on the migration of human microvascular endothelial cells (hMEC). They reported a linear increase in drift speed with increasing slope. However, when Liu et al. [8] studied migration of bovine aortic endothelial cells on alkanethiol gradients functionalized with fibronectin, they reported no significant difference in speed between cells' migration on a steep slope gradient versus a shallow one. The only significant difference in speed they observed was for cells migrating on gradients when compared with a uniform density control. A further confounding factor is that certain protein densities are better for migrating cells than others, so that in theory a cell may migrate faster on an optimal density of protein than on a gradient of the same protein.

For these reasons we decided not to use speed as a measure of cell ability to detect a haptotaxis gradient. Instead, our method uses a microfluidic device to create small and large

differences in protein density. We then calculate the ratio of cells present in the zones of high protein density to cells present in the low density zones to determine haptotaxis ability.

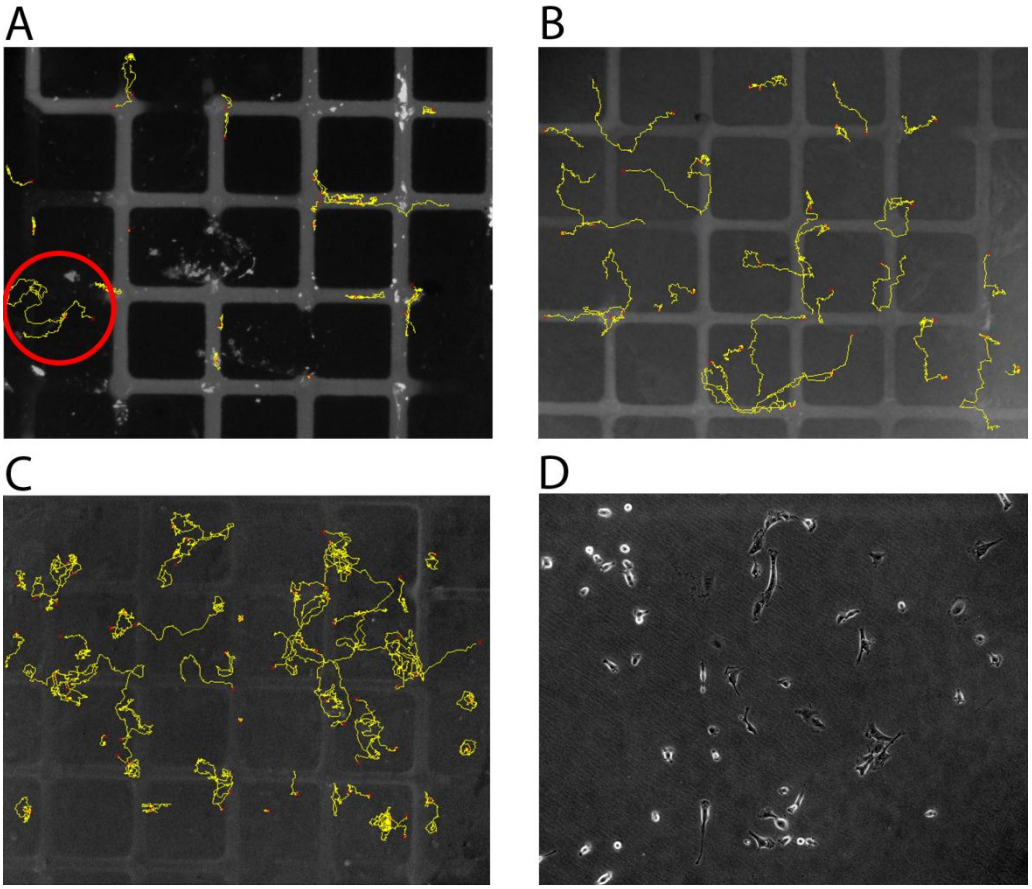

Figure 6.1. A) Tracks of HT1080 cells migrating on a type IV collagen pattern where a large difference in surface-bound type IV collagen density exists between the high and low protein zones. Note that cells on the left (image A – red circle), where there has been insufficient adsorption, are using random walk migration, while cells on the right use biased migration alon the grid pattern. B) As the difference in the density between the two zones decreases, the preference of cells for the high density zones decreases as well. C) When the difference between high and low concentration dips below the threshold of detection, all cells revert to random walk migration. D) Brightfield image of HT1080 cells.

Results and Discussion

To determine if cells show a preference for zones where the bound surface density of protein is high, we used a microfluidic device to create zones of high and low protein density. In

the high protein zones we deposited protein for over 90 minutes at room temperature which based on previous work should result in a surface density concentration of $7.69 \cdot 10^{10}$ molecules cm^{-2} [9].We were able to vary the amount deposited in the zones where protein density was low so that the difference between the two zones spanned a wide range (Figs. 6.1A and 6.1C). We observed that cells which were plated on dishes where the difference in surface-bound protein density was high preferred to migrate in the zones of high density. This is not due to an inability of cells to migrate in zones of low protein density, as cells which were migrating in low density zones away from the high density zones and therefore unable to detect them migrated with path lengths comparable to cells in high density areas (Fig. 6.1A). As the difference between the zones of high and low density was decreased, we observed that cell preference for the zones of high density diminished as well (Figs. 6.1B and 6.1C).

To quantify the cell preference for the zones of high density as a function of protein density, we compared the spatial distribution of protein densities in the image to the spatial distribution of cell centroids. If the cells have no preference for a particular protein density, we expect the distribution of cell centroids to match the distribution of protein densities with most of the centroids overlapping the most distributed protein density. If, however, the cells prefer high protein densities, we expect the shape of the distributions to diverge, with the cell centroid distribution peaking over high protein densities. We observed that indeed this divergence in distributions occurs (Fig. 6.2A) and that it decreases as the difference in protein densities decreases (Fig. 6.2B), with a match in the shape of the two distributions occurring when the average difference in densities decreases to 9% (Fig. 6.2C).

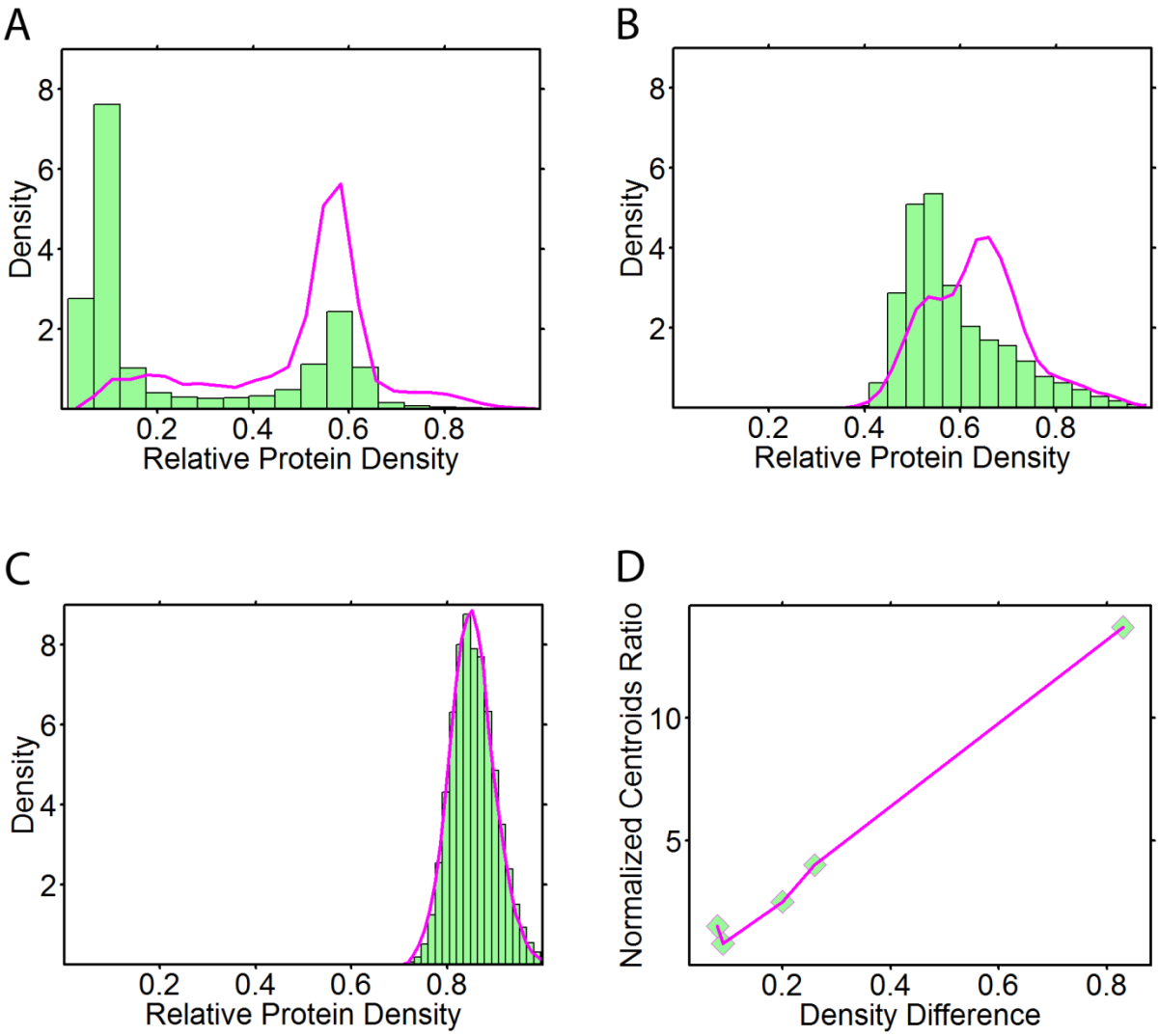

Figure 6.2. Quantifying cell preference for zones of high protein density. A) At high differences in surface-bound type IV collagen density, the cell centroid distribution (magenta line) is shifted to the right of the protein density distribution (green histogram), indicating cells' preference for areas of high protein density (N=24 cells over 160 frames, 3594 centroid positions). B,C) As the difference in densities decreases the shift decreases (B)(N=22 cells over 160 frames, 3055 centroid positions) until the two distributions overlap (C)(N=64 cells over 160 frames, 9589 centroid positions). D) Plot of normalized centroids ratio θ versus percent density difference indicates cells do not show a high density preference for differences in protein density smaller than 20%.

To further quantify cell preference for zones of high density, we calculated $\theta = \frac{C_H A_L}{C_L A_H}$, where C_H, C_L are the number of centroids in the high density zone and, respectively, the low density zone, while A_H, A_L are the areas of the two zones. With θ calculated in this fashion a

value of one indicates no preference for either area, while a value significantly above one indicates a preference for high density areas. We observed that cells plated on zones with very large differences in protein density showed a strong preference for high density zones with a maximum θ value of thirteen, indicating that thirteen times more cell centroids were present in the zones of high density than expected. The value of θ decreased as the difference in densities decreased, and it hovered around one for density differences below 20% (Fig. 6.2D).

These results show that HT1080 cells stop exhibiting a preference for zones of high protein density at relatively high differences in surface-bound protein concentration. In addition, the linear relationship between the size of the difference and the proportion of cells present in the high density zones seems to indicate a lack of internal gradient amplification by the cells. We propose that HT1080 cells are not able to detect differences in type IV collagen in an efficient manner and haptotaxis may not be a relevant mechanism of directed migration in these cells on this protein.

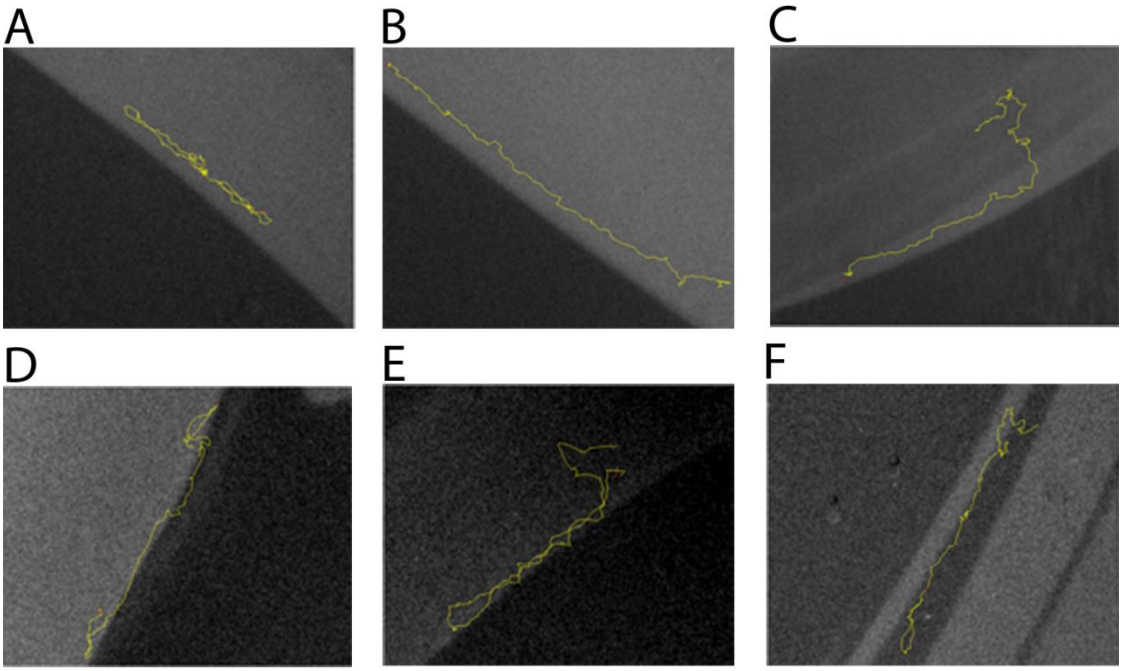

Figure 6.3. Examples of a possible novel migration mode "density guidance" on surface-bound type IV collagen patterns generated using Faraday waves. Some cells switch from random motility to directed migration to travel along the contour line defined by the change in density zones.This behavior does not occur in many cells, possibly due to population heterogeneity, possibly because this behavior is the result of a not-yet appreciated property of either the cells or the substrate.

Indeed, we have on occasion observed cells migrating along the boundary of the two surface density zones (Fig. 6.3). This behavior occurred only in a small number of cells. It is possible this behavior is only an experimental artifact; however, we have seen this behavior occurring in multiple movies with both flow printing and Faraday wave deposition, so it merits further study. The fact that it only occurs in a small number of cells may be due to population heterogeneity.This mechanism, if it turns out to be a true phenomenon and not just an experimental artifac, would be able to provide cells with navigational cues for much longer distances than a haptotaxis gradient, which in order to be detected by these cells would have to

have a very steep slope, which means surface saturation by the protein would be reached in a very short distance.

We call this mode of migration “density guidance,” similar to contact guidance, where cells migrate along topographical cues in the substrate. However, we do not think that the cells here are detecting any changes in topography for several reasons. First is the manner in which we create the density zone patterns. A long time deposition step is followed by a very short second deposition step where the entire dish is flooded with the same protein, which is then followed by a very long blocking step where the dish is flooded again with a different protein. We argue that any changes in topography resulting from the first patterned step would be attenuated by both the second flooding step and, more importantly, the last flood deposition, which blocks remaining sites for protein adsorption. This last blocking step is for a long time interval and a significant amount of protein is deposited. Second, we form our patterns using type IV collagen, which forms sheets that are poorly suited for contact guidance, unlike the fibrils formed by collagen I. Finally, we observed the same behavior on density differences produced using Faraday wave patterning, which would eliminate guidance cues being produced by the alignment of features in the microfluidic device used to create the pattern. However, the only way to prove that indeed there are no topographical cues at the interface between the two density zones would be to run an AFM scan of the interface and we have not done that yet.

A reason why cells may choose to navigate using “density guidance” may be to maintain optimal cell-substratum adhesiveness. It has been shown previously that cells migrate fastest at intermediate levels of cell-substratum adhesion strength [10]. Density contour lines may just be

regions where this optimal adhesion strength can be achieved and maintained.

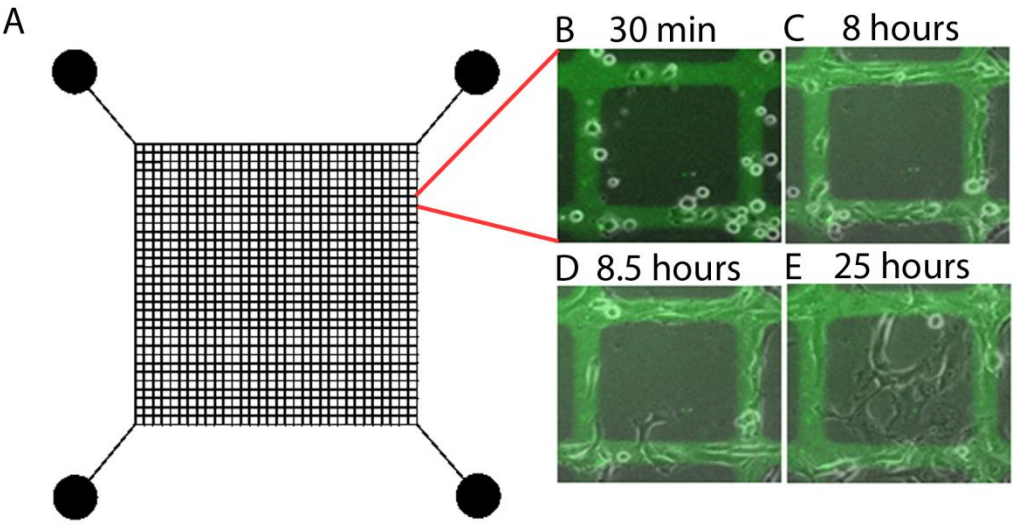

Figure 6.4. (A) Microfluidic grid device master. The side of a small square is 200 µm and the width of the channels is 40 µm. Dark circles indicate ports. Protein solution is injected through one of the ports (dark circle) until it exits through the three remaining ports. (B) Cell dynamics experiment. HT1080 cells start spreading in the areas containing type IV collagen after 30 min. (C) For the first 8 hours cells are unable to migrate in the blocked areas. (D) At 8.5 hours cells start to invade the areas previously blocked. (E) Any difference in cell density between the blocked and patterned areas is completely extinguished within 25 hours.

Materials and Methods

Microfluidic Patterns

Polydimethylsiloxane (PDMS) microfluidic devices were fabricated using soft lithography techniques described elsewhere [11]. Briefly, the device was designed in AutoCAD (Autodesk, San Rafael, CA) and printed on a commercial 35 mm film printer (Polaroid ProPallete 8000, Polaroid Corporation, Waltham, MA). A negative master was produced from the film using SU-8 2050 photoresist (Microchem Corp., Newton, MA) spun on a silicon wafer. A cast was obtained by mixing PDMS pre-polymer and curing agent (Sylgard 184, Dow Corning, Midland, MI) in a 10:1 ratio and pouring it over the SU-8 master in a polystyrene Petri

dish. The dish was placed in a vacuum desiccator and left overnight in a drying oven at 65℃ (Yamato DX400, Yamato Scientific America Inc., Santa Clara, CA). After curing in the oven, the cast was peeled from the master, cut to size and access holes were punched using a sharpened 16 gauge needle (Becton Dickinson, Franklin Lakes, NJ).

The cast device was clamped to a polystyrene Petri dish (430588 Corning Incorporated, Corning, NY) and Tygon tubing (Saint-Gobain, Akron, OH) with internal diameter of 0.508 mm and external diameter of 1.524 mm was connected to one of the ports. Human type IV collagen conjugated to OregonGreen 488 fluorophores (Molecular Probes, Invitrogen, Carlsbad, CA) was diluted in PBS (Gibco, Invitrogen, Carlsbad, CA) to a concentration of 6 μg/ml. Protein solution was injected into the device through the port with tubing until drops appeared at the other three ports. Protein was allowed to adsorb for two hours, creating zones of high protein density. Then the protein was withdrawn, and we injected PBS into the device. We then removed the device, rinsed the Petri dish twice with PBS and added protein to the entire surface of the dish in a second deposition step. Finally, we rinsed the device with PBS again and added 10% non-fat dry milk solution in PBS to block additional binding sites.

To determine how long the patterns were viable, we produced a pattern where the second deposition step was skipped, resulting in zones that were completely blocked. We then observed how long it took before the cells were able to invade the completely blocked areas (Fig. 6.4D). The first cell invaded after 8.5 hours, with complete invasion occurring after 24 hours. We therefore limited our cell migration movies to the first 8 hours after plating the cells.

Faraday Wave Patterns

We produced Faraday wave patterns by vibrating a Petri dish containing the protein solution at 250 Hz. The vibrations cause generation of standing waves in the fluid. This type of wave is called a Faraday wave. There is a large body of literature on the production of various patterns in a fluid using Faraday waves [12,13]. This has been done for the study of pattern formation in nonlinear systems, and in those studies the fluid depth is much greater than the wavelength. As such, Faraday waves cannot be used for pattern formation since any patterns are extinguished a short distance below the fluid surface. However, if the fluid layer is thin, the waves cause significant flow in the region near the bottom. Any particles present in the fluid tend to accumulate at regions on the bottom where net flow is zero. This property has been used by others to generate patterns in particulate films [14].

To generate our patterns we used a Harman/Kardon model HK206 speaker (Samsung Electro-Mechanics Co., Suwon, Korea) connected to a Dell Dimension 9200 desktop computer (Dell Computers, Round Rock, TX). Sinusoidal waves of the desired frequency were produced on the PC using the Audacity sound editor (http://audacity.sourceforge.net). A Petri dish containing a thin film of the collagen solution was placed on the speaker cone. The size of the pattern created is dependent on the wave frequency f_w and the thickness of the fluid layer H according to the equation

$$c^2 = \frac{\sigma}{\rho}\left(\frac{2\pi}{\lambda}\right) tanh\left(\frac{2\pi H}{\lambda}\right),$$

where $c = \lambda f_w$, λ is the wavelength, σ is the surface tension and ρ is the density of the fluid. By increasing the driving frequency $f_d = 2f_w$ or reducing the fluid depth H, we can reduce the feature size (Fig. 6.5).

Patterns were allowed to form for 2 hours. For the initial experiments we used f_d=200 Hz and H=0.7 mm. Afterwards, the dish was rinsed three times with PBS, and a solution of 10% non-fat dry milk in PBS was added to block additional binding sites.

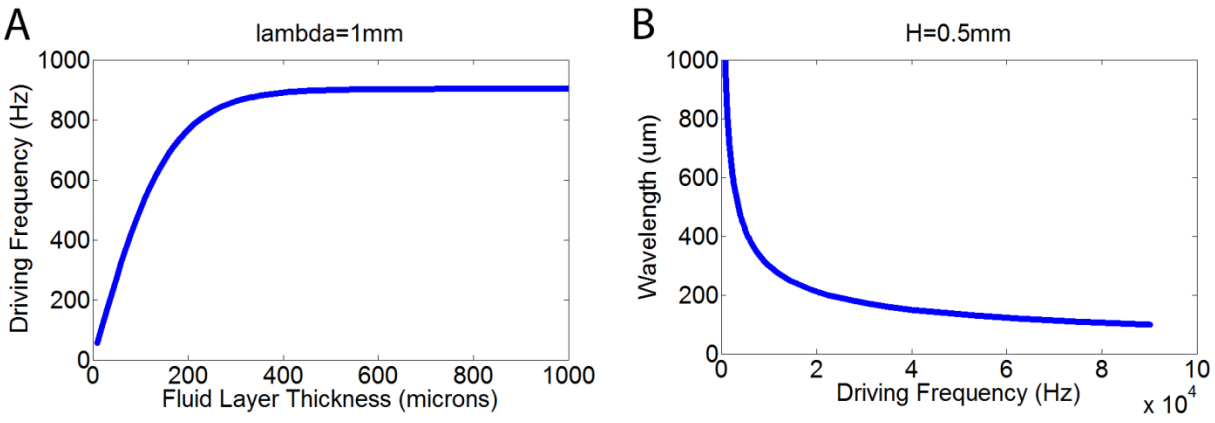

Figure 6.5. (A) The driving frequency required to produce a wavelength lambda= 1 mm decreases as the thickness of the fluid layer becomes smaller than lambda. (B) Relationship between driving frequency and wavelength at a constant fluid thickness H=0.5 mm.

Migration Movies and Gradient Calculation

Petri dishes containing type IV collagen patterns (generated using either microfluidic or Faraday waves patterning) were rinsed three times with PBS to remove any unattached protein. HT1080 human fibrosarcoma cells were trypsinized and diluted to a monocellular suspension of 3×10^5 cells/ml. Four ml of cell solution suspended in DMEM+5%FBS+P/S/G were added to the patterned Petri dish. The dish was placed in an inverted microscope equipped with an incubator hood (37 ℃, 5% CO_2). We took epifluorescent pictures of the pattern as well as background and shading pictures. We corrected for uneven and stray illumination by applying background and shading correction as described in the MetaMorph (Molecular Devices, Downingtown, PA) documentation. To correct for unwanted fluorescence due to the polystyrene

in the Petri dish, we used a simplified form of linear unmixing [15]. Lastly, bright-field time-lapse automated pictures were taken for at least 8 hours at the rate of one frame every 3 minutes in MetaMorph using a monochrome, cooled CCD camera (CoolSNAP HQ, Roper Scientific, Trenton, NJ), mounted on a Zeiss Axiovert 200M inverted microscope (Zeiss, Thornwood, NY). Movies were then imported into ImageJ (http://rsb.info.nih.gov/ij/) and manually tracked using the MTrackJ plugin.

Conclusions

We have demonstrated a technique to measure the ability of cells to detect differences in the density of surface-bound chemoattractants. This method may also provide insights into the efficiency with which cells can amplify an external density gradient. We have also observed migration along density differences, a mode of migration which we term "density guidance," and we propose that migration in this fashion is more efficient for long distance navigation, especially in cells with a limited ability to migrate using haptotaxis. We have shown that HT1080 cells have a weak ability to detect density differences in type IV collagen, but will readily migrate to the zones of high density when a difference is detected.

In the future this method may be used to test the ability of professional migrating cells, such as neutrophils, to detect changes in bound protein density. A difference in the internal mechanism of detection between these cells and occasional migrators may be indicated by a nonlinear response to changes in the density difference. In addition, it would be interesting to test if cancer cells that have undergone EMT transition are able to detect smaller differences in

protein concentration and more likely to use “density guidance” to migrate than cells that have not undergone EMT.

Acknowledgements

The authors would like to thank Allison Price for her editorial assistance.

Funding: This research is funded in part by National Institutes of Health/National Cancer Institute grants R01CA47858 to VQ, and by the Vanderbilt Institute for Integrative Biosystems Research and Education.

References

[1] A.J. Ridley, M.A. Schwartz, K. Burridge, R.A. Firtel, M.H. Ginsberg, G. Borisy, et al., Cell migration: integrating signals from front to back, Science (New York, N.Y.). 302 (2003) 1704-9.

[2] A. Müller, B. Homey, H. Soto, N. Ge, D. Catron, M.E. Buchanan, et al., Involvement of chemokine receptors in breast cancer metastasis, Nature. 410 (2001) 50-6.

[3] F. Balkwill, Cancer and the chemokine network, Nature Reviews. Cancer. 4 (2004) 540-50.

[4] S.B. Carter, Haptotaxis and the mechanism of cell motility, Nature. 213 (1967) 256-260.

[5] S.H. Zigmond, Ability of polymorphonuclear leukocytes to orient in gradients of chemotactic factors, The Journal of Cell Biology. 75 (1977) 606-616.

[6] I.C. Schneider, J.M. Haugh, Quantitative elucidation of a distinct spatial gradient-sensing mechanism in fibroblasts, The Journal of Cell Biology. 171 (2005) 883-92.

[7] J.T. Smith, J.T. Elkin, W.M. Reichert, Directed cell migration on fibronectin gradients: effect of gradient slope, Experimental Cell Research. 312 (2006) 2424-32.

[8] L. Liu, B.D. Ratner, E.H. Sage, S. Jiang, Endothelial cell migration on surface-density gradients of fibronectin, VEGF, or both proteins., Langmuir. 23 (2007) 11168-73.

[9] W. Georgescu, J. Jourquin, L. Estrada, A.R. a Anderson, V. Quaranta, J.P. Wikswo, Model-controlled hydrodynamic focusing to generate multiple overlapping gradients of surface-immobilized proteins in microfluidic devices, Lab on a Chip. 8 (2008) 238-44.

[10] S.P. Palecek, J.C. Loftus, M.H. Ginsberg, D.A. Lauffenburger, A.F. Horwitz, P.P. Deshpande, et al., Integrin – ligand binding properties govern cell migration speed through cell – substratum adhesiveness, Nature. 388 (1997) 537-540.

[11] G.M. Whitesides, E. Ostuni, S. Takayama, X. Jiang, D.E. Ingber, Soft lithography in biology and biochemistry, Annual Review of Biomedical Engineering. 3 (2001) 335-73.

[12] D. Binks, W. van de Water, Nonlinear Pattern Formation of Faraday Waves, Physical Review Letters. 78 (1997) 4043-4046.

[13] A. Kudrolli, J.P. Gollub, Patterns and spatiotemporal chaos in parametrically forced surface waves: a systematic survey at large aspect ratio, Physica D: Nonlinear Phenomena. 97 (1996) 133-154.

[14] P.H. Wright, J.R. Saylor, Patterning of particulate films using Faraday waves, Review of Scientific Instruments. 74 (2003) 4063-4070.

[15] B. Kraus, M. Ziegler, H. Wolff, Linear fluorescence unmixing in cell biological research, in: Modern Research and Educational Topics in Microscopy, 2007: pp. 863-872.

CHAPTER VII

CONCLUSIONS AND FUTURE DIRECTIONS

Conclusions

In this work we have investigated cancer cell motility in an *in vitro* system and developed tools and techniques that have been used to quantify cancer cell motility and other dynamic cell properties, such as single cell lifespan and morphology. Our study into the mechanisms responsible for increased motility in a hepsin-overexpressing prostate cancer cell line have confirmed our hypothesis that these cells interact differently with laminin-332 compared to low hepsin-expressing cells. We have also linked for the first time the overexpression of hepsin with a switch to integrin α6, the only integrin whose expression has been shown to increase in prostate cancer. Our study has also shown that the hepsin-overexpressing cells have a more mesenchymal morphology than their low hepsin-expressing counterparts, indicating a possible EMT transition in these cells. Mesenchymal morphology has been associated with increased motility, and in our study we have confirmed that these cells have increased motility *in vitro*. In addition, we have shown for the first time that this enhanced motility is due to an increase in speed and not due to a change in the directional persistence of the cells.

During the course of this work we have also developed a microscopy framework that improves on the current state-of-the-art in high-throughput cell tracking. In addition to providing us with the necessary means to investigate prostate cancer motility at the single cell level, the framework has been used in a variety of other studies both in our labs and other labs, here at Vanderbilt and abroad. Using the assays available in CellAnimation, we and other researchers

have extracted large sets of dynamic single cell properties that, in addition to their particular value to each individual study, will also prove very useful for the parameterization of future cancer cell models, such as the ones being developed at the Center for Cancer Systems Biology at Vanderbilt University.

Finally, we have developed a simple yet powerful method that will allow quantification of haptotaxis potential in cancer cell lines. We have used it to demonstrate the haptotaxis ability of a fibrosarcoma cell line and have developed a series of metrics that can be used not only to compare the efficiency of haptotaxis sensing in different cell lines or for different extracellular matrix proteins, but that may also provide clues to the dynamics of the internal mechanisms responsible for gradient detection. Our method provides a way to further the understanding of haptotaxis, which has been largely neglected by researchers in the field of directed migration not in small part due to the complexity of the techniques presently required for its study.

Future Directions

Like most scientific endeavor, in addition to providing answers to our questions, our work has opened a large number of avenues for future research. In our hepsin-overexpression studies we have identified a possible signaling pathway through which hepsin can affect cell motility and invasion. We plan to test our hypothesis by blocking the effect of hepsin and other components of the hypothetical signaling pathway such as uPA and FAK. We would also like to know if the fragment of laminin-332 β3 chain that is being cleaved by hepsin plays a role in the observed increase in motility. The protein core at Vanderbilt is working on producing this fragment of the β3 chain. Once it has been produced, we can test its effects on the low hepsin

expressing cell line. We will then know if it causes an increase in cell motility and a switch in integrin adhesion similar to those observed in the hepsin-overexpressing line. We are also interested in creating an antibody to the β3 fragment. This will allow us to block fragment activity in the hepsin-overexpressing cell line and observe if it leads to a decrease in motility and a switch away from integrin α6β1.

We would also like to investigate the effects of hepsin on directed cell migration. The method we developed in aim III can be used to determine if there is an increased ability of the hepsin-overexpressing cells to detect changes in surface-bound protein density. These cells have been shown to have increased invasion *in vivo*, compared to low hepsin-expressing cells, and an increased ability to navigate using cues provided by changes in substrate-bound protein density may be a way to achieve it. The methods we have developed in this dissertation will allow us to explore haptotaxis with an unprecedented level of detail. This will greatly expand knowledge in the field. One may test, for example, if different mechanisms of gradient sensing are at work in cells that are professional migrators such as neutrophils and cells that migrate only occasionally. This has been shown to be the case in chemotaxis. For haptotaxis, the mechanism through which the cells sense and possibly amplify the gradient is not yet known.

We can now test the effects of blocking different pathways with known roles in cell motility on the haptotaxis ability of cells. We expect that blocking pathways involved in gradient sensing will result in cells exhibiting no preference or a very small preference for the areas of high protein density as compared to cells where the pathway has not been blocked. Using this method, we can quickly test the ability of different substances to induce a haptotactic response, and test the strength of that response. Because our method extracts individual cell responses we will be able to observe the heterogeneity of the cell response to different stimuli for haptotaxis.

Even if only a small percentage of the cell population responds to a stimulus, it may lead to a significant clinical outcome in a patient. For example, only a few cells may escape the primary tumor and intravasate but the result may be the formation of a secondary tumor.

Another interesting avenue of future research is the behavior we termed "density guidance". We plan on using atomic force microscopy (AFM) to test our hypothesis that the interface separating the high and low surface density zones contains no topographical cues for migration. If this hypothesis is proved correct we plan on testing multiple cell lines that vary in their metastatic potential and their migratory phenotype. It is possible that, *in vivo,* cells that are following a change in topography are following a change in density as well. This can occur in a groove, for example, where some parts of the substrate become less accessible to the cell. This in turn can lead to a sharp change in the availability of substrate at the groove edges in effect creating a sharp density interface. It should be possible to test the effect of changes in ECM protein density on contact guidance. One way would be to vary the density of protein being deposited in the high and low areas of the topography and observing the effect on contact guidance.

Another test which may be performed is deposition of protein patterns going across the direction of topographical cues. One can then observe if cells bias their migration to follow the topography or if they choose instead to follow along the areas of sharp density change and across the topography. Still another way would be to vary the accessibility of the ECM by using a substrate where narrow grooves of different depths have been created. The grooves must be narrow to prevent cells from accessing the bottom of the grooves by crawling inside them. One can create deep grooves where the ECM substrate is not accessible and shallow grooves where the ECM substrate can be accessed by cells migrating above. This assertion may be tested by

blocking adhesion everywhere except the bottom of the grooves. In this case, the expectation would be that cells in areas containing deep grooves will be unable to adhere, while cells in areas with shallow grooves will be able to successfully adhere to the substrate. Reversing the deposition pattern and blocking the groove floors instead, should result in contact and density guidance in both cells. By substituting the blocking protein at the bottom of the grooves with the same ECM protein used everywhere else, one should be able to largely eliminate the difference in density in the shallow grooves while maintaining the topography difference. This may result in a marked difference between cells migrating on shallow grooves and cells migrating on deep grooves, if density guidance plays a role in contact guidance.

Another area of future development is our high-throughput analysis pipeline. CellAnimation has given us the capability to do high-throughput extraction of dynamic cell properties such as motility, morphology and lifespan in both 2D and 3D. This will allow us to test the response of many different cell lines to many different compounds in a short amount of time. A limitation to this is that our tracking assays work best with nuclearly-labeled cells which requires additional time initially. However, once a cell line has been labeled it can be subsequently used for any number of tests. We will be able to use our assays to quickly screen through a drug library for drugs that have the desired effects on cell motility or cell cycle. Because we are extracting data with single cell level resolution, we will be able to quickly spot areas of potential trouble. One such area may be the presence of a cell subpopulation that is resistant to the effects of the drug. By destroying the reminder of the cells and allowing only a small subset of potentially more invasive cells to proliferate a drug can actually turn a less aggressive tumor into a tumor that quickly metastasizes to other areas of the body.

Our high-throughput analysis pipeline can also be used to evaluate the response of cells from an individual patient tumor to a particular set of drugs. This will help in customizing the treatment, so that the combination of drugs used is the one which has been shown to be the most efficacious for that particular patient. Of course, this can only be done for drugs where there is a known correlation between the *in vitro* response of tumor cells to the drug and the *in vivo* activity of the drug.

As the number of assays and users of CellAnimation grows, so does the number of experimental settings for which an assay has been created, tested and validated. The problem right now is that many users are not aware of the existence of a particular assay, or they are not sure which particular assay they should use from a number of similar assays. In the future, we plan to implement a database of assays that will provide detailed descriptions of individual assays along with sample data and recommended experimental setup. However, as the number of assays is already quite large it becomes tedious for a potential user to read the detailed description of every individual assay. This in turn may once again result in the user selecting an inappropriate assay that will not perform well for the particular experimental setup. This can lead to generation of inaccurate data. This data may need extensive manual correction or worse can lead to wrong conclusions if the results of the automatic data extraction are accepted as-is.

To prevent this problem, we plan on developing a web application that will allow the user to quickly narrow down the list of potential assays based on a set of tags. The tags will help the user filter the selection not only based on assay type, such as tracking or colony counting, but on other parameters that are relevant to the user. At present, to our knowledge, there are no applications that allow a user to customize an analysis based on anything other than assay type and, possibly, the type of illumination used. This is in no small part due to the large amount of

overhead that will result from maintaining multiple versions of what is essentially the same assay.

Because we created CellAnimation to be very modular, we don't incur the same performance penalty as a typical microscopy application. We can maintain multiple versions of the same assay with minimal cost in both code complexity and memory use. Another advantage of CellAnimation, is that it can easily export and import the settings for each image processing module using small text files. Many times the difference between one experimental setup and another can be accounted by adjustments to these parameters with no changes to the underlying code.

This flexibility will allow us to maintain multiple sets of the same assay that have been heavily optimized for a particular application. The user will then be able to zoom in on a particular assay, not only based on assay and illumination type, but also on many other parameters that affect the quality of the extracted data such as range of cell morphology and motility parameters, microscope type, frame rate, etc. For example, a user will be able to select a cell tracking assay for directional, fast-moving cells, stained using mRFP and imaged with a BD Pathway microscope. Of course, not all combinations will be available, but being able to get at least close to the experimental setup used with a particular set of images will make it much easier and faster to accurately analyze the image set. Once users have optimized the assay for their particular experimental setup, they will be able to upload and tag their particular data set so that future users can take advantage of their optimizations.

We are also interested in creating a set of tagged data sets for the developers of microscopy assays not just the users. During our manual validation of CellAnimation assays, we

have created a set of data files which we used to validate our accuracy in tracking and mitotic event detection. We plan to make this data publicly available, so that developers of similar applications can evaluate the performance of their implementations. Because CellAnimation can easily incorporate external code, it will also be possible to use it to test individual algorithms.

We intend to develop an application that will allow developers to upload their tracking or segmentation algorithm. The application will then wrap the external algorithm into a CellAnimation shell and execute the test assay against one or multiple manually-validated data sets. Because both the data sets will be manually validated and all the auxiliary image processing code will be identical the only differences in tracking should be due to differences in the performance of the external algorithm. This approach can be easily implemented for segmentation algorithms where the input is an image and the output is another image or a set of object outlines. It is more difficult for tracking algorithms, where the structure of the algorithm output is unknown, but it could be implemented by requiring the developer to provide an auxiliary structure which will map the output of the algorithm to a standard layout. By using this application, developers will be able to rapidly obtain a trusted ranking of the performance of their algorithms against others.

In addition to these future avenues specific to our research interests, the set of tools we have developed, and especially the microscopy framework, are broad-purpose tools that may be used to answer any number of scientific questions. The adoption by other groups of CellAnimation to study problems ranging from the dynamics of focal adhesions in 2D and 3D to tracking of labeled RNA in cell cultures is a testament to both the usefulness and the flexibility of our software. We are confident that in the future both the number of users benefitting from our

application and the diversity of scientific settings in which CellAnimation plays a part will continue to increase.

Another benefit on the increasing user base of CellAnimation is the large amount of accurate single cell level data being generated. There is an effort in our lab to make these data sets publicly available. Some data sets have already been updated to the NCI ICBP portal, where if not publicly available, they are at least available to a wider research community. This data will be of particular interest to people involved in the mathematical modeling of cancer. A problem that is encountered when building *in silico* models of cancer is the lack of numerical distributions for dynamic cell processes such as motility, morphology, metabolism and lifespan. In our own data, we have observed large variations in many of the parameters we have measured including cell speed and lifespans. In many biological papers, values are reported only as means or worse, from the point of view of the mathematician, using a qualitative measure. In these cases, modelers end up with a model that fails to capture the heterogeneity of the cell population by either representing all cells as having uniform behavior or by using a set of numerical values that are biologically incorrect. Predictions from such models are less likely to be accurate and are hard if not impossible to test because the described behavior is not representative of the behavior of the cells *in vitro*.

The data sets we are providing offer a rich numerical description of the cell. Because our data sets capture heterogeneous behavior, it is more likely that the models incorporating this data will be able to generate realistic behaviors in response to simulated perturbations such as drug response or response to hypoxic stress in a tumor. Because the simulations are based on biologically correct ranges, changes in behavior that arise as a result of a simulation can be tested *in vitro*. This in turn should lead to the establishment of a productive cycle where predictions

from models using our data sets are used iteratively to set up *in vitro* experiments, dynamic data from those experiments is extracted using CellAnimation, and the new data sets are used to improve the accuracy of the mathematical model.

CHAPTER VIII

APPENDICES

Appendix 1: CellAnimation Help File

Cell Animation

Introduction

CellAnimation is a framework for microscopy assays. To allow for fast assay creation and code reuse each assay is implemented as a modular pipeline. A CellAnimation assay is a chain of MATLAB structures. Each structure describes a module and its connectivity. We have provided a series of assays that we have developed for various microscopy tasks for our lab and others. The intended purpose for each of the assays is described in the "Assays" section.

The CellAnimation core functions are responsible for reading the module chain, creating a dependency tree, populating the input values, executing each module and saving those output values required by modules further downstream. A module is a reusable set of functions that has a narrow specific use (image input-output, thresholding, segmentation, annotation, etc.). Each module we provide is documented in the "Internal Modules" section. We have also included a set of control modules. These are special modules that operate on other modules. Through the use of control modules we can implement looping and branching at the pipeline level. The benefits to this approach are two-fold. First, the assay logic becomes easier to follow and modify. Second, it

allows us to use smaller, more reusable modules. Individual control modules are documented in the "Control Modules" section.

In addition to our modules, a pipeline may include modules that are just simple wrappers for MATLAB functions or modules that encapsulate functions developed by others. For the MATLAB wrapper modules the documentation may be found in the in the MATLAB help file for that particular function. The caveat is that a particular wrapper may only provide access to some of the functionality of the MATLAB function. In general, MATLAB wrapper modules are named using the name of the MATLAB function concatenated with the string "Wrapper" however, there are some older wrapper modules that only use the name of the MATLAB function. For other external modules the documentation is provided by the original developers.

Assay Editor

Usage

The assay editor is used to create new assays and edit existing assays. To start the assay editor make sure the current directory in MATLAB is set to the directory containing the CellAnimation source files and type *assayEditorGUI* at the MATLAB command prompt.

Main Window

The main window of the assay editor displays all the available CellAnimation modules, the modules in the current assay and a description of the currently selected assay. The available modules are displayed in the top left corner listbox. Clicking on a module in the listbox shows its description in the textbox below. The listbox on the right shows the modules in the current assay. Modules whose names are in a bold font are control modules and have their own submodules. The submodule lists are indicated by an indented italic font immediately below the name of the

control module. Double-clicking on a submodule list expands it so the submodules it contains become visible. Double-clicking the submodule list again collapses it. Double-clicking a module name in the *Current Assay* listbox opens the *Edit Module Parameters* dialog box.

Editing an Assay

Open the assay to be edited using *File|Open.* In the *Open Existing Assay* dialog box the listbox on the left shows the available assays while the textbox on the right shows a description of the currently selected assay. To open an assay, double-click its name or select it and then click the *OK* button. Once an assay is opened its main modules become visible in the current assay listbox. There are three different levels of assay editing. Most users will only need to change script parameters of an existing assay to get the assay to work with their data set.

To modify a script variable, click on *Edit|Script Variables*. The *Edit Script Variables* dialog box displays the script variables listed in order of change frequency in the *Name* listbox. Variables whose values are likely to be changed often are displayed at the top of the list while variables that are unlikely to change are displayed towards the bottom of the list. A description of each variable can be found in the assay .m file or in this help file. To change the value of a variable, click on its name, then type a new value in the *Value* textbox on the right.

Advanced users can change values of variables at the individual module level. To change the value of module variables, double-click the desired module level in the *Current Assay* listbox in the main window. This opens the *Edit Module Parameters* dialog box, which displays the input values to the module along with the providers of those values. Each input argument name is displayed in regular font in the *Input Arguments* listbox. Below each argument name is a list of providers for that argument in indented italic font. Input arguments for which no provider has

been yet specified are displayed in red. In general each input argument needs at least one provider with the exception of optional arguments. There are two types of providers and they are indicated by the prefix *Value* or *Output*.

Providers starting with the prefix *Value* are constant values that are entered manually by the user. To change a static value, click on a static provider name (usually *Value1*) and type a new value in the *Manual Value* textbox. The value is automatically saved when selecting something else in the *Input Arguments* box or clicking the *OK* button. String values such as file names must be surrounded by single quotation marks. Text entered without quotation marks is interpreted either as a script variable name or a number, if the text consists only of digits. To add a new static value provider, click on the input argument name, enter the value in the *Manual Value* textbox, then click on the *Add Manual Value* button.

Providers starting with the prefix *Output* are dynamic values that are provided at runtime by another module. To modify the provider for a dynamic value, click on a dynamic provider name (*Output1*, *Output2*, etc.) then select another module from the *Module Instance* popup box and/or another output argument from the *Output Argument* popup box. To add a new dynamic provider, click on the input argument name for which you wish to add the provider, select the appropriate module instance and output argument from the popup boxes on the right, then click on *Add Output Argument*.

It is not unusual for an input argument to have both a static and a dynamic provider, with the dynamic provider downstream from the module receiving the value. The static provider is used as an initial value, while subsequent values are obtained from the dynamic provider and change

as the assay iterates. The values from the dynamic provider overwrite the static ones as soon as the module providing the value is called.

Creating a New Assay

To create a new assay, click on *File|New*. To add a module to the assay select it in the *Available Modules* listbox, then click on the button displaying an arrow pointing to the right. In the dialog box that appears type an instance name for the current module. The instance name must be a unique name that is not used by any other modules in the assay. You may also select a parent module from the popup box. If the parent module is blank the module will be added as a main module. If the parent module is not blank the module will be added as a submodule belonging to a control module. The choice between a main module and a submodule depends on the intended purpose of the module. A main module will be called exactly once while a submodule may be called multiple times if it's part of a control module that iterates, such as a forLoop module. Once a module instance has been filled in, click the *OK* button. A module that has been added by mistake can be removed by selecting it in the *Current Assay* listbox, then clicking on the button displaying the red X shape.

After the module has been added, providers need to be specified for each of its input arguments. A provider is a value specified either statically when the assay is edited or provided dynamically by one of the modules when the assay is run. To specify providers, double-click on the module, then specify dynamic or static parameters as described in the "Editing an Assay" section.

Modules will be executed in the order they appear in the current assay listbox so you need to make sure that each module appears in its proper place. For example, a *readImage* module should appear above any modules that operate on that image. To change the position of a module

in the assay, select it then click either the up arrow button or the down arrow button to move the module up or down respectively. As the module is moved it may become part of a submodule list or exit a submodule list and become a main module instead. This is indicated by the indent level of the module. Main modules are not indented from the left side of the *Current Assay* listbox while submodules are indented one or more spaces from the left side depending on the level of their parent module.

If a parameter is common to several modules or it's likely to change every time an assay is run we recommend you add it as a script variable. To add a script variable, click on *Edit*|*Script Variables*. In the *Edit Script Variables* dialog box click on the *Add* button. Type the name of the new script variable in the *Name* textbox, then click the *OK* button. This name should be different from any existing script variables belonging to this assay. Type the value of the new script variable in the *Value* textbox, then click on the *OK* button or another script variable name to save it. In the same way as with module variables, a string literal must be enclosed within single quotation marks or it will be interpreted as a script variable name or a number if it consists only of digits. An example of a script variable may be an image file name that is used both by the *readImage* module and the *saveImage* module and is likely to change every time the assay is run. Instead of editing the file name every time the assay is run in both modules, one may add a script variable named *ImageFileName* and set its value to a string literal containing the image file name, for example *'C:/test/image1.jpg'*. Then in the modules, instead of entering the string literal one can simply enter *ImageFileName* and the assay will retrieve the value from the script variable. The value of the script variable can be easily changed when the assay is run on another image as described in "Editing an Assay" and both modules will receive the updated value.

Finally, we recommend adding an assay description so others can easily see the purpose of the assay when they click on it. An assay description can be added by clicking on *Edit|Assay Description* and typing the assay description in the *Assay Description* textbox. Once all the modules are in their proper positions and the input parameters have at least one provider, save the assay using *File|Save*. A saved assay may be run using *File|Run* or by typing the assay name at the MATLAB command prompt.

Exporting Script Variables

Once an assay has been optimized for a particular set of images we recommend exporting the set of optimized script variables to the directory containing the images using *Tools|Export Script Variables*. This way if the analysis needs to be repeated at a later date it can be easily done by importing the optimized variables from the image directory using *Tools|Import Script Variables*.

Incorporating External MATLAB Functions

At times functionality may be required that is not available as a CellAnimation module but is available in some MATLAB function written by others. The assay editor makes it easy to access such functions via a CellAnimation wrapper module. The function which is to be made available to CellAnimation needs to be in a folder that is part of the MATLAB path. The MATLAB path can be modified using *File|Set Path* from the main MATLAB window. To wrap an external function in a CellAnimation module, click on *Tools|Wrap MATLAB Function*. Select the MATLAB function to wrap using the dialog box. A wrapper module will be automatically generated and a dialog box presented to the user that allows selection of the directory and file name where the wrapper module is to be saved. We recommend using the automatically generated file name as it makes it easy to recognize the module as a wrapper module. The wrapper module needs to be saved in the directory where the CellAnimation source files reside.

Assays

assayBrightFieldCytoTestNN

Usage

This assay has been optimized for tracking cells imaged using bright-field microscopy using a nearest-neighbor algorithm. The assay tracks the cells, detects mitotic events and records tracking data, cell shape parameters and ancestry information to spreadsheets. For each tracked image, an overlayed image is saved to disk that displays the detected cell outlines, cell IDs and cell generation.

Script Variables

ImageFolder - String variable that specifies the absolute location of the directory which contains the time-lapse images. An example of such a string variable would be *'c:/sample images/high-density'*.

ImageFilesRoot - String variable specifying the root image file name. The root image file name for a set of images is the image file name of any of the images without the number or the file extension. For example, if the file name is *'Experiment-0002_Position(8)_t021.tif*' the root image file name will be *'Experiment-0002_Position(8)_t'*.

ImageExtension – String variable specifying the image file extension including the preceding dot. For example, if the file name is *'image003.jpg'* the image extension is *'.jpg'*.

StartFrame – Number specifying the first image in the sequence to be analyzed. The minimum value for this variable depends on the numbering of the image sequence, so if the first image in the sequence is 'image003.tif' then the minimum value is 3.

FrameCount – Number specifying how many images from the image sequence should be processed.

TimeFrame – Number specifying the time between consecutive images in minutes.

FrameStep - Number specifying the step size when reading images. Set this variable to 1 to read every image in the sequence, 2 to read every other image and so on.

NumberFormat – String value specifying the number of digits in the image file names in the sequence. For example, if the image file name is '*image020.jpg'* the value for the *NumberFormat* is *'%03d'*, while if the file name is '*image000020.jpg'* the value should be *'%06d'*.

MaxFramesMissing – Number specifying for how many frames a cell may disappear before its track is ended.

OutputFolder – The folder where the overlayed images and track data will be saved. By default this value is set to a folder named *'output'* within the folder where the images to be analyzed are located.

AncestryFolder – The folder where the overlayed images and ancestry data will be saved. By default this value is set to a folder named *'ancestry'* within the output folder.

AncestrySpreadsheet – The path name to the spreadsheet containing the ancestry data. By default this value is set to a file named *'ancestry.csv'* within the ancestry folder.

ShapesSpreadsheet – The path name to the spreadsheet containing the position and shape properties for each cell in the time-lapse sequence at every time point. By default this is set to to a file named *'shapes.csv'* within the ancestry folder.

TracksFolder – The folder where the label matrixes containing the cell outlines are saved. By default this value is set to a folder named *'track'* within the output folder.

SegmentationFilesRoot – The root file name of the label matrixes containing the cell outlines.

ImageFileBase – The path name to the images. This value is generated from the *ImageFolder* and the *ImageFilesRoot* and should not be changed.

GradientThreshold – Number specifying the threshold value for the image generated by the *generateBinImgUsingGradient* filter. Any pixels in the gradient image below this value will be set to zero while the rest will be set to one.

ObjectArea – Number specifying the threshold area for the areaFilterLabel, clearSmallObjects, segmentObjectsUsingClusters filters. Objects below this value will be removed from the filtered image.

ObjectReduce – Number specifying how much a cluster of objects should be reduced before it is processed by the *segmentObjectsUsingClusters* module. By default this value is set to 1 and it should not be changed unless *segmentObjectsUsingClusters* throws an out-of-memory error. In that case the value can be reduced to something lower than 1.

ClusterDist – Number specifying the height at which the hierarchical cluster tree will be cut to determine the number of clusters. Setting this value higher results in a cluster being split into fewer objects by the *segmentObjectsUsingClusters* module.

ClearBorder – Boolean value specifying whether objects next to or touching the image border in the binary images generated by the *generateBinImgUsingGradient* module will be erased (*true*) or not (*false*).

ClearBorderDist – Number specifying how close to the border objects may be and still be erased if the *ClearBorder* parameter is set to *true* in the *generateBinImgUsingGradient* module.

ResizeImageScale – Number specifying by what ratio the images will be resized before they are processed. By default this value is set to 0.5. Used by the *resizeImage* module.

ApproximationDistance – Number specifying how close the convex hull in the *getConvexObjects* module approximates the object outline. By default this value is set to 2.5. Setting it to a lower value will result in convex hulls that more closely resemble the object outlines, however this increases the chance of detecting insignificant concavities.

MaxMergeDistance – Number specifying the maximum distance that one track may be from another track for the duration and still be considered for possible merging with the other track. Used by *detectMergeCandidatesUsingDistance* module.

MaxSplitArea – Number specifying the maximum area a nucleus may be and still be considered as a part of a possible mitotic event. Used by *detectMitoticEvents* module.

MaxSplitDistance – Number specifying the maximum distance a new nucleus may be from another nucleus and still be considered as part of a possible mitotic event. Used by *detectMitoticEvents* module.

MinSplitEccentricity – Number specifying the minimum eccentricity a new nucleus may have and still be considered as part of a possible mitotic event. Used by *detectMitoticEvents* module.

MaxSplitEccentricity – Number specifying the maximum eccentricity a new nucleus may have and still be considered as part of a possible mitotic event. Used by *detectMitoticEvents* module.

MinTimeForSplit – Number specifying the minimum time in minutes a track needs to exist before it is considered for a possible mitotic event. Used by *detectMitoticEvents* module.

MinLifespan – Number specifying the minimum length in frames a frame has to be to not be removed by the *removeShortTracks* module.

Important Modules

areaFilterLabel, assignCellToTrackUsingNN, clearSmallObjects, detectMergeCandidatesUsingDistance, detectMitoticEvents, generateBinImgUsingGradient, removeShortTracks, segmentObjectsUsingClusters, segmentObjectsUsingMarkers, splitTracks.

assayBrightFieldCytoTestWG

Usage

This assay has been optimized for tracking cells imaged using bright-field microscopy using a custom algorithm. This algorithm is more accurate and flexible than the nearest-neighbor algorithm at the expense of execution speed. The assay tracks the cells, detects mitotic events and records tracking data, cell shape parameters and ancestry information to spreadsheets. For each tracked image, an overlayed image is saved to disk that displays the detected cell outlines, cell IDs and cell generation.

Script Variables

ImageFolder - String variable that specifies the absolute location of the directory which contains the time-lapse images. An example of such a string variable would be *'c:/sample images/high-density'*.

ImageFilesRoot - String variable specifying the root image file name. The root image file name for a set of images is the image file name of any of the images without the number or the file extension. For example, if the file name is *'Experiment-0002_Position(8)_t021.tif'* the root image file name will be *'Experiment-0002_Position(8)_t'*.

ImageExtension – String variable specifying the image file extension including the preceding dot. For example, if the file name is *'image003.jpg'* the image extension is *'.jpg'*.

StartFrame – Number specifying the first image in the sequence to be analyzed. The minimum value for this variable depends on the numbering of the image sequence, so if the first image in the sequence is ‘image003.tif’ then the minimum value is 3.

FrameCount – Number specifying how many images from the image sequence should be processed.

TimeFrame – Number specifying the time between consecutive images in minutes.

FrameStep - Number specifying the step size when reading images. Set this variable to 1 to read every image in the sequence, 2 to read every other image and so on.

NumberFormat – String value specifying the number of digits in the image file names in the sequence. For example, if the image file name is ‘*image020.jpg’* the value for the *NumberFormat* is *'%03d'*, while if the file name is ‘*image000020.jpg’* the value should be *‘%06d’*.

MaxFramesMissing – Number specifying for how many frames a cell may disappear before its track is ended.

OutputFolder – The folder where the overlayed images and track data will be saved. By default this value is set to a folder named *‘output’* within the folder where the images to be analyzed are located.

AncestryFolder – The folder where the overlayed images and ancestry data will be saved. By default this value is set to a folder named *‘ancestry’* within the output folder.

AncestrySpreadsheet – The path name to the spreadsheet containing the ancestry data. By default this value is set to a file named *‘ancestry.csv’* within the ancestry folder.

ShapesSpreadsheet – The path name to the spreadsheet containing the position and shape properties for each cell in the time-lapse sequence at every time point. By default this is set to to a file named *'shapes.csv'* within the ancestry folder.

TracksFolder – The folder where the label matrixes containing the cell outlines are saved. By default this value is set to a folder named *'track'* within the output folder.

SegmentationFilesRoot – The root file name of the label matrixes containing the cell outlines.

ImageFileBase – The path name to the images. This value is generated from the *ImageFolder* and the *ImageFilesRoot* and should not be changed.

GradientThreshold – Number specifying the threshold value for the image generated by the *generateBinImgUsingGradient* filter. Any pixels in the gradient image below this value will be set to zero while the rest will be set to one.

ObjectArea – Number specifying the threshold area for the areaFilterLabel, clearSmallObjects, segmentObjectsUsingClusters filters. Objects below this value will be removed from the filtered image.

ObjectReduce – Number specifying how much a cluster of objects should be reduced before it is processed by the *segmentObjectsUsingClusters* module. By default this value is set to 1 and it should not be changed unless *segmentObjectsUsingClusters* throws an out-of-memory error. In that case the value can be reduced to something lower than 1.

ClusterDist – Number specifying the height at which the hierarchical cluster tree will be cut to determine the number of clusters. Setting this value higher results in a cluster being split into fewer objects by the *segmentObjectsUsingClusters* module.

ClearBorder – Boolean value specifying whether objects next to or touching the image border in the binary images generated by the *generateBinImgUsingGradient* module will be erased (*true*) or not (*false*).

ClearBorderDist – Number specifying how close to the border objects may be and still be erased if the *ClearBorder* parameter is set to *true* in the *generateBinImgUsingGradient* module.

ResizeImageScale – Number specifying by what ratio the images will be resized before they are processed. By default this value is set to 0.5. Used by the *resizeImage* module.

ApproximationDistance – Number specifying how close the convex hull in the *getConvexObjects* module approximates the object outline. By default this value is set to 2.5. Setting it to a lower value will result in convex hulls that more closely resemble the object outlines, however this increases the chance of detecting insignificant concavities.

MaxMergeDistance – Number specifying the maximum distance that one track may be from another track for the duration and still be considered for possible merging with the other track. Used by *detectMergeCandidatesUsingDistance* module.

MaxSearchRadius – Number specifying the absolute lower bound for the search radius to prevent selecting too few candidate objects for a track. Used by *assignCellToTrackUsingAll* module.

MinSearchRadius – Number specifying the absolute higher bound for the search radius to prevent selecting too many candidate objects for a track. Used by *assignCellToTrackUsingAll* module.

MinSecondDistance – Number specifying the minimum significant distance between the closest candidate object to a track and the second closest. Used to determine when distance should be used as a ranking parameter. Used by *assignCellToTrackUsingAll* module.

MaxDistRatio – Number specifying the maximum allowed distance ratio between the two nearest candidate objects. If the ratio is higher than this value, distance ranking will not be used. Used by *assignCellToTrackUsingAll* module.

MaxAngleDiff – Number specifying the maximum allowed angle difference between a track and a candidate object. If the angle is larger than this value, direction ranking will not be used for this object. Used by *assignCellToTrackUsingAll* module.

NrParamsForSureMatch – Number specifying the minimum number of closest matches between a candidate object parameters and a track's object parameters that make the candidate object a sure match to the track. Used by *assignCellToTrackUsingAll* module.

SearchRadiusPct – Number specifying the size of the neighborhood from which candidate objects for matching the track are selected. It is a multiple of the distance to the nearest candidate in the current frame. Setting this variable equal to 1 turns this module into a nearest-neighbor algorithm (only the nearest cell can be a candidate). It does not make sense to have a value lower than 1. Used by *assignCellToTrackUsingAll* module.

MaxSplitArea – Number specifying the maximum area a nucleus may be and still be considered as a part of a possible mitotic event. Used by *detectMitoticEvents* module.

MaxSplitDistance – Number specifying the maximum distance a new nucleus may be from another nucleus and still be considered as part of a possible mitotic event. Used by *detectMitoticEvents* module.

MinSplitEccentricity – Number specifying the minimum eccentricity a new nucleus may have and still be considered as part of a possible mitotic event. Used by *detectMitoticEvents* module.

MaxSplitEccentricity – Number specifying the maximum eccentricity a new nucleus may have and still be considered as part of a possible mitotic event. Used by *detectMitoticEvents* module.

MinTimeForSplit – Number specifying the minimum time in minutes a track needs to exist before it is considered for a possible mitotic event. Used by *detectMitoticEvents* module.

MinLifespan – Number specifying the minimum length in frames a frame has to be to not be removed by the *removeShortTracks* module.

FrontParams – Numeric array specifying a set of column indices from the shape and motility parameters matrix. The parameters in those columns will be heavily weighted, and have more influence in determining the best match for a track from a list of objects. Used by *assignCellToTrackUsingAll* module.

DefaultParamWeights – Numeric array specifying a set of weights that is assigned to each shape and motility parameter based on its prediction power. Parameters with high prediction power are assigned high weights and parameters with low prediction power are assigned lower weights. Used by *assignCellToTrackUsingAll* module.

DistanceRankingOrder – Numeric array specifying the default order of shape and motility parameters for slow-moving objects when it cannot be determined based on prediction power. Used by *assignCellToTrackUsingAll* module.

DirectionRankingOrder – Numeric array specifying the default order of shape and motility parameters for fast-moving directional objects when it cannot be determined based on prediction power. Used by *assignCellToTrackUsingAll* module.

RelevantParametersIndex – Boolean array specifying column indexes in the shape and motility matrix that have been determined to be irrelevant for tracking. This indicates to the module not to use the parameters of those columns in computing track assignment probabilities. The order of column indexes is provided in *TracksLayout* variable. Used by *assignCellToTrackUsingAll* module.

UnknownParamWeights – Numeric array specifying a set of weights to be assigned to shape and motility parameters when the prediction power of the parameters cannot be determined. Used by *assignCellToTrackUsingAll* module.

UnknownRankingOrder – Numeric array specifying the order of the shape and motility parameters when their predictive power cannot be determined. If the objects cannot be categorized as either slow-moving or fast directional the parameter order provided in this variable is used. Used by *assignCellToTrackUsingAll* module.

Important Modules

areaFilterLabel, assignCellToTrackUsingAll, clearSmallObjects, detectMergeCandidatesUsingDistance, detectMitoticEvents, generateBinImgUsingGradient, removeShortTracks, segmentObjectsUsingClusters, segmentObjectsUsingMarkers, splitTracks.

assayCellCoverage

Usage

This assay is used to determine what percentage of an image is occupied by objects.

Script Variables

ImageFileName – String variable specifying the absolute image file name of the image to be analyzed.
ImageDirectory – String variable specifying the directory where the image to be analyzed is located.
MaskFileName – String variable specifying the file name of the resulting binary image from which the object percentage is calculated.

Important Modules

manualSegmentationReview.

assayCellPropsSingleImage

Usage

This assay is used to segment objects in an image subject to manual review, then extract their shape properties.

Script Variables

ImageDirName – String variable specifying the directory where the image to be analyzed is located. An example of such a variable would be *'c:/test/'*.

FileRoot – String variable specifying the image file name without the extension. For example, if the file name is *'Experiment-0002_Position(8)_t021.tif*' the root file name will be *'Experiment-0002_Position(8)_t021'*.

ImageExt – String variable specifying the extension of the image file name. For example, if the file name is *'Experiment-0002_Position(8)_t021.tif*' the root file name will be *'.tif'*.

ImageFileName – String variable that is automatically generated from the *ImageDirName*, *FileRoot* and *ImageExt.*

OutputDir – String variable specifying the directory where the spreadsheet containing the shape properties and the label matrix containing the cell outlines will be saved.

SpreadsheetFileName – String variable specifying the file name of the spreadsheet containing the shape properties of the objects in the original image. This variable is automatically generated from the *OutputDir* and *FileRoot* variables.

LabelFileName – String variable specifying the file name of the label matrix containing the detected objects in the original image. This variable is automatically generated from the *OutputDir* and *FileRoot* variables.

BrightnessGradientThreshold – Number specifying the threshold value for the image generated by the *generateBinImgUsingLocAvg* filter. Any pixels in the gradient image below this value will be set to zero while the rest will be set to one.

ClearBorder – Boolean value specifying whether objects next to or touching the image border in the binary images generated by the *generateBinImgUsingGradient* module will be erased (*true*) or not (*false*).

ClearBorderDist – Number specifying how close to the border objects may be and still be erased if the *ClearBorder* parameter is set to *true* in the *generateBinImgUsingGradient* module.

MedianFilterSize – Number specifying the size of the median filter used by the *distanceWatershed* module. Setting this to a higher integer value will reduce the number of objects detected by the module and can be used to prevent oversegmentation.

ObjectArea – Number specifying the threshold area for the *clearSmallObjects* filter. Objects below this value will be removed from the filtered image.

Strel – String variable specifying the type of filter used to generate the local average image in *generateBinImgUsingLocAvg*. Currently *'disk'* is the only value supported.

StrelSize – Number specifying the size of the local neighborhood used to calculate the average for each pixel in the local average image generated by the *generateBinImgUsingLocAvg* module.

Important Modules

clearSmallObjects, distanceWatershed, generateBinImgUsingLocAvg, manualSegmentationReview, segmentObjectsUsingMarkers.

assayFlCytoLNCapManSegWG

Usage

This assay is used to manually review segmentation of objects labeled using a fluorescent dye after they have been segmented using another assay.

Script Variables

ImageFolder - String variable that specifies the absolute location of the directory which contains the time-lapse images. An example of such a string variable would be *'c:/sample images/high-density'*.

ImageFilesRoot - String variable specifying the root image file name. The root image file name for a set of images is the image file name of any of the images without the number or the file extension. For example, if the file name is *'Experiment-0002_Position(8)_t021.tif'* the root image file name will be *'Experiment-0002_Position(8)_t'*.

ImageExtension – String variable specifying the image file extension including the preceding dot. For example, if the file name is *'image003.jpg'* the image extension is *'.jpg'*.

StartFrame – Number specifying the first image in the sequence to be analyzed. The minimum value for this variable depends on the numbering of the image sequence, so if the first image in the sequence is 'image003.tif' then the minimum value is 3.

FrameCount – Number specifying how many images from the image sequence should be processed.

FrameStep - Number specifying the step size when reading images. Set this variable to 1 to read every image in the sequence, 2 to read every other image and so on.

NumberFormat – String value specifying the number of digits in the image file names in the sequence. For example, if the image file name is '*image020.jpg'* the value for the

NumberFormat is *'%03d'*, while if the file name is '*image000020.jpg'* the value should be *'%06d'*.

OutputFolder – The folder where the overlayed images and track data will be saved. By default this value is set to a folder named *'output'* within the folder where the images to be analyzed are located.

TracksFolder – The folder where the label matrixes containing the cell outlines are saved. By default this value is set to a folder named *'track'* within the output folder.

SegmentationFilesRoot – The root file name of the label matrixes containing the cell outlines.

ImageFileBase – The path name to the images. This value is generated from the *ImageFolder* and the *ImageFilesRoot* and should not be changed.

ApproximationDistance – Number specifying how close the convex hull in the *getConvexObjects* module approximates the object outline. By default this value is set to 2.5. Setting it to a lower value will result in convex hulls that more closely resemble the object outlines, however this increases the chance of detecting insignificant concavities.

BrightnessThresholdPct – Number specifying the percentage threshold value for the image generated by the *generateBinImgUsingLocAvg* filter. Any pixel in the original image smaller than the threshold value times the corresponding value in the local average image below this value will be set to zero while the rest will be set to one.

ClearBorder – Boolean value specifying whether objects next to or touching the image border in the binary images generated by the *generateBinImgUsingGradient* module will be erased (*true*) or not (*false*).

ClearBorderDist – Number specifying how close to the border objects may be and still be erased if the *ClearBorder* parameter is set to *true* in the *generateBinImgUsingGradient* module.

MedianFilterSize – Number specifying the size of the median filter used by the *distanceWatershed* module. Setting this to a higher integer value will reduce the number of objects detected by the module and can be used to prevent oversegmentation.

ObjectArea – Number specifying the threshold area for the *clearSmallObjects, polygonalAssistedWatershed* filter. Objects below this value will be removed from the filtered image.

Strel – String variable specifying the type of filter used to generate the local average image in *generateBinImgUsingLocAvg*. Currently *'disk'* is the only value supported.

StrelSize – Number specifying the size of the local neighborhood used to calculate the average for each pixel in the local average image generated by the *generateBinImgUsingLocAvg* module.

ResizeImageScale – Number specifying by what ratio the images will be resized before they are processed. By default this value is set to 0.5. Used by the *resizeImage* module.

Important Modules

clearSmallObjects, distanceWatershed, generateBinImgUsingLocAvg, getConvexObjects, manualSegmentationReview, polygonalAssistedWatershed, segmentObjectsUsingMarkers.

assayFlCytoLNCapTracksReviewWG

Usage

This assay is used to manually review the automatic tracks generated using a tracking assay.

Script Variables

ImageFolder - String variable that specifies the absolute location of the directory which contains the time-lapse images. An example of such a string variable would be *'c:/sample images/high-density'*.

ImageFilesRoot - String variable specifying the root image file name. The root image file name for a set of images is the image file name of any of the images without the number or the file extension. For example, if the file name is *'Experiment-0002_Position(8)_t021.tif'* the root image file name will be *'Experiment-0002_Position(8)_t'*.

ImageExtension – String variable specifying the image file extension including the preceding dot. For example, if the file name is *'image003.jpg'* the image extension is *'.jpg'*.

StartFrame – Number specifying the first image in the sequence to be analyzed. The minimum value for this variable depends on the numbering of the image sequence, so if the first image in the sequence is 'image003.tif' then the minimum value is 3.

FrameCount – Number specifying how many images from the image sequence should be processed.

TimeFrame – Number specifying the time between consecutive images in minutes.

FrameStep - Number specifying the step size when reading images. Set this variable to 1 to read every image in the sequence, 2 to read every other image and so on.

NumberFormat – String value specifying the number of digits in the image file names in the sequence. For example, if the image file name is '*image020.jpg'* the value for the *NumberFormat* is *'%03d'*, while if the file name is '*image000020.jpg'* the value should be *'%06d'*.

MaxFramesMissing – Number specifying for how many frames a cell may disappear before its track is ended.

OutputFolder – The folder where the overlayed images and track data will be saved. By default this value is set to a folder named *'output'* within the folder where the images to be analyzed are located.

AncestryFolder – The folder where the overlayed images and ancestry data will be saved. By default this value is set to a folder named *'ancestry'* within the output folder.

AncestrySpreadsheet – The path name to the spreadsheet containing the ancestry data. By default this value is set to a file named *'ancestry.csv'* within the ancestry folder.

ShapesSpreadsheet – The path name to the spreadsheet containing the position and shape properties for each cell in the time-lapse sequence at every time point. By default this is set to a file named *'shapes.csv'* within the ancestry folder.

TracksFolder – The folder where the label matrixes containing the cell outlines are saved. By default this value is set to a folder named *'track'* within the output folder.

SegmentationFilesRoot – The root file name of the label matrixes containing the cell outlines.

ImageFileBase – The path name to the images. This value is generated from the *ImageFolder* and the *ImageFilesRoot* and should not be changed.

Important Modules

manualTrackingReview.

assayFlCytoLNCapTrackWG

Usage

This assay is used to automatically track cells that have been segmented using another assay.

Script Variables

ImageFolder - String variable that specifies the absolute location of the directory which contains the time-lapse images. An example of such a string variable would be *'c:/sample images/high-density'*.

ImageFilesRoot - String variable specifying the root image file name. The root image file name for a set of images is the image file name of any of the images without the number or the file extension. For example, if the file name is *'Experiment-0002_Position(8)_t021.tif'* the root image file name will be *'Experiment-0002_Position(8)_t'*.

ImageExtension – String variable specifying the image file extension including the preceding dot. For example, if the file name is *'image003.jpg'* the image extension is *'.jpg'*.

StartFrame – Number specifying the first image in the sequence to be analyzed. The minimum value for this variable depends on the numbering of the image sequence, so if the first image in the sequence is 'image003.tif' then the minimum value is 3.

FrameCount – Number specifying how many images from the image sequence should be processed.

TimeFrame – Number specifying the time between consecutive images in minutes.

FrameStep - Number specifying the step size when reading images. Set this variable to 1 to read every image in the sequence, 2 to read every other image and so on.

NumberFormat – String value specifying the number of digits in the image file names in the sequence. For example, if the image file name is '*image020.jpg'* the value for the *NumberFormat* is *'%03d'*, while if the file name is '*image000020.jpg'* the value should be *'%06d'*.

MaxFramesMissing – Number specifying for how many frames a cell may disappear before its track is ended.

OutputFolder – The folder where the overlayed images and track data will be saved. By default this value is set to a folder named *'output'* within the folder where the images to be analyzed are located.

AncestryFolder – The folder where the overlayed images and ancestry data will be saved. By default this value is set to a folder named *'ancestry'* within the output folder.

AncestrySpreadsheet – The path name to the spreadsheet containing the ancestry data. By default this value is set to a file named *'ancestry.csv'* within the ancestry folder.

ShapesSpreadsheet – The path name to the spreadsheet containing the position and shape properties for each cell in the time-lapse sequence at every time point. By default this is set to to a file named *'shapes.csv'* within the ancestry folder.

TracksFolder – The folder where the label matrixes containing the cell outlines are saved. By default this value is set to a folder named *'track'* within the output folder.

SegmentationFilesRoot – The root file name of the label matrixes containing the cell outlines.

ImageFileBase – The path name to the images. This value is generated from the *ImageFolder* and the *ImageFilesRoot* and should not be changed.

MaxSearchRadius – Number specifying the absolute lower bound for the search radius to prevent selecting too few candidate objects for a track. Used by *assignCellToTrackUsingAll* module.

MinSearchRadius – Number specifying the absolute higher bound for the search radius to prevent selecting too many candidate objects for a track. Used by *assignCellToTrackUsingAll* module.

MinSecondDistance – Number specifying the minimum significant distance between the closest candidate object to a track and the second closest. Used to determine when distance should be used as a ranking parameter. Used by *assignCellToTrackUsingAll* module.

MaxDistRatio – Number specifying the maximum allowed distance ratio between the two nearest candidate objects. If the ratio is higher than this value, distance ranking will not be used. Used by *assignCellToTrackUsingAll* module.

MaxAngleDiff – Number specifying the maximum allowed angle difference between a track and a candidate object. If the angle is larger than this value, direction ranking will not be used for this object. Used by *assignCellToTrackUsingAll* module.

NrParamsForSureMatch – Number specifying the minimum number of closest matches between a candidate object parameters and a track's object parameters that make the candidate object a sure match to the track. Used by *assignCellToTrackUsingAll* module.

SearchRadiusPct – Number specifying the size of the neighborhood from which candidate objects for matching the track are selected. It is a multiple of the distance to the nearest candidate in the current frame. Setting this variable equal to 1 turns this module into a nearest-neighbor algorithm (only the nearest cell can be a candidate). It does not make sense to have a value lower than 1. Used by *assignCellToTrackUsingAll* module.

MaxSplitArea – Number specifying the maximum area a nucleus may be and still be considered as a part of a possible mitotic event. Used by *detectMitoticEvents* module.

MaxSplitDistance – Number specifying the maximum distance a new nucleus may be from another nucleus and still be considered as part of a possible mitotic event. Used by *detectMitoticEvents* module.

MinSplitEccentricity – Number specifying the minimum eccentricity a new nucleus may have and still be considered as part of a possible mitotic event. Used by *detectMitoticEvents* module.

MaxSplitEccentricity – Number specifying the maximum eccentricity a new nucleus may have and still be considered as part of a possible mitotic event. Used by *detectMitoticEvents* module.

MinTimeForSplit – Number specifying the minimum time in minutes a track needs to exist before it is considered for a possible mitotic event. Used by *detectMitoticEvents* module.

MinLifespan – Number specifying the minimum length in frames a frame has to be to not be removed by the *removeShortTracks* module.

FrontParams – Numeric array specifying a set of column indices from the shape and motility parameters matrix. The parameters in those columns will be heavily weighted, and have more influence in determining the best match for a track from a list of objects. Used by *assignCellToTrackUsingAll* module.

DefaultParamWeights – Numeric array specifying a set of weights that is assigned to each shape and motility parameter based on its prediction power. Parameters with high prediction power are assigned high weights and parameters with low prediction power are assigned lower weights. Used by *assignCellToTrackUsingAll* module.

DistanceRankingOrder – Numeric array specifying the default order of shape and motility parameters for slow-moving objects when it cannot be determined based on prediction power. Used by *assignCellToTrackUsingAll* module.

DirectionRankingOrder – Numeric array specifying the default order of shape and motility parameters for fast-moving directional objects when it cannot be determined based on prediction power. Used by *assignCellToTrackUsingAll* module.

RelevantParametersIndex – Boolean array specifying column indexes in the shape and motility matrix that have been determined to be irrelevant for tracking. This indicates to the module not to use the parameters of those columns in computing track assignment probabilities. The order of column indexes is provided in *TracksLayout* variable. Used by *assignCellToTrackUsingAll* module.

UnknownParamWeights – Numeric array specifying a set of weights to be assigned to shape and motility parameters when the prediction power of the parameters cannot be determined. Used by *assignCellToTrackUsingAll* module.

UnknownRankingOrder – Numeric array specifying the order of the shape and motility parameters when their predictive power cannot be determined. If the objects cannot be categorized as either slow-moving or fast directional the parameter order provided in this variable is used. Used by *assignCellToTrackUsingAll* module.

Important Modules

assignCellToTrackUsingAll, detectMitoticEvents, splitTracks.

assayFluoNuclTestCA

Usage

This module is used to track cells that have been stained with a nuclear stain using a custom CellAnimation algorithm. This algorithm is more accurate and flexible than the nearest-neighbor algorithm at the expense of execution speed. The assay tracks the cells, detects mitotic events and records tracking data, cell shape parameters and ancestry information to spreadsheets. For each tracked image, an overlayed image is saved to disk that displays the detected cell outlines, cell IDs and cell generation.

Script Variables

ImageFolder - String variable that specifies the absolute location of the directory which contains the time-lapse images. An example of such a string variable would be *'c:/sample images/high-density'*.

ImageFilesRoot - String variable specifying the root image file name. The root image file name for a set of images is the image file name of any of the images without the number or the file extension. For example, if the file name is *'Experiment-0002_Position(8)_t021.tif'* the root image file name will be *'Experiment-0002_Position(8)_t'*.

ImageExtension – String variable specifying the image file extension including the preceding dot. For example, if the file name is *'image003.jpg'* the image extension is *'.jpg'*.

StartFrame – Number specifying the first image in the sequence to be analyzed. The minimum value for this variable depends on the numbering of the image sequence, so if the first image in the sequence is 'image003.tif' then the minimum value is 3.

FrameCount – Number specifying how many images from the image sequence should be processed.

TimeFrame – Number specifying the time between consecutive images in minutes.

FrameStep - Number specifying the step size when reading images. Set this variable to 1 to read every image in the sequence, 2 to read every other image and so on.

NumberFormat – String value specifying the number of digits in the image file names in the sequence. For example, if the image file name is '*image020.jpg'* the value for the *NumberFormat* is *'%03d'*, while if the file name is '*image000020.jpg'* the value should be *'%06d'*.

MaxFramesMissing – Number specifying for how many frames a cell may disappear before its track is ended.

OutputFolder – The folder where the overlayed images and track data will be saved. By default this value is set to a folder named *'output'* within the folder where the images to be analyzed are located.

AncestryFolder – The folder where the overlayed images and ancestry data will be saved. By default this value is set to a folder named *'ancestry'* within the output folder.

AncestrySpreadsheet – The path name to the spreadsheet containing the ancestry data. By default this value is set to a file named *'ancestry.csv'* within the ancestry folder.

ShapesSpreadsheet – The path name to the spreadsheet containing the position and shape properties for each cell in the time-lapse sequence at every time point. By default this is set to a file named *'shapes.csv'* within the ancestry folder.

TracksFolder – The folder where the label matrixes containing the cell outlines are saved. By default this value is set to a folder named *'track'* within the output folder.

SegmentationFilesRoot – The root file name of the label matrixes containing the cell outlines.

ImageFileBase – The path name to the images. This value is generated from the *ImageFolder* and the *ImageFilesRoot* and should not be changed.

BrightnessThresholdPct – Number specifying the percentage threshold value for the image generated by the *generateBinImgUsingLocAvg* filter. Any pixel in the original image smaller than the threshold value times the corresponding value in the local average image below this value will be set to zero while the rest will be set to one.

ObjectArea – Number specifying the threshold area for the *clearSmallObjects, polygonalAssistedWatershed* and *areaFilterLabel* filters. Objects below this value will be removed from the filtered image.

Strel – String variable specifying the type of filter used to generate the local average image in *generateBinImgUsingLocAvg*. Currently *'disk'* is the only value supported.

StrelSize – Number specifying the size of the local neighborhood used to calculate the average for each pixel in the local average image generated by the *generateBinImgUsingLocAvg* module.

ClearBorder – Boolean value specifying whether objects next to or touching the image border in the binary images generated by the *generateBinImgUsingLocAvg* module will be erased (*true*) or not (*false*).

ClearBorderDist – Number specifying how close to the border objects may be and still be erased if the *ClearBorder* parameter is set to *true* in the *generateBinImgUsingLocAvg* module.

MedianFilterSize – Number specifying the size of the median filter used by the *distanceWatershed* module. Setting this to a higher integer value will reduce the number of objects detected by the module and can be used to prevent oversegmentation.

MinSolidity – Number specifying a threshold solidity value for the *solidityFilterLabelObjects* filter. Objects whose solidity is below this value will be removed from the filtered image.

MinAreaOverPerimeter – Number specifying a threshold AOP ratio value for the *areaOverPerimeterFilterLabel* filter. Objects with an AOP ratio smaller than this value will be removed.

ResizeImageScale – Number specifying by what ratio the images will be resized before they are processed. By default this value is set to 0.5. Used by the *resizeImage* module.

ApproximationDistance – Number specifying how close the convex hull in the *getConvexObjects* module approximates the object outline. By default this value is set to 2.5. Setting it to a lower value will result in convex hulls that more closely resemble the object outlines, however this increases the chance of detecting insignificant concavities.

MaxSearchRadius – Number specifying the absolute lower bound for the search radius to prevent selecting too few candidate objects for a track. Used by *assignCellToTrackUsingAll* module.

MinSearchRadius – Number specifying the absolute higher bound for the search radius to prevent selecting too many candidate objects for a track. Used by *assignCellToTrackUsingAll* module.

MinSecondDistance – Number specifying the minimum significant distance between the closest candidate object to a track and the second closest. Used to determine when distance should be used as a ranking parameter. Used by *assignCellToTrackUsingAll* module.

MaxDistRatio – Number specifying the maximum allowed distance ratio between the two nearest candidate objects. If the ratio is higher than this value, distance ranking will not be used. Used by *assignCellToTrackUsingAll* module.

MaxAngleDiff – Number specifying the maximum allowed angle difference between a track and a candidate object. If the angle is larger than this value, direction ranking will not be used for this object. Used by *assignCellToTrackUsingAll* module.

NrParamsForSureMatch – Number specifying the minimum number of closest matches between a candidate object parameters and a track's object parameters that make the candidate object a sure match to the track. Used by *assignCellToTrackUsingAll* module.

SearchRadiusPct – Number specifying the size of the neighborhood from which candidate objects for matching the track are selected. It is a multiple of the distance to the nearest candidate in the current frame. Setting this variable equal to 1 turns this module into a nearest-neighbor algorithm (only the nearest cell can be a candidate). It does not make sense to have a value lower than 1. Used by *assignCellToTrackUsingAll* module.

MaxMergeDistance – Number specifying the maximum distance that one track may be from another track for the duration and still be considered for possible merging with the other track. Used by *detectMergeCandidatesUsingDistance* module.

MaxSplitArea – Number specifying the maximum area a nucleus may be and still be considered as a part of a possible mitotic event. Used by *detectMitoticEvents* module.

MaxSplitDistance – Number specifying the maximum distance a new nucleus may be from another nucleus and still be considered as part of a possible mitotic event. Used by *detectMitoticEvents* module.

MinSplitEccentricity – Number specifying the minimum eccentricity a new nucleus may have and still be considered as part of a possible mitotic event. Used by *detectMitoticEvents* module.

MaxSplitEccentricity – Number specifying the maximum eccentricity a new nucleus may have and still be considered as part of a possible mitotic event. Used by *detectMitoticEvents* module.

MinTimeForSplit – Number specifying the minimum time in minutes a track needs to exist before it is considered for a possible mitotic event. Used by *detectMitoticEvents* module.

MinLifespan – Number specifying the minimum length in frames a frame has to be to not be removed by the *removeShortTracks* module.

FrontParams – Numeric array specifying a set of column indices from the shape and motility parameters matrix. The parameters in those columns will be heavily weighted, and have more influence in determining the best match for a track from a list of objects. Used by *assignCellToTrackUsingAll* module.

DefaultParamWeights – Numeric array specifying a set of weights that is assigned to each shape and motility parameter based on its prediction power. Parameters with high prediction power are assigned high weights and parameters with low prediction power are assigned lower weights. Used by *assignCellToTrackUsingAll* module.

DistanceRankingOrder – Numeric array specifying the default order of shape and motility parameters for slow-moving objects when it cannot be determined based on prediction power. Used by *assignCellToTrackUsingAll* module.

DirectionRankingOrder – Numeric array specifying the default order of shape and motility parameters for fast-moving directional objects when it cannot be determined based on prediction power. Used by *assignCellToTrackUsingAll* module.

RelevantParametersIndex – Boolean array specifying column indexes in the shape and motility matrix that have been determined to be irrelevant for tracking. This indicates to the module not to use the parameters of those columns in computing track assignment probabilities. The order of column indexes is provided in *TracksLayout* variable. Used by *assignCellToTrackUsingAll* module.

UnknownParamWeights – Numeric array specifying a set of weights to be assigned to shape and motility parameters when the prediction power of the parameters cannot be determined. Used by *assignCellToTrackUsingAll* module.

UnknownRankingOrder – Numeric array specifying the order of the shape and motility parameters when their predictive power cannot be determined. If the objects cannot be categorized as either slow-moving or fast directional the parameter order provided in this variable is used. Used by *assignCellToTrackUsingAll* module.

Important Modules

areaFilterLabel, areaOverPerimeterFilterLabel, assignCellToTrackUsingAll, clearSmallObjects, detectMergeCandidatesUsingDistance, detectMitoticEvents, distanceWatershed, generateBinImgUsingLocAvg, getConvexObjects, polygonalAssistedWatershed, removeShortTracks, segmentObjectsUsingMarkers, solidityFilterLabel, splitTracks.

assayFluoNuclTestNN

Usage

This module is used to track cells that have been stained with a nuclear stain using a nearest-neighbor algorithm. The assay tracks the cells, detects mitotic events and records tracking data, cell shape parameters and ancestry information to spreadsheets. For each tracked image, an overlayed image is saved to disk that displays the detected cell outlines, cell IDs and cell generation.

Script Variables

ImageFolder - String variable that specifies the absolute location of the directory which contains the time-lapse images. An example of such a string variable would be *'c:/sample images/high-density'*.

ImageFilesRoot - String variable specifying the root image file name. The root image file name for a set of images is the image file name of any of the images without the number or the file extension. For example, if the file name is *'Experiment-0002_Position(8)_t021.tif'* the root image file name will be *'Experiment-0002_Position(8)_t'*.

ImageExtension – String variable specifying the image file extension including the preceding dot. For example, if the file name is *'image003.jpg'* the image extension is *'.jpg'*.

StartFrame – Number specifying the first image in the sequence to be analyzed. The minimum value for this variable depends on the numbering of the image sequence, so if the first image in the sequence is 'image003.tif' then the minimum value is 3.

FrameCount – Number specifying how many images from the image sequence should be processed.

TimeFrame – Number specifying the time between consecutive images in minutes.

FrameStep - Number specifying the step size when reading images. Set this variable to 1 to read every image in the sequence, 2 to read every other image and so on.

NumberFormat – String value specifying the number of digits in the image file names in the sequence. For example, if the image file name is '*image020.jpg'* the value for the *NumberFormat* is *'%03d'*, while if the file name is '*image000020.jpg'* the value should be *'%06d'*.

MaxFramesMissing – Number specifying for how many frames a cell may disappear before its track is ended.

OutputFolder – The folder where the overlayed images and track data will be saved. By default this value is set to a folder named *'output'* within the folder where the images to be analyzed are located.

AncestryFolder – The folder where the overlayed images and ancestry data will be saved. By default this value is set to a folder named *'ancestry'* within the output folder.

AncestrySpreadsheet – The path name to the spreadsheet containing the ancestry data. By default this value is set to a file named *'ancestry.csv'* within the ancestry folder.

ShapesSpreadsheet – The path name to the spreadsheet containing the position and shape properties for each cell in the time-lapse sequence at every time point. By default this is set to to a file named *'shapes.csv'* within the ancestry folder.

TracksFolder – The folder where the label matrixes containing the cell outlines are saved. By default this value is set to a folder named *'track'* within the output folder.

SegmentationFilesRoot – The root file name of the label matrixes containing the cell outlines.

ImageFileBase – The path name to the images. This value is generated from the *ImageFolder* and the *ImageFilesRoot* and should not be changed.

BrightnessThresholdPct – Number specifying the percentage threshold value for the image generated by the *generateBinImgUsingLocAvg* filter. Any pixel in the original image smaller than the threshold value times the corresponding value in the local average image below this value will be set to zero while the rest will be set to one.

ObjectArea – Number specifying the threshold area for the areaFilterLabel, clearSmallObjects, segmentObjectsUsingClusters filters. Objects below this value will be removed from the filtered image.

Strel – String variable specifying the type of filter used to generate the local average image in *generateBinImgUsingLocAvg*. Currently *'disk'* is the only value supported.

StrelSize – Number specifying the size of the local neighborhood used to calculate the average for each pixel in the local average image generated by the *generateBinImgUsingLocAvg* module.

ClearBorder – Boolean value specifying whether objects next to or touching the image border in the binary images generated by the *generateBinImgUsingGradient* module will be erased (*true*) or not (*false*).

ClearBorderDist – Number specifying how close to the border objects may be and still be erased if the *ClearBorder* parameter is set to *true* in the *generateBinImgUsingGradient* module.

MedianFilterSize – Number specifying the size of the median filter used by the *distanceWatershed* module. Setting this to a higher integer value will reduce the number of objects detected by the module and can be used to prevent oversegmentation.

MinSolidity – Number specifying a threshold solidity value for the *solidityFilterLabelObjects* filter. Objects whose solidity is below this value will be removed from the filtered image.

MinAreaOverPerimeter – Number specifying a threshold AOP ratio value for the *areaOverPerimeterFilterLabel* filter. Objects with an AOP ratio smaller than this value will be removed.

ResizeImageScale – Number specifying by what ratio the images will be resized before they are processed. By default this value is set to 0.5. Used by the *resizeImage* module.

ApproximationDistance – Number specifying how close the convex hull in the *getConvexObjects* module approximates the object outline. By default this value is set to 2.5. Setting it to a lower value will result in convex hulls that more closely resemble the object outlines, however this increases the chance of detecting insignificant concavities.

MaxMergeDistance – Number specifying the maximum distance that one track may be from another track for the duration and still be considered for possible merging with the other track. Used by *detectMergeCandidatesUsingDistance* module.

MaxSplitArea – Number specifying the maximum area a nucleus may be and still be considered as a part of a possible mitotic event. Used by *detectMitoticEvents* module.

MaxSplitDistance – Number specifying the maximum distance a new nucleus may be from another nucleus and still be considered as part of a possible mitotic event. Used by *detectMitoticEvents* module.

MinSplitEccentricity – Number specifying the minimum eccentricity a new nucleus may have and still be considered as part of a possible mitotic event. Used by *detectMitoticEvents* module.

MaxSplitEccentricity – Number specifying the maximum eccentricity a new nucleus may have and still be considered as part of a possible mitotic event. Used by *detectMitoticEvents* module.

MinTimeForSplit – Number specifying the minimum time in minutes a track needs to exist before it is considered for a possible mitotic event. Used by *detectMitoticEvents* module.

MinLifespan – Number specifying the minimum length in frames a frame has to be to not be removed by the *removeShortTracks* module.

<u>Important Modules</u>

areaFilterLabel, areaOverPerimeterFilterLabel, assignCellToTrackUsingNN, clearSmallObjects, detectMergeCandidatesUsingDistance, detectMitoticEvents, distanceWatershed,

generateBinImgUsingLocAvg, getConvexObjects, polygonalAssistedWatershed, removeShortTracks, segmentObjectsUsingMarkers, solidityFilterLabel, splitTracks.

assayFluoNuclThresholding

Usage

This module is used to threshold a series of images (no object segmentation and no tracking are performed).

Script Variables

ImageFolder - String variable that specifies the absolute location of the directory which contains the time-lapse images. An example of such a string variable would be *'c:/sample images/high-density'*.

ImageFilesRoot - String variable specifying the root image file name. The root image file name for a set of images is the image file name of any of the images without the number or the file extension. For example, if the file name is *'Experiment-0002_Position(8)_t021.tif'* the root image file name will be *'Experiment-0002_Position(8)_t'*.

ImageExtension – String variable specifying the image file extension including the preceding dot. For example, if the file name is *'image003.jpg'* the image extension is *'.jpg'*.

StartFrame – Number specifying the first image in the sequence to be analyzed. The minimum value for this variable depends on the numbering of the image sequence, so if the first image in the sequence is 'image003.tif' then the minimum value is 3.

FrameCount – Number specifying how many images from the image sequence should be processed.

FrameStep - Number specifying the step size when reading images. Set this variable to 1 to read every image in the sequence, 2 to read every other image and so on.

NumberFormat – String value specifying the number of digits in the image file names in the sequence. For example, if the image file name is '*image020.jpg'* the value for the *NumberFormat* is *'%03d'*, while if the file name is '*image000020.jpg'* the value should be *'%06d'*.

OutputFolder – The folder where the thresholded images will be saved. By default this value is set to a folder named *'output'* within the folder where the images to be analyzed are located.

BrightnessThresholdPct – Number specifying the percentage threshold value for the image generated by the *generateBinImgUsingLocAvg* filter. Any pixel in the original image smaller than the threshold value times the corresponding value in the local average image below this value will be set to zero while the rest will be set to one.

ClearBorder – Boolean value specifying whether objects next to or touching the image border in the binary images generated by the *generateBinImgUsingGradient* module will be erased (*true*) or not (*false*).

ClearBorderDist – Number specifying how close to the border objects may be and still be erased if the *ClearBorder* parameter is set to *true* in the *generateBinImgUsingGradient* module.

ObjectArea – Number specifying the threshold area for the *clearSmallObjects, polygonalAssistedWatershed* filter. Objects below this value will be removed from the filtered image.

Strel – String variable specifying the type of filter used to generate the local average image in *generateBinImgUsingLocAvg*. Currently *'disk'* is the only value supported.

StrelSize – Number specifying the size of the local neighborhood used to calculate the average for each pixel in the local average image generated by the *generateBinImgUsingLocAvg* module.

Important Modules

generateBinImgUsingLocAvg.

assayGetStageOffset

Usage

This assay is used to calculate the microscope stage offset at each frame by manually clicking on a fixed point in every image.

Script Variables

ImageFolder - String variable that specifies the absolute location of the directory which contains the time-lapse images. An example of such a string variable would be *'c:/sample images/high-density'*.

ImageFilesRoot - String variable specifying the root image file name. The root image file name for a set of images is the image file name of any of the images without the number or the file extension. For example, if the file name is *'Experiment-0002_Position(8)_t021.tif'* the root image file name will be *'Experiment-0002_Position(8)_t'*.

ImageFileBase – The path name to the images. This value is generated from the *ImageFolder* and the *ImageFilesRoot* and should not be changed.

ImageExtension – String variable specifying the image file extension including the preceding dot. For example, if the file name is *'image003.jpg'* the image extension is *'.jpg'*.

NumberFormat – String value specifying the number of digits in the image file names in the sequence. For example, if the image file name is '*image020.jpg'* the value for the

NumberFormat is *'%03d'*, while if the file name is '*image000020.jpg'* the value should be *'%06d'*.

StartFrame – Number specifying the first image in the sequence to be analyzed. The minimum value for this variable depends on the numbering of the image sequence, so if the first image in the sequence is 'image003.tif' then the minimum value is 3.

FrameStep - Number specifying the step size when reading images. Set this variable to 1 to read every image in the sequence, 2 to read every other image and so on.

FrameCount – Number specifying how many images from the image sequence should be processed.

XYOffsets – Initial value for the stage offsets array. Set to zero by default.

Important Modules

None.

assayOffsetFrames

Usage

This module is used to offset and crop frames to remove incorrect offsets due to errors in automatic stage return. Use with offset data acquired using an assay such as assayGetStageOffset. The original images will be untouched and the cropped images will be saved in a "cropped frames" directory under the original image folder.

Script Variables

ImageFolder - String variable that specifies the absolute location of the directory which contains the time-lapse images. An example of such a string variable would be *'c:/sample images/high-density'*.

ImageFilesRoot - String variable specifying the root image file name. The root image file name for a set of images is the image file name of any of the images without the number or the file extension. For example, if the file name is *'Experiment-0002_Position(8)_t021.tif'* the root image file name will be *'Experiment-0002_Position(8)_t'*.

ImageFileBase – String variable specifying the path name to the images. This value is generated from the *ImageFolder* and the *ImageFilesRoot* and should not be changed.

CroppedImagesFolder – String variable specifying the folder where the cropped images will be saved. Set by default to a folder named *'cropped'* within the original images folder.

ImageExtension – String variable specifying the image file extension including the preceding dot. For example, if the file name is *'image003.jpg'* the image extension is *'.jpg'*.

NumberFormat – String value specifying the number of digits in the image file names in the sequence. For example, if the image file name is '*image020.jpg'* the value for the *NumberFormat* is *'%03d'*, while if the file name is '*image000020.jpg'* the value should be *'%06d'*.

StartFrame – Number specifying the first image in the sequence to be analyzed. The minimum value for this variable depends on the numbering of the image sequence, so if the first image in the sequence is 'image003.tif' then the minimum value is 3.

FrameStep - Number specifying the step size when reading images. Set this variable to 1 to read every image in the sequence, 2 to read every other image and so on.

FrameCount – Number specifying how many images from the image sequence should be processed.

Important Modules

None.

Control Modules

forLoop

Usage

This module is used to loop through a series of modules. The modules that will be looped through are listed in the *LoopFunctions* input structure member.

Input Structure Members

EndLoop – The loop will stop when the loop counter reaches this value.

IncrementLoop – The value by which the loop counter will be incremented every time one loop cycle is completed.

LoopFunctions – The list of module input structures that will be looped through.

StartLoop – The loop counter will be initialized to this value.

Any other arguments may be added to the *FunctionArgs* structure and they will be available to the modules contained in the *LoopFunctions* structure member. This allows the loop modules to have access to values generated by modules outside the loop.

Output Structure Members

Any values generated by the modules in the loop can be made available to modules outside the loop by adding them to the *KeepValues* structure.

if_statement

Usage

This module is used to create branching execution in an assay. Depending on the result of a test function, either the modules in the *IfFunctions* or the modules in the *ElseFunctions* structures will be executed.

Input Structure Members

TestVariable – The module which will be used to determine what subset of modules will be executed. This module needs to return a *true*/*false* value.

IfFunctions – Set of modules which will be executed if the value returned by the test function is *true*.

ElseFunctions – Set of modules which will be executed if the value returned by the test function is *false*.

whileLoop

Usage

This module is used to loop through a series of modules. The modules that will be looped through are listed in the *LoopFunctions* input structure member.

Input Structure Members

TestFunction – The module which will be used to determine how many times the loop will iterate over the modules in *LoopFunctions*. This module needs to return a *true*/*false* value and be part of the *whileLoop*'s module loop functions.

Internal Modules

addFunction

Usage

This module adds two variables.

Input Structure Members

Number1 – The first variable to be added.

Number2 – The second variable to be added.

Output Structure Members

Sum – The result of the addition.

areaFilterLabel

Usage

This module is used to remove objects below and/or above a certain area from a label matrix.

Input Structure Members

MaxArea – Objects with an area larger than this value will be removed.

MinArea – Objects with an area smaller than this value will be removed.

ObjectsLabel – The label matrix from which objects will be removed.

Output Structure Members

LabelMatrix – The filtered label matrix.

areaOverPerimeterFilterLabel

Usage

This module is used to remove objects below and/or above a certain area/perimeter ratio from a label matrix. Montages consisting of multiple images can exhibit noise at the points where the individual images are joined. The objects from this type of noise have a higher area/perimeter ratio than true objects and can be removed using this filter.

Input Structure Members

MaxAreaOverPerimeter – Objects with an AOP ratio larger than this value will be removed.
MinAreaOverPerimeter – Objects with an AOP ratio smaller than this value will be removed.
ObjectsLabel – The label matrix from which objects will be removed.

Output Structure Members

LabelMatrix – The filtered label matrix.

assignCellToTrackUsingAll

Usage

This module tracks objects in a time-lapse sequence of label matrices. It uses distance, speed, direction and shape parameters to determine which candidate object best matches a track. It can be optimized for tracking fast-moving directional objects or slow-moving objects.

Input Structure Members

CheckCellPath – To allow paths that go through another cell, set this value to true, otherwise set it to false.

CellsCentroids – Current object centroids.

CellsLabel – The current time step label matrix.

CurrentTracks – Matrix containing the track assignments and shape parameters for the objects in the previous frame.

DefaultParamWeights – A set of weights is assigned to each parameter based on its prediction power. Parameters with high prediction power are assigned high weights and parameters with low prediction power are assigned lower weights. If an object can be assigned to a matching group or is a sure match for a track the weights listed in this variable are used.

DirectionRankingOrder – When the order of the parameters based on prediction power cannot be determined using a group match, a set of default parameter ranks is used. The parameter order for fast directional objects is provided in this variable.

DistanceRankingOrder – When the order of the parameters based on prediction power cannot be determined using a group match, a set of default parameter ranks is used. The parameter order for slow-moving objects is provided in this variable.

ExcludedTracks – List of track assignments that should not be changed.

FrontParams – To force a set of parameters to the front so they are heavily weighted, enter their column indices in the variable; otherwise set the variable to an empty vector.

MatchingGroups – Matrix of current matching groups (vectors of shape or motility indices ordered by their prediction power).

MatchingGroupsStats – Matrix of mean values for each parameter in each group.

MaxAngleDiff – The maximum allowed angle difference between a track and a candidate object. If the angle is larger than this value, direction ranking will not be used for this object.

MaxDistRatio – The maximum allowed distance ratio between the two nearest candidate objects. If the ratio is higher than this value, distance ranking will not be used.

MaxSearchRadius - Sets an absolute lower bound for the search radius to prevent selecting too few candidate objects for a track.

MaxTrackID – Current maximum track ID.

MinSecondDistance – Minimum significant distance between the closest candidate object to a track and the second closest. Used to determine when distance should be used as a ranking parameter.

MinSearchRadius – Sets an absolute higher bound for the search radius to prevent selecting too many candidate objects for a track.

NrParamsForSureMatch – This value is used to indicate the minimum number of closest matches between a candidate object parameters and a track's object parameters that make the candidate object a sure match to the track.

ParamsCoeffOfVariation – Matrix containing the coefficient of variation for each parameter.

PreviousCellsLabel – The matrix label from the previous time step.

PreviousTracks – Matrix containing the track assignments and shape parameters for the objects in the frame previous to those in *CurrentTracks*.

RelevantParametersIndex – Shape or motility parameters that have been determined to be irrelevant for tracking the objects can be eliminated by setting the corresponding index in the variable to false. This indicates to the module not to use the parameters in computing track assignment probabilities.

SearchRadiusPct – This value determines the size of the neighborhood from which candidate objects for matching the track are selected. It is a multiple of the distance to the nearest candidate

in the current frame. Setting this variable equal to 1 turns this module into a nearest-neighbor algorithm (only the nearest cell can be a candidate). It does not make sense to have a value lower than 1.

ShapeParameters – Shape parameters (area, eccentricity, perimeter, etc.) extracted from the current label.

TrackAssignments – List of track assignments that have already been completed.

TracksLayout – Matrix describing the order of the columns in the tracks matrix.

UnassignedCells – List of object IDs currently unassigned.

UnknownParamWeights – A set of weights is assigned to each parameter based on its prediction power. Parameters with high prediction power are assigned high weights and parameters with low prediction power are assigned lower weights. If an object cannot be assigned to a matching group and is not a sure match for a track the weights listed in this variable are used.

UnknownRankingOrder – When the order of the parameters based on prediction power cannot be determined using a group match, a set of default parameter ranks is used. If the objects cannot be categorized as either slow-moving or fast directional the parameter order provided in this variable is used.

Output Structure Members

UnassignedIDs – List of object IDs currently unassigned.

TrackAssignments – List of track assignments that have already been completed.

MatchingGroups – Matrix of mean values for each parameter in each group.

GroupIndex – Indicates the index of the group to which the object has been assigned.

ExcludedTracks – List of track assignments that should not be changed.

assignCellToTrackUsingNN

Usage

This module tracks objects in a time-lapse sequence of label matrices using a nearest-neighbor algorithm.

Input Structure Members

CellsCentroids – Current object centroids.

CurrentTracks – Matrix containing the track assignments for the objects in the previous frame.

MaxTrackID – Current maximum track ID.

TrackAssignments – List of track assignments that have already been completed.

TracksLayout – Matrix describing the order of the columns in the tracks matrix.

UnassignedCells – List of object IDs currently unassigned.

Output Structure Members

ExcludedTracks – Returns empty value. Included for compatibility only.

GroupIndex – Returns zero. Included for compatibility only.

MatchingGroups – Returns empty value. Included for compatibility only.

TrackAssignments – List of track assignments that have already been completed.

UnassignedIDs – List of object IDs currently unassigned.

calculateCropSize

Usage

This module is used to calculate how much a image will need to be cropped to remove the microscope stage offsets.

Input Structure Members

Image – Image in the sequence to be cropped..

XYOffsets – Array containing the offsets for all the images in the series.

Output Structure Members

CropSize – Array indicating how much the image will need to be cropped in each direction.

clearSmallObjects

Usage

This module is used to remove objects below a certain area from a binary image.

Input Structure Members

Image – Binary image from which objects will be removed.

MinObjectArea – Objects with an area smaller than this value will be removed.

Output Structure Members

Image – Filtered binary image.

combineImages

Usage

This module is used to combine two binary images using logical operations.

Input Structure Members

CombineOperation – String value indicating the logical operation used to combine the images. Currently, only ‘AND’ and ‘OR’ are supported.

Image1 – First binary image.

Image2 – Second binary image.

Output Structure Members

Image – Binary image resulting from the logical operation.

compareValues

Usage

This module is used to compare two numerical values using the specified operation.

Input Structure Members

Arg1 – First numerical value.

Arg2 – Second numerical value.

Operation – String representing the mathematical operation to be performed. Currently, ‘>’,’<’,’>=’,’<=’ and ’==’ are supported.

Output Structure Members

BooleanOut – The result of the operation. Can be either 1 (true) or 0 (false).

continueTracks

Usage

This module is used to continue the tracks with the new track assignments as tracking progresses from frame to frame.

Input Structure Members

CurFrame – Integer value representing the current frame number.

TrackAssignments – Matrix containing the new track assignments.

TimeFrame – Integer value representing the current time frame.

Output Structure Members

NewTracks – Matrix containing the new tracks for the current frame.
Tracks – Matrix containing the tracks including the new track assignments.

cropImage

Usage

This module is used to crop an image from a point specified from the (0,0) coordinate to the specified size.

Input Structure Members

CropSize – Two number vector containing the desired dimensions for the cropped image.

Image – The image to be processed.

XYOffset – Offset point (from the (0,0) coordinate) from which the image should be cropped.

Output Structure Members

Image – Cropped image.

cropImageWithOffset

Usage

This module is used to crop an image from a point specified from the center of the image to the specified size.

Input Structure Members

CropSize – Two number vector containing the desired dimensions for the cropped image.

Image – The image to be processed.

XYOffset – Offset point (from the center of the original image) at which the cropped image should be centered.

Output Structure Members

Image – Cropped image.

detectMergeCandidatesUsingDistance

Usage

This module is used to detect tracks that are never further than a small distance apart for possible merging.

Input Structure Members

MaxMergeDistance – Maximum distance between tracks. Any tracks that drift further apart than this distance at any time point will be merged.

TrackIDs – Track IDs to be tested for merging.

Tracks – Tracks matrix including time stamps and object centroids.

TracksLayout – Matrix describing the order of the columns in the tracks matrix.

Output Structure Members

TracksToBeMerged – Paired list of the track IDs to be merged.

detectMitoticEvents

Usage

This module is used to detect mitotic events in time-lapse movies of cells stained with a nuclear stain.

Input Structure Members

MaxSplitArea – Maximum area a nucleus may have at the time it splits.

MaxSplitDistance – Nuclei that are further apart than this distance will not be considered as a potential split pair.

MaxSplitEccentricity – Any nucleus with an eccentricity above this value will not be considered a split candidate.

MinSplitEccentricity – Any nucleus with an eccentricity below this value will not be considered a split candidate.

MinTimeForSplit – A cell needs to have a lifespan above this value to be considered a split candidate.

Tracks – Tracks matrix including time stamps and object centroids.

TracksLayout – Matrix describing the order of the columns in the tracks matrix.

UntestedIDs - Track IDs to be tested for mitosis.

Output Structure Members

SplitCells – List of mitotic cell pairs.

displayAncestryData

Usage

This module is used to overlay cell outlines (using different colors to indicate different cell generations) and cell labels on the original images after tracking and save the resulting image.

Input Structure Members

AncestryLayout – Matrix describing the order of the columns in the tracks matrix.

CellsAncestry – Matrix containing the ancestry records for the cells in the image.

CellsLabel – The label matrix containing the cell outlines for the current image.

ColorMap – Color map to be used in drawing the cell outlines for each generation. Each generation will use the next color in the color map until all colors have been used. Afterwards, the colors in the map are recycled.

CurFrame – Integer containing the current frame number.

CurrentTracks – The list of the tracks for the current image.

DS – The directory separator to be used when generating file names ("\" for Windows, "/" for Unix/Linux).

Image – The original image which will be used to generate the image with overlayed outlines and labels.

ImageFileName – The root of the image file name to be used when generating the image file name for the current image in combination with the current frame number.

NumberFormat – A string indicating the number format of the file name to be used when saving the overlayed image.

ProlDir – Output directory where the resulting image will be saved.

TracksLayout – Matrix describing the order of the columns in the tracks matrix.

Output Structure Members

None.

displayImage

Usage

This module is used to display an image in the specified figure number.

Input Structure Members

FigureNr – The figure number where the image should be displayed.

Image – Matrix containing the image to be displayed.

Output Structure Members

None.

displayTracksData

Usage

This module is used to overlay cell outlines (using different colors to indicate different cell generations) and cell labels on the original images after tracking and save the resulting image.

Input Structure Members

CellsLabel – The label matrix containing the cell outlines for the current image.

CurFrame – Integer containing the current frame number.

CurrentTracks – The list of the tracks for the current image.

FileRoot – The root of the image file name to be used when generating the image file name for the current image in combination with the current frame number.

Image – The original image which will be used to generate the image with overlayed outlines and labels.

NumberFormat – A string indicating the number format of the file name to be used when saving the overlayed image.

TracksLayout – Matrix describing the order of the columns in the tracks matrix.

Output Structure Members

None.

displayVariable

Usage

This module is used to display a variable name and its value.

Input Structure Members

Variable – The variable whose value is to be displayed.

VariableName – The variable name to be displayed along with the variable value.

Output Structure Members

None.

distanceWatershed

Usage

This module is used to compute the distance watershed of a binary image.

Input Structure Members

Image – The binary image for which the distance watershed will be computed.

MedianFilterNhood – The size of the median filter which will be used to smooth the watershed.

Output Structure Members

LabelMatrix – The result of the distance watershed.

eccentricityFilterLabel

Usage

This module is used to remove objects below and/or above a certain eccentricity from a label matrix.

Input Structure Members

MaxEccentricity – Any object with an eccentricity higher than this value will be removed.

MinEccentricity – Any object with an eccentricity lower than this value will be removed.

ObjectsLabel – The label matrix from which the objects will be removed.

Output Structure Members

LabelMatrix – The filtered label matrix.

generateBinImgUsingGlobInt

Usage

This module is used to convert a grayscale image to binary using a global intensity threshold.

Input Structure Members

ClearBorder – If this value is set to *true,* objects that are within *ClearBorderDist* of the image edges will be erased.

ClearBorderDist – Objects that are within this distance from the edges of the image will be erased if *ClearBorder* is set to *true*.

Pixel intensities greater than the threshold intensity are converted to 1 in the binary image.

Image – Grayscale image to be converted.

IntensityThresholdPct – This is a percentage of the intensity range of the image. The threshold intensity value is calculated as *IntensityThresholdPct*double(max_pixel-min_pixel)+min_pixel.*

Output Structure Members

Image – Resulting binary image.

generateBinImgUsingGradient

Usage

This module is used to convert a grayscale image to a binary image using values of the image gradient.

Input Structure Members

ClearBorder – If this value is set to *true,* objects that are within *ClearBorderDist* of the image edges will be erased.

ClearBorderDist – Objects that are within this distance from the edges of the image will be erased if *ClearBorder* is set to *true*.

GradientThreshold – Areas in the gradient image where the gradient is higher than this value will be set to 1 in the binary image.

Image – Grayscale image to be converted.

Output Structure Members

Image – Resulting binary image.

generateBinImgUsingLocAvg

Usage

This module is used to convert a grayscale image to a binary image using local average values.

Input Structure Members

BrightnessThresholdPct – This value indicates what percentage of the local average value will be used to threshold the image. If the pixel intensity is higher than this value times the local average value the corresponding pixel in the binary image will be set to one.

ClearBorder – If this value is set to *true,* objects that are within *ClearBorderDist* of the image edges will be erased.

ClearBorderDist – Objects that are within this distance from the edges of the image will be erased if *ClearBorder* is set to *true*.

Image – Grayscale image to be converted.

Strel – Filter type used to generate the local average image. Currently *'circular'* is the only value supported.

StrelSize – Size of the local neighborhood used to calculate the average for each pixel in the image.

Output Structure Members

Image – Resulting binary image.

getArrayVal

Usage

This module returns a value or set of values from an array.

Input Structure Members

Array – Array from which the value is to be extracted.

Index – The index of the values to be returned.

Output Structure Members

ArrayVal – Set of values extracted from the array.

getCentroids

Usage

This module returns a list of centroids for the objects in a label matrix.

Input Structure Members

LabelMatrix – Label matrix from which the object centroids will be calculated.

Output Structure Members

Centroids – The list of centroids extracted from the label matrix.

getConvexObjects

Usage

This module is used to find and return the index of convex objects in a binary image. The object outlines are simplified using a Douglas-Pecker algorithm to prevent detection of insignificant convexities.

Input Structure Members

ApproximationDistance – This value represents the minimum distance between the approximated outline and the real one. Increasing this value makes the contours simpler but less like the original outlines.

Image – Binary image to be processed.

Output Structure Members

ConvexObjectsIndex – List containing the index of the convex objects. The index of each object is based on performing a *bwlabeln* operation on the binary image.

getCurrentTracks

Usage

Module to extract a subset of tracks out of the tracks matrix starting with the current frame.

Input Structure Members

CurFrame – The current frame index.

FrameStep – How many frames to skip when reading the track subset. If every frame is to be read set this value to 1.

MaxMissingFrames – This value indicates if tracks not present in the current frame should be included in the track subset and if so how many frames away from the current frame a track is allowed to be and still be included in the subset. Setting this value to zero ensures that only tracks present in the current frame are included.

OffsetFrame – The number of frames from the current frame the subset should include. This value can be a positive or negative integer.

TimeCol – Index of the time column in the tracks matrix.

TimeFrame – The amount of time elapsed between each frame.

TrackIDCol – Index of the track ID column in the tracks matrix.

Tracks – Matrix containing the set of tracks from which the subset is to be extracted.

Output Structure Members

Tracks – The subset of tracks extracted from the tracks matrix.

getFileInfo

Usage

This module is used to extract the file name, extension and directory from the absolute path.

Input Structure Members

DirSep – Directory separator ("\" for Windows, "/" for Linux/Unix).

PathName – Absolute path name from which file name, extension and directory will be extracted.

Output Structure Members

DirName – The extracted directory name.

FileName – The extracted file name.

ExtName – The extracted extension name.

getMaxTrackID

Usage

This module is used to return the current maximum track ID from a track matrix.

Input Structure Members

TrackIDCol – Index of the track ID column in the tracks matrix.

Tracks – Matrix containing the set of tracks.

Output Structure Members

MaxTrackID – The maximum track ID.

getObjectIntensities

Usage

This module is used to return the object intensities from objects in a label matrix using the corresponding intensity image.

Input Structure Members

LabelMatrix – The label matrix containing the objects for which the mean intensity data will be extracted.

IntensityImage – Matrix containing the intensity image.

Output Structure Members

MeanIntensities – Array containing the mean intensities for each object in the label matrix.

getObjectsMeanDisplacement

Usage

This module estimates the average displacement of the cells in the frame using the centroids from the previous and current frame.

Input Structure Members

Centroid1Col – The index of the first coordinate of the object centroids in the tracks matrix.

Centroid2Col – The index of the second coordinate of the object centroids in the tracks matrix.

CurrentTracks – Set of tracks belonging to previous frame.

ObjectCentroids – The list of centroids in the current frame.

Output Structure Members

MeanDisplacement – The estimated mean displacement of the objects in the frame.

SDDisplacement – The estimated standard deviation of the objects in the frame.

getShapeParams

Usage

This module is used to extract the shape parameters (Area, Eccentricity, etc.) of objects in a label matrix. Mostly, a wrapper for the MATLAB *regionprops* function. Adds a blob id to the output so that objects that belong to the same blob may be identified at a later time.

Input Structure Members

LabelMatrix – The label matrix from which the shape parameters will be extracted.

Output Structure Members

Centroids – The centroids of the objects in the label matrix.

ShapeParameters – The shape parameters of the objects in the label matrix.

getShapeParamsWithDisconnects

Usage

This module is used to extract the shape parameters (Area, Eccentricity, etc.) of objects in a label matrix. The difference between this module and *getShapeParams* is that *getShapeParamsWithDisconnects* supports objects which are disjointed (spread across multiple blobs) at the cost of execution speed.

Input Structure Members

LabelMatrix – The label matrix from which the shape parameters will be extracted.

Output Structure Members

Centroids – The centroids of the objects in the label matrix.

ShapeParameters – The shape parameters of the objects in the label matrix.

getTrackIDs

Usage

This module is used to retrieve the list of track IDs from the track matrix.

Input Structure Members

TrackIDCol – Index of the track ID column in the tracks matrix.

Tracks – Matrix containing the set of tracks from which IDs will be extracted.

Output Structure Members

TrackIDs – The IDs extracted from the tracks matrix.

holdValue

Usage

The only purpose for this module is to hold a value so that other modules may use it, for example by having the *holdValue* module at a higher level in the hierarchy, thereby preventing modules at lower levels from overwriting the value.

Input Structure Members

ValueToHold – The value to hold.

Output Structure Members

ValueToHold – The held value.

imNorm

Usage

This module is used to normalize an image so that the lowest pixel value is zero and the highest pixel value is the maximum allowed value for the specified integer class.

Input Structure Members

IntegerClass – The integer class of the image. Has to be an integer class supported by MATLAB such as 'int8' or 'uint8'.
RawImage – The image to be processed.

Output Structure Members

Image – The normalized image.

loadAncestryLayout

Usage

This module is used to load a structure containing the order of each column in the ancestry matrix from a MATLAB .mat file.

Input Structure Members

FileName – The name of the .mat file containing the ancestry layout structure.

Output Structure Members

AncestryLayout – Structure containing the order of each column in the ancestry matrix.

loadCellsLabel

Usage

This module is used to load cells/nuclei labels. It looks for a variable named cells_lbl in the .mat data file.

Input Structure Members

MatFileName – The name of the data file.

Output Structure Members

LabelMatrix – The label matrix loaded from the file.

loadTracksLayout

Usage

This module is used to load a structure containing the order of each column in the tracks matrix from a MATLAB .mat file.

Input Structure Members

FileName – The name of the .mat file containing the tracks layout structure.

Output Structure Members

TracksLayout – Structure containing the order of each column in the tracks matrix.

makeAncestryForCellsEnteringFrames

Usage

This module is used to add ancestry records for cells entering the field of view after the first frame (not the result of a mitotic event in the field of view).

Input Structure Members

CellsAncestry – Current cell ancestry records.

FirstFrameIDs – The IDs of tracks starting in the first frame.

SplitCells – The IDs of tracks that are the result of mitosis.

TimeCol – The index of the time column in the tracks matrix.

TrackIDCol – The index of the track ID column in the tracks matrix.

TrackIDs – The IDs of all the tracks.

Tracks – The matrix containing all the tracks.

Output Structure Members

CellsAncestry – The current cell ancestry record with ancestry of cells entering the field of view after the first frame appended to the end.

makeAncestryForFirstFrameCells

Usage

This module is used to create ancestry records for cells present in the first frame of a time-lapse movie. These cells are generation zero.

Input Structure Members

TimeCol – The index of the time column in the tracks matrix.

TrackIDCol – The index of the track ID column in the tracks matrix.

TrackIDs – The IDs of all the tracks.

Tracks – The matrix containing all the tracks.

Output Structure Members

CellsAncestry – The ancestry records for the cells in the first frame.

FirstFrameIDs – IDs of cells in the first frame.

UntestedIDs – IDs of cells in the movie that were not present in the first frame.

makeExcludedTracksList

Usage

This module wraps its input, a list of cell IDs, in a MATLAB cell array.

Input Structure Members

UnassignedCellsIDs – The list of cell IDs.

Output Structure Members

ExcludedTracks – The MATLAB cell array.

makeImgFileName

Usage

This module is used to build the filename of a frame from a time-lapse movie using the root file name, current frame number and a specified number format and file extension.

Input Structure Members

CurFrame – The index of the current frame.

FileBase – The root of the image file name.

FileExt – The file extension.

NumberFmt – The number format to be used. See MATLAB *sprintf* documentation for format documentation.

Output Structure Members

FileName – String containing the resulting file name.

makeUnassignedCellsList

Usage

This module is used to create a list of IDs for each cell centroid in a list.

Input Structure Members

CellsCentroids – List of cell centroids.

Output Structure Members

UnassignedCellsIDs – List of IDs.

manualSegmentationReview

Usage

This module is used to manually correct errors in automatic segmentation. The module loads a GUI for each frame with all the available segmentation corrections. To start the user has to select between blob correction and object correction by clicking on the *"Select Blob"* or *"Select Object"* button. A blob is a contiguous region as defined by the background pixels and it may contain one or more objects. An object is the set of pixels with the same non-zero ID in the label matrix. An object may span multiple blobs.

In general (with the exception of the *"Restore Blob"* operation), an object or blob must be selected before an operation can be performed on it. Selection is performed by clicking on the object or blob of interest. A selected item is indicated by a checkerboard pattern. If the "Select Multiple" box is checked, clicking on an unselected item adds it to the selection.

The types of operations that may be performed on a blob are resegmentation, deletion and restoration. To resegment a blob one needs to indicate how many objects the new blob will contain and their approximate boundaries. Clicking on the selected blob after the *"Resegment Blob"* button has been pressed indicates that the pixels at those locations belong to the first object in the blob. To move to the next object, press the letter *"n"* on the keyboard and click within the blob to indicate rough boundaries. The boundaries do not have to be specified precisely and a blob may be separated into two objects in as few as two clicks. Once all the objects have been defined, press the letter *"d"* on the keyboard and the blob will be resegmented. During resegmentaion all the pixels in the blob are assigned to objects using a nearest-neighbor

classifier based on the pixels selected by the user. A blob may be deleted by selecting it and then clicking the *"Remove Blob"* button. To restore a blob removed by mistake, click on the *"Restore Blob"* button. The GUI will display the *"Raw Label"* image which shows all the blobs present after thresholding before any removal by filters or manual deletions. Choose the blob to restore by clicking on it.

Two types of operations can be performed on objects: joining and deletion. To join a number of objects into a single object, click on the *"Join Objects"* button then select the objects you want to join by clicking on them. When you are done with object selection and want to join the objects, press the *"d"* letter on the keyboard. To delete an object, select it, then click on the *"Remove Object"* button.

Once all the corrections have been performed, click on the *"Save Changes & Continue"* button to save your changes and move on to the next frame.

Input Structure Members

Image – The microscopy image for the current *ObjectsLabel*.

ObjectsLabel – The label matrix containing the objects for which the automatic segmentation will be evaluated or corrected.

PreviousLabel – The label matrix containing objects from the previous time step.

RawLabel – The label matrix containing the objects before filtering.

Output Structure Members

LabelMatrix – The label matrix containing manual corrections, if any.

manualTrackingReview

Usage

The manual tracking review module can be used to correct errors in automatic tracking and as a visualization module to select subsets of tracks based on speed, motility and ancestry parameters. For track correction there are six actions that can be performed: continue a track, switch tracks, delete a track, break a track, add a split and remove a split. The GUI displays population statistics at the top under the menu bar and individual cell statistics (if a cell is selected) in a text box on the right side. Detected cell outlines and cell IDs may be displayed by checking the *"Outlines"* and/or *"Labels"* checkboxes.

Track continuation can be used when automatic tracking loses a cell it is tracking and starts a new track for that same cell. To continue the pre-existing track with the new track, select the pre-existing track. Selecting a track is done by clicking on the cell body. After selecting the pre-existing track, click on the *"Continue Track"* button, navigate to a frame where the new track exists and click on the cell body again to select it. The new track is then appended to the pre-existing track. To see the updated track navigate to another frame.

Switching tracks can be used to correct errors resulting from tracks being switched from one cell to the other during automatic tracking. To switch the tracks, navigate to the frame where the error occurs, click on the first cell, click on the *"Switch tracks"* button and finally click on the second cell. To see the updated tracks navigate to another frame.

To erase a track, select it, then click on the *"Delete Track"* button.

Breaking a track splits the track in two at the current frame. The newly created track starts at the current frame and the old track ends at the frame before that. This can be used to correct complex

errors by separating a track into smaller segments than using the other actions to fix the automatic tracking errors.

Errors in mitotic event detection can be corrected using the “Add Split” and “Remove Split” buttons. To correct a missed mitotic event click on the parent cell (the cell with the older track), click on the “Add Split” button, then click on the second cell that is part of the mitotic event. The older track is broken at the current frame, a new track is created and the ancestry records are updated for all three tracks. To remove a spurious mitotic event select the track you want to use to continue the parent track, and then click on the “Remove Split” button. The new track is merged with the parent track and the ancestry records are updated for both the parent track and the other remaining track.

The visualization part of the GUI is controlled using the “Manage Selection Layers” button. A selection layer is a transparency overlaid on the original image that highlights certain cells based on a user-defined criterion. The criterion for comparison may be an exact value (such as all cells with an area larger than 500 square pixels) or a percentage (cells with an area in the top 20%). Any combination of shape, motility and ancestry parameters may be used alone or in combination to define a layer. This allows the user to define layers that are either very broad in scope, such as all cells that are larger than average in a movie, or extremely tailored, such as selection for small, rounded, fast cells with a specific parent ID. Multiple layers may be present at one time and, due to the use of transparencies and a broad selection of layer colors, cells that are part of multiple layers can be detected. The layers are automatically updated as the user moves backward or forward through the timelapse sequence, and the resulting images themselves may be saved.

To define a selection layer, click on the “Manage Selection Layers” button, then click the “Add Layer” button. Type in the name of the layer, select a layer color from the dropdown box, then add a number of conditions. Conditions may be combined using “AND” and “OR” logical connectors. A condition consists of a property such as “Area” or “Cell ID”, an operation (“=”,”<”,”>” are supported) and a value. The value can be either an absolute number or a percentage. Once all the conditions have been set, click the “Save Layer” button, then close the selection layers GUI to apply the layer. To delete a layer, click on “Manage Selection Layers”, then select the layer to be removed and click on the “Remove Layer” button.

Input Structure Members

AncestryLayout – Matrix describing the order of the columns in the tracks matrix.
CellsAncestry – Matrix containing cell ancestry records.

ColorMap – Color map to be used in drawing the cell outlines for each generation. Each generation will use the next color in the color map until all colors have been used. Afterwards, the colors in the map are recycled.
FirstFrameIDs – The IDs of tracks starting in the first frame.

FrameCount – The number of frames to track.
FrameStep – Read one out of every x frames when reading the image set. Default value is one, meaning every frame will be read.
ImageFileBase – The root file name of the images in the sequence. For example, if the image names in the time-lapse sequence are “Experiment-0002_Position(8)_t001.jpg”,”Experiment-0002_Position(8)_t002.jpg”, etc., the root image file name is “Experiment-0002_Position(8)_t”.
ImgExt – String indicating the image file extension. Usually, “.jpg” or “.tif”.
MaxMissingFrames – This value indicates if tracks not present in the current frame should be included in the track subset and if so how many frames away from the current frame a track is allowed to be and still be included in the subset.
NumberFormat – A string indicating the number format of the file name to be used when saving the overlayed image.
SegFileRoot – The root of the data file name containing the segmented objects.
StartFrame – Integer indicating at which frame the tracking should start.
SplitCells – The IDs of tracks that are the result of mitosis.

TimeCol – The index of the time column in the tracks matrix.

TimeFrame – Time interval between consecutive frames.

TrackIDCol – The index of the track ID column in the tracks matrix.

TrackIDs – The IDs of all the tracks.

Tracks – The matrix containing all the tracks (track IDs and shape parameters for every cell at every time point).

TracksLayout – Matrix describing the order of the columns in the tracks matrix.

Output Structure Members

CellsAncestry – Matrix containing corrected cell ancestry records.

Tracks – The matrix containing all the corrected tracks (track IDs and shape parameters for every cell at every time point).

mergeTracks

Usage

This module is used to merge a list of track pairs. This module is not used to determine whether a set of tracks *should* be merged. Other modules are provided for that purpose, such as *detectMergeCandidatesUsingDistance*.

Input Structure Members

FrameCount – The number of frames that are being processed.

FrameStep – How many frames to skip when reading frames. Setting this value to one will cause all the frames to be read.

NumberFormat – A string indicating the number format of the file name to be used when saving the overlayed image.

SegFileRoot – The root of the data file name containing the segmented objects.

StartFrame – The first frame in the processed set.

TimeFrame – Time interval between consecutive frames.

Tracks – The current set of tracks.

TracksLayout – Matrix describing the order of the columns in the tracks matrix.

TracksToBeMerged – The matrix of track IDs to be merged. Primary IDs (the IDs which will remain after the merge) are listed in the first column and secondary IDs (the IDs which will be removed after the merge) are listed in the second column of the matrix.

Output Structure Members

Tracks – The new track set with the results from the merge incorporated.

negativeImage

Usage

This module returns the negative of the image provided as an argument.

Input Structure Members

Image – Image to be processed.

Output Structure Members

Image – Negative image.

overlayAncestry

Usage

This module is used to overlay the cell outlines (color-coded according to generation number) and the track ids on top of the original cell image.

Input Structure Members

AncestryLayout – Matrix describing the order of the columns in the ancestry matrix.

CurrentTracks – The set of tracks for the current image.

CellsLabel – The label matrix containing the detected cell shapes for the current image.

CellsAncestry – Matrix containing the ancestry records for the cells in the time-lapse movie.

ColorMap – Color map to be used in drawing the cell outlines for each generation. Each generation will use the next color in the color map until all colors have been used. Afterwards, the colors in the map are recycled.

Image – This is the original cell image.

ShowLabels – Boolean value. If set to false the cell IDs will not be overlayed.

ShowOutlines – Boolean value. If set to false the cell outlines will not be overlayed.

TracksLayout – Matrix describing the order of the columns in the tracks matrix.

Output Structure Members

Image – The overlayed image.

percentageForeground

Usage

This module calculates the percentage of foreground pixels in a binary image.

Input Structure Members

Image – Binary image for which the percentage of foreground pixels is to be calculated.

Output Structure Members

PercentageForeground – The percentage of foreground pixels.

polygonalAssistedWatershed

Usage

This module is used to prevent a watershed segmentation module (can be another type of segmentation module) from splitting convex objects. The assumption is that convex objects are atomic and should not be split. This assumption works well for nuclear stains.

Input Structure Members

ConvexObjectsIndex – List containing the index of convex objects. This list may be generated using the *getConvexObjects* module.

ImageLabel – Label matrix containing the original objects before segmentation.

MinBlobArea – Objects resulting from segmentation that have an area smaller than this value will be unsegmented.

WatershedLabel – Label matrix containing the objects after segmentation.

Output Structure Members

LabelMatrix – A label matrix containing the objects segmented according to the segmentation in *WatershedLabel* with the exception of convex objects, which are left unaltered.

refineSegmentation

Usage

This module is used to retain only objects in a label matrix that are nearest to objects in another matrix.

Input Structure Members

CurrentLabel – The label matrix from which objects may be removed if they don't have an object to which they are nearest in the *PreviousLabel* matrix.

PreviousLabel – The objects in this label will determine the objects that will be retained in the current label.

Output Structure Members

LabelMatrix – The filtered label matrix.

removeShortTracks

Usage

This module is used to erase tracks with a lifespan shorter than a specified period of time.

Input Structure Members

AncestryLayout – Matrix describing the order of the columns in the ancestry matrix.

Tracks – The tracks matrix to be processed.

CellsAncestry – Matrix containing the ancestry records for the cells in the time-lapse movie.

TracksLayout – Matrix describing the order of the columns in the tracks matrix.

MinLifespan – Tracks with a lifespan shorter than this value will be erased.

Output Structure Members

Tracks – The filtered tracks matrix.

saveAncestry

Usage

This module is used to save the cells ancestry matrix. The matrix is saved with the variable name *cells_ancestry*.

Input Structure Members

AncestryFileName – The file name to which the cell ancestry matrix should be saved.

CellsAncestry – The matrix containing the cells ancestry records.

Output Structure Members

None

saveAncestrySpreadsheets

Usage

This module is used to save the tracks and ancestry records spreadsheets.

Input Structure Members

CellsAncestry – Matrix containing the ancestry records for the cells in the time-lapse movie.

ProlXlsFile – The desired file name for the spreadsheet containing the ancestry records.

ShapesXlsFile – The desired file name for the spreadsheet containing the tracks and shape parameters data.

Tracks – The tracks matrix to be processed.

TracksLayout – Matrix describing the order of the columns in the tracks matrix.

Output Structure Members

None.

saveCellsLabel

Usage

This module is used to save a MATLAB label matrix containing cell objects.

Input Structure Members

CellsLabel – The label matrix containing cell objects.

CurFrame – The index of the frame to which the label matrix corresponds.

FileRoot – String containing the root of the file name to be used when saving the label matrix.

NumberFormat – String indicating the number format to be used when formatting the current frame number to be concatenated to the file root string. See the MATLAB sprintf help file for example number format strings.

Output Structure Members

CellsLabel – The label matrix containing cell objects.

saveRegionPropsSpreadsheets

Usage

This module is used to save the shape parameters extracted using the *getRegionProps* wrapper module.

Input Structure Members

RegionProps – The matrix containing the shape parameters.

SpreadsheetFileName – The desired file name for the saved file.

Output Structure Members

None.

saveTracks

Usage

This module is used to save the tracks matrix.

Input Structure Members

Tracks – Matrix containing the tracks to be saved.

TracksFileName – The desired file name for the saved tracks data.

Output Structure Members

None.

segmentObjectsUsingClusters

Usage

This module is used to segment objects in a label matrix using hierarchical clustering.

Input Structure Members

ClusterDistance – The height threshold for the cluster tree. All leaves below this value will be grouped in a cluster. See documentation for the MATLAB function *cluster* for more details.

MinimumObjectArea – Objects with an area smaller than this value will be unsegmented and distributed between the neighboring objects.

ObjectsLabel – The label matrix containing the objects to be segmented.

ObjectReduce – Used to reduce the size of the objects. If the objects are too large the clustering function will run out of memory. When this happens set *ObjectReduce* to a value lower than one.

Output Structure Members

LabelMatrix – The label matrix containing the segmented objects.

segmentObjectsUsingMarkers

Usage

This module is used to segment objects in a label matrix using markers from another label matrix.

Input Structure Members

MarkersLabel – The label matrix containing the marker objects.

ObjectsLabel – The label matrix containing the objects to be segmented.

Output Structure Members

LabelMatrix – Label matrix containing the segmented objects.

setArrayVar

Usage

This module is used to set a set of values in an array.

Input Structure Members

Array – The array where the values will be entered.

Index – The index in the array where the values will be entered.

Var – The set of values that will be entered in the array.

Output Structure Members

Array – The array with the new set of values.

showImageAndPause

Usage

This module is used to show an image and pause execution.

Input Structure Members

FigureNr – The handle number of the MATLAB figure. If it doesn't exist it will be created.

Image – Matrix containing the image to be shown.

Output Structure Members

None.

showLabelMatrixAndPause

Usage

This module is used to show a MATLAB label matrix and pause execution.

Input Structure Members

FigureNr – The handle number of the MATLAB figure. If it doesn't exist it will be created.

LabelMatrix – The label matrix to be displayed.

Output Structure Members

None.

solidityFilter

Usage

This module is used to remove objects below or above a threshold solidity from a binary image.

Input Structure Members

Image – The binary image from which objects will be removed.

MaxSolidity – Objects whose solidity is above this value will be removed from the image.

MinSolidity - Objects whose solidity is below this value will be removed from the image.

Output Structure Members

Image – The filtered binary image.

solidityFilterLabel

Usage

This module is used to remove objects below or above a threshold solidity value from a MATLAB label matrix.

Input Structure Members

ObjectsLabel – The label matrix from which objects will be removed.

MaxSolidity – Objects whose solidity is above this value will be removed from the image.

MinSolidity - Objects whose solidity is below this value will be removed from the image.

Output Structure Members

LabelMatrix – The filtered label matrix.

startTracks

Usage

This module is used to start a tracks matrix.

Input Structure Members

CellsLabel – The label matrix identifying the objects in the first frame of the time-lapse.

CurFrame – The index value of the current frame.

ShapeParameters – The shape parameters for the objects in the first frame (Area, Eccentricity, etc.).

TimeFrame – The time interval between consecutive frames in the time-lapse.

Output Structure Members

Tracks – The new tracks matrix.

subtractFunction

Usage

This module subtracts the first variable from the second one.

Input Structure Members

Number1 – The first variable.

Number2 – The variable to be subtracted.

Output Structure Members

Difference – The result of the subtraction.

Appendix 2: CellAnimation Source Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% addCondition.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function addCondition()
%helper function for the manual tracking review GUI. used to add conditions
%for a selection layer
global al_gui_struct;

cell_properties=al_gui_struct.CellProperties;
logic_connectors=al_gui_struct.LogicConnectors;
operators=al_gui_struct.Operators;

condition_struct.ComboLogicConnector=logic_connectors{get(al_gui_struct.ComboLogicConnectorHandle
,'Value')};
condition_struct.ComboCellProperty=cell_properties{get(al_gui_struct.ComboCellPropertyHandle,'Val
ue')};
condition_struct.ComboOperator=operators{get(al_gui_struct.ComboOperatorHandle,'Value')};
condition_struct.EditValue=get(al_gui_struct.EditValueHandle,'String');
if (isempty(condition_struct.EditValue))
    warndlg('The value cannot be empty.');
    return;
end
status_text=get(al_gui_struct.TextConditionsHandle,'String');
if isempty(status_text)
    status_text=[condition_struct.ComboCellProperty condition_struct.ComboOperator...
        num2str(condition_struct.EditValue)];
    condition_struct.ComboLogicConnector='AND';
    al_gui_struct.Conditions=condition_struct;
else
    status_text=[status_text ' ' condition_struct.ComboLogicConnector ' '
condition_struct.ComboCellProperty...
        condition_struct.ComboOperator num2str(condition_struct.EditValue)];
    al_gui_struct.Conditions=[al_gui_struct.Conditions; condition_struct];
end
set(al_gui_struct.TextConditionsHandle,'String',status_text);

%end addCondition
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% addFunction.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=addFunction(input_args)
%Usage
%This module adds two variables.
%
%Input Structure Members
%Number1 - The first variable to be added.
%Number2 - The second variable to be added.
%
%Output Structure Members
%Sum - The result of the addition.
%
%Example
%
%get_previous_frame_nr_function.InstanceName='GetPreviousFrameNr';
%get_previous_frame_nr_function.FunctionHandle=@addFunction;
%get_previous_frame_nr_function.FunctionArgs.Number1.FunctionInstance='Segment
%ationLoop';
%get_previous_frame_nr_function.FunctionArgs.Number1.OutputArg='LoopCounter';
%get_previous_frame_nr_function.FunctionArgs.Number2.Value=-1;
%image_read_loop_functions=addToFunctionChain(image_read_loop_functions,get_pr
%evious_frame_nr_function);
%
%make_mat_name_function.FunctionArgs.CurFrame.FunctionInstance='GetPreviousFra
%meNr';
%make_mat_name_function.FunctionArgs.CurFrame.OutputArg='Sum';

arg_1=input_args.Number1.Value;
arg_2=input_args.Number2.Value;
output_args.Sum=(arg_1+arg_2);

%end addFunction
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% addLayer.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function addLayer()
%helper function for manual tracking review GUI. Used to start the add
%layer GUI and save the new selection layer
global sl_gui_struct;
global al_gui_struct;
al_gui_struct=[];
al_gui_struct.NewSelectionLayer=[];

gui_handle=addLayerGUI;
al_gui_struct.GUIHandle=gui_handle;
children_handles=get(gui_handle,'children');
al_gui_struct.EditSelectionLayerNameHandle=findobj(children_handles,'tag','editSelectionLayerName
');
al_gui_struct.ComboColorHandle=findobj(children_handles,'tag','comboColor');
al_gui_struct.ComboLogicConnectorHandle=findobj(children_handles,'tag','comboLogicConnector');
al_gui_struct.ComboCellPropertyHandle=findobj(children_handles,'tag','comboCellProperty');
al_gui_struct.ComboOperatorHandle=findobj(children_handles,'tag','comboOperator');
al_gui_struct.EditValueHandle=findobj(children_handles,'tag','editValue');
al_gui_struct.TextConditionsHandle=findobj(children_handles,'tag','textConditions');
cell_properties=...
    {'Area';'Cell ID';'Generation';'Eccentricity';'End Frame';'Speed';'Start Frame';'Parent
ID';'Perimeter';'RMS';'Solidity'};
set(al_gui_struct.ComboCellPropertyHandle,'String',cell_properties);
al_gui_struct.CellProperties=cell_properties;
logic_connectors={'AND';'OR'};
set(al_gui_struct.ComboLogicConnectorHandle,'String',logic_connectors);
al_gui_struct.LogicConnectors=logic_connectors;
operators={'=';'>';'<'};
set(al_gui_struct.ComboOperatorHandle,'String',operators);
al_gui_struct.Operators=operators;
layer_colors={'Aquamarine';'Black';'Blue';'Dark Brown';'Dark Green';'Lime
Green';'Grey';'Orange';'Pink';'Purple';'Red';'Sienna';'Turqoise';'Violet';'Yellow'};
set(al_gui_struct.ComboColorHandle,'String',layer_colors);
al_gui_struct.LayerColors=layer_colors;
selection_names=sl_gui_struct.SelectionNames;
al_gui_struct.SelectionNames=selection_names;
waitfor(gui_handle);

new_selection_layer=al_gui_struct.NewSelectionLayer;
clear al_gui_struct;
if (~isempty(new_selection_layer))
    selection_names=[selection_names {new_selection_layer.Name}];
    sl_gui_struct.SelectionNames=selection_names;
    sl_gui_struct.SelectionLayers=[sl_gui_struct.SelectionLayers; {new_selection_layer}];
    set(sl_gui_struct.ListboxSelectionLayersHandle,'String',selection_names);
end


%end addLayer
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% addLayerGUI.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function varargout = addLayerGUI(varargin)
% ADDLAYERGUI M-file for addLayerGUI.fig
%      ADDLAYERGUI, by itself, creates a new ADDLAYERGUI or raises the existing
%      singleton*.
%
%      H = ADDLAYERGUI returns the handle to a new ADDLAYERGUI or the handle to
%      the existing singleton*.
%
%      ADDLAYERGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ADDLAYERGUI.M with the given input arguments.
%
%      ADDLAYERGUI('Property','Value',...) creates a new ADDLAYERGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before addLayerGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to addLayerGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
```

```
% Edit the above text to modify the response to help addLayerGUI

% Last Modified by GUIDE v2.5 23-Jul-2010 23:10:37

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @addLayerGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @addLayerGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before addLayerGUI is made visible.
function addLayerGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to addLayerGUI (see VARARGIN)

% Choose default command line output for addLayerGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes addLayerGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = addLayerGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in comboLogicConnector.
function comboLogicConnector_Callback(hObject, eventdata, handles)
% hObject    handle to comboLogicConnector (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns comboLogicConnector contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from comboLogicConnector


% --- Executes during object creation, after setting all properties.
function comboLogicConnector_CreateFcn(hObject, eventdata, handles)
% hObject    handle to comboLogicConnector (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in comboCellProperty.
function comboCellProperty_Callback(hObject, eventdata, handles)
% hObject    handle to comboCellProperty (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns comboCellProperty contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from comboCellProperty


% --- Executes during object creation, after setting all properties.
function comboCellProperty_CreateFcn(hObject, eventdata, handles)
% hObject    handle to comboCellProperty (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in comboOperator.
function comboOperator_Callback(hObject, eventdata, handles)
% hObject    handle to comboOperator (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns comboOperator contents as cell array
%        contents{get(hObject,'Value')} returns selected item from comboOperator


% --- Executes during object creation, after setting all properties.
function comboOperator_CreateFcn(hObject, eventdata, handles)
% hObject    handle to comboOperator (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editValue_Callback(hObject, eventdata, handles)
% hObject    handle to editValue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editValue as text
%        str2double(get(hObject,'String')) returns contents of editValue as a double


% --- Executes during object creation, after setting all properties.
function editValue_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editValue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editSelectionLayerName_Callback(hObject, eventdata, handles)
% hObject    handle to editSelectionLayerName (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editSelectionLayerName as text
%        str2double(get(hObject,'String')) returns contents of editSelectionLayerName as a double


% --- Executes during object creation, after setting all properties.
function editSelectionLayerName_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editSelectionLayerName (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in buttonSaveLayer.
function buttonSaveLayer_Callback(hObject, eventdata, handles)
% hObject    handle to buttonSaveLayer (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
saveLayer();


% --- Executes on button press in buttonCancel.
function buttonCancel_Callback(hObject, eventdata, handles)
% hObject    handle to buttonCancel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
addLayerCancel();


% --- Executes on button press in buttonAddCondition.
function buttonAddCondition_Callback(hObject, eventdata, handles)
% hObject    handle to buttonAddCondition (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
addCondition();


% --- Executes on selection change in comboColor.
function comboColor_Callback(hObject, eventdata, handles)
% hObject    handle to comboColor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns comboColor contents as cell array
%        contents{get(hObject,'Value')} returns selected item from comboColor


% --- Executes during object creation, after setting all properties.
function comboColor_CreateFcn(hObject, eventdata, handles)
% hObject    handle to comboColor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% addManualValue.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function addManualValue(handles)
%helper function for assayEditorGUI. add a manual value to the currently selected input argument
assay_list=get(handles.listboxInputArgumens,'String');
selection_idx=get(handles.listboxInputArgumens,'Value');
selection_text=assay_list{selection_idx};
if (length(selection_text)>9)&&strcmp(selection_text(1:9),'<html><i>')
    warndlg('You need to select an argument name (not in italics)');
    return;
end
if (length(selection_text)>6&&strcmp(selection_text(1:6),'<html>'))
    %remove the html formatting
    selection_text=selection_text(25:(end-14));
    %display the selection text without the red warning format
    assay_list{selection_idx}=selection_text;
end
output_arg(1)={selection_text};
module_struct=handles.ModuleStruct;
manual_value=get(handles.editManualValue,'String');
output_arg(2)={manual_value};
i=selection_idx+1;
j=i;
```

```
list_len=length(assay_list);
while 1
    if (j>list_len)
        break;
    end
    selection_text=assay_list{j};
    if (length(selection_text)<21)
        break;
    end
    if strcmp(selection_text(1:21),'<html><i> Output')
        j=j+1;
        continue;
    end
    if ~strcmp(selection_text(1:20),'<html><i> Value')
        break;
    end
    j=j+1;
    i=i+1;
end
module_struct.StaticParameters=[module_struct.StaticParameters {output_arg}];
value_arg_string=['<html><i> Value' num2str(i-selection_idx) '</i></html>'];
assay_list=[assay_list(1:(j-1));value_arg_string;assay_list(j:end)];
set(handles.listboxInputArgumens,'String',assay_list);
handles.ModuleStruct=module_struct;
guidata(handles.figure1,handles);

%end addManualValue
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% addModuleToAssay.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function addModuleToAssay(handles)
%helper function for assayEditorGUI. add a module to the current assay
assay_list=get(handles.listboxCurrentAssay,'String');
selection_idx=get(handles.listboxCurrentAssay,'Value');

available_modules=get(handles.listboxAvailableModules,'String');
cur_idx=get(handles.listboxAvailableModules,'Value');
module_name=available_modules{cur_idx};
module_name=module_name(1:end-2);
insertModule(module_name,assay_list,selection_idx,handles);

%end addModuleToAssay
end

function insertModule(module_name,assay_list,selection_idx,handles)
%insert the new module in the modules list
module_struct.ModuleName=module_name;
if isempty(assay_list)
    parents_list={''};
    chains_list={'functions_list'};
    cur_module=[];
else
    [parents_list chains_list
cur_module]=getPossibleParentModules(assay_list,selection_idx,handles);
end
[is_ok parent_idx module_instance]=moduleParentGUI('ParentsList',parents_list);
if (~is_ok)
    return;
end
modules_list=handles.ModulesList;
existing_instances=cellfun(@(x) x.InstanceName,modules_list,'UniformOutput',false);
if max(strcmp(module_instance,existing_instances))
    %this instance name is taken
    warndlg('This instance name is already taken! Please use another name.');
    return;
end
module_struct.InstanceName=module_instance;
switch module_name
    case {'forLoop','whileLoop'}
        module_struct.IsParent=true;
        module_struct.ChainVars={'LoopFunctions'};
        module_struct.Chains{1}=[module_instance module_struct.ChainVars{1}];
    case 'if_statement'
        module_struct.IsParent=true;
        module_struct.ChainVars={'ElseFunctions','IfFunctions'};
        module_struct.Chains{1}=lower([module_instance module_struct.ChainVars{1}]);
        module_struct.Chains{2}=lower([module_instance module_struct.ChainVars{2}]);
    otherwise
```

```
        module_struct.IsParent=false;
        module_struct.ChainVars={};
        module_struct.Chains={};
end
module_struct.Parent=parents_list{parent_idx};
module_struct.ChainName=chains_list{parent_idx};
%get the static args
module_struct.StaticParameters={};
module_struct.OutputArgs={};
module_struct.InputArgs={};
module_struct.KeepOutputArgs={};
modules_map=handles.ModulesMap;
%figure out the level of the new module
if isempty(module_struct.Parent)
    module_struct.Level=1;
else
    parent_idx=modules_map.get(module_struct.Parent);
    parent_struct=modules_list{parent_idx};
    module_struct.Level=parent_struct.Level+1;
end
%add the new module to the modules list
if isempty(cur_module)
    %special case - empty list
    insert_idx=1;
    modules_list={module_struct};
elseif (selection_idx==length(assay_list))
    %special case - at the end of the list
    if (module_struct.Level==1)
        %add the module at the end of the modules list
        modules_list=[modules_list; module_struct];
        insert_idx=length(modules_list);
    else
        %add the module right before its parent
        insert_idx=modules_map.get(module_struct.Parent);
        modules_list=[modules_list(1:(insert_idx-1)); module_struct;
modules_list(insert_idx:end)];
    end
elseif isempty(cur_module.Chains)||(module_struct.Level==cur_module.Level)
    %regular module or adding at same level as control module - add the new
    %module right after
    cur_module_idx=modules_map.get(cur_module.InstanceName);
    modules_list=[modules_list(1:cur_module_idx); module_struct;
modules_list((cur_module_idx+1):end)];
    insert_idx=cur_module_idx+1;
else
    %control module - add the new module before the first sub-module
    submodules_idx=find(cellfun(@(x) strcmp(x.ChainName,module_struct.ChainName),modules_list));
    min_idx=min(submodules_idx);
    if isempty(min_idx)
        %no submodule-add before parent
        modules_list=[modules_list; module_struct];
        insert_idx=length(modules_list);
    else
        insert_idx=modules_map.get(module_struct.Parent);
        modules_list=[modules_list(1:(insert_idx-1)); module_struct;
modules_list(insert_idx:end)];
    end
end

modules_set=modules_map.entrySet;
set_iter=modules_set.iterator;
while (set_iter.hasNext())
    map_entry=set_iter.next();
    cur_val=double(map_entry.getValue());
    if (cur_val>=insert_idx)
        map_entry.setValue(cur_val+1);
    end
end
%update the module map
modules_map.put(module_struct.InstanceName,insert_idx);

%update the handles struct
handles.ModulesList=modules_list;
handles.ModulesMap=modules_map;
guidata(handles.figure1, handles);
module_text=formatModuleItem(module_struct);
%add the new module instance to the dialog
if isempty(cur_module)
    %special case - empty list
    assay_list=module_text;
    insert_idx=0;
```

```
elseif (selection_idx==length(assay_list))
    %special case - at the end of the list
    assay_list=[assay_list; module_text];
    insert_idx=selection_idx;
elseif isempty(cur_module.Chains)||(module_struct.Level==cur_module.Level)
    %regular module or adding at same level as control module - add the new module right after
    if (cur_module.IsParent)&&(selection_idx~=length(assay_list))
        %add the new module past the chains
        selection_text=assay_list{selection_idx+1};
        while(strcmp(selection_text(1:9),'<html><i>'))
            selection_idx=selection_idx+1;
            if selection_idx>=length(assay_list)
                break;
            end
            selection_text=assay_list{selection_idx+1};
        end
    end
    assay_list=[assay_list(1:selection_idx); module_text; assay_list((selection_idx+1):end)];
    insert_idx=selection_idx;
else
    %control module - add the new module below the appropriate chain
    chain_name=module_struct.ChainName;
    %find the chain var
    chain_idx=strcmp(chain_name,parent_struct.Chains);
    chain_var=parent_struct.ChainVars(chain_idx);
    selection_text=assay_list{selection_idx};
    module_level=module_struct.Level;
    if strcmp(selection_text(1:9),'<html><i>')
        insert_idx=selection_idx;
    else
        %find the insert index
        i=selection_idx+1;
        while 1
            selection_text=assay_list{i};
            if
(strcmp(selection_text(1:9),'<html><i>')&&((getSelectionLevel(selection_text)+1)==module_level))
                chain_name=regexp(selection_text,'<html><i>(?: )*(\w*)<','tokens','once');
                if strfind(chain_var{1},chain_name{1})
                    insert_idx=i;
                    break;
                end
            end
            i=i+1;
        end
    end
    [is_expanded selection_level]=isChainExpanded(assay_list,insert_idx);
    if (~is_expanded)&&(selection_level==(module_level-1))

assay_list=expandText(assay_list,insert_idx,modules_list,modules_map,parent_struct.Level);
    else
        assay_list=[assay_list(1:insert_idx); module_text; assay_list((insert_idx+1):end)];
    end
end

set(handles.listboxCurrentAssay,'String',assay_list);
set(handles.listboxCurrentAssay,'Value',insert_idx+1);

%end insertModule
end

function [parents_list chains_list
cur_module]=getPossibleParentModules(assay_list,selection_idx,handles)
%get a list of possible parent modules for the module about to be inserted
%along with the corresponding chain variable name - if a chain variable is selected return that,
for control modules return the first chain var
%,for everything else return the chain they're attached to. also return the
%module structure for the current module
list_len=length(assay_list);
level_1=getSelectionLevel(assay_list{selection_idx});
[cur_module is_chain chain_name]=getModuleStruct(assay_list,selection_idx,handles);
if (is_chain)
    if(selection_idx==list_len)
        parents_list{1}=cur_module.InstanceName;
        chains_list{1}=chain_name;
    else
        parents_list{1}=cur_module.InstanceName;
        chains_list{1}=chain_name;
    end
else
    parents_list{1}=cur_module.Parent;
    chains_list{1}=chain_name;
```

```
end
if (selection_idx==list_len)
    if ~isempty(parents_list{1})
        %if the selection is at the end of the list we may wish to add the
        %module to one of several levels. get the parents for all the levels
        min_level=1;
        max_level=level_1;
        modules_list=handles.ModulesList;
        modules_map=handles.ModulesMap;
        parent_module=cur_module;
        for i=2:(max_level-min_level+1)
            if ((i==2)&&(~is_chain))
                %parent for current module has been added already
                continue;
            end
            parent_id=parent_module.Parent;
            parents_list{i-min_level+1}=parent_id;
            chains_list{i-min_level+1}=parent_module.ChainName;
            parent_idx=modules_map.get(parent_id);
            parent_module=modules_list{parent_idx};
        end
        %add the main function chain
        parents_list=[parents_list {''}];
        chains_list=[chains_list 'functions_list'];
    end
else
    level_2=getSelectionLevel(assay_list{selection_idx+1});
    if (level_1>level_2)
        [module_struct is_chain]=getModuleStruct(assay_list,selection_idx+1,handles);
        chains_list{2}=chain_name;
        if is_chain
            parents_list=[parents_list {module_struct.InstanceName}];
        else
            parents_list=[parents_list {module_struct.Parent}];
        end
    end
end



%end getPossibleParentModules
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% addOutputArgument.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function addOutputArgument(handles)
%helper function for assayEditorGUI. add an output argument provider to the currently selected
input argument
assay_list=get(handles.listboxInputArgumens,'String');
selection_idx=get(handles.listboxInputArgumens,'Value');
selection_text=assay_list{selection_idx};
if (length(selection_text)>9)&&strcmp(selection_text(1:9),'<html><i>')
    warndlg('You need to select an argument name (not in italics)');
    return;
end
if (length(selection_text)>6&&strcmp(selection_text(1:6),'<html>'))
    %remove the html formatting
    selection_text=selection_text(25:(end-14));
    %display the selection text without the red warning format
    assay_list{selection_idx}=selection_text;
end
output_arg(1)={selection_text};
module_struct=handles.ModuleStruct;
module_idx=get(handles.popupModuleInstance,'Value');
popup_list=get(handles.popupModuleInstance,'String');
module_instance=popup_list{module_idx};
output_arg(2)={module_instance};
module_idx=get(handles.popupOutputArgument,'Value');
popup_list=get(handles.popupOutputArgument,'String');
module_output=popup_list{module_idx};
output_arg(3)={['''' module_output '''']};
i=selection_idx+1;
list_len=length(assay_list);
while 1
    if (i>list_len)
        break;
    end
    selection_text=assay_list{i};
    if (length(selection_text)<21)
```

```
        break;
    end
    if ~strcmp(selection_text(1:21),'<html><i> Output')
        break;
    end
    i=i+1;
end
module_struct.OutputArgs=[module_struct.OutputArgs {output_arg}];
%get the number of output arguments that belong to this input arg
cur_arg_idx=cellfun(@(x) strcmp(x{1},output_arg(1)), module_struct.OutputArgs);
output_arg_string=['<html><i> Output' num2str(sum(cur_arg_idx)) '</i></html>'];
assay_list=[assay_list(1:i-1);output_arg_string;assay_list(i:end)];
set(handles.listboxInputArgumens,'String',assay_list);
handles.ModuleStruct=module_struct;
guidata(handles.figure1,handles);

%end addOutputArgument
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% addScriptVars.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function vars_text=addScriptVars(handles)
%helper function for assayEditorGUI. create a text file defining the script variables

vars_text=['%script variables' 10];
script_vars=handles.ScriptVariables;
vars_def=[];
for i=1:length(script_vars)
    cur_var=script_vars{i};
    vars_def=[vars_def cur_var{1} '=' cur_var{2} ';' 10];
end
vars_text=[vars_text vars_def '%end script variables' 10];

%end addScriptVars
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% addSegmentationError.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function addSegmentationError(error_type, original_blob_id)
%helper function for the manual segmentation review GUI. Used to add a
%segmentation error to the list of segmentation errors.
global msr_gui_struct;

error_blob_ids=msr_gui_struct.ErrorBlobIDs;
for i=1:length(original_blob_id)
    cur_blob_id=original_blob_id(i);
    other_errors_idx=(error_blob_ids==cur_blob_id);
    other_errors_nr=sum(other_errors_idx);
    if (other_errors_nr)
        if (strcmp(error_type,'BlobThresholding'))
            %we are deleting this blob so remove all other errors associated with
            %it
            msr_gui_struct.TotalErrors=msr_gui_struct.TotalErrors-other_errors_nr+1;
            error_types=msr_gui_struct.ErrorTypes;
            error_types(other_errors_idx)=[];
            msr_gui_struct.ErrorTypes=[error_types; {error_type}];
            error_blob_ids(other_errors_idx)=[];
            msr_gui_struct.ErrorBlobIDs=[error_blob_ids; original_blob_id];
        end
        if (strcmp(error_type,'Undersegmentation')||strcmp(error_type,'Oversegmentation')||...
                strcmp(error_type,'Distribution'))
            error_types=msr_gui_struct.ErrorTypes;
            error_types{other_errors_idx}=error_type;
            msr_gui_struct.ErrorTypes=error_types;
        end
    else
        msr_gui_struct.TotalErrors=msr_gui_struct.TotalErrors+1;
        msr_gui_struct.ErrorTypes=[msr_gui_struct.ErrorTypes; {error_type}];
        msr_gui_struct.ErrorBlobIDs=[error_blob_ids; cur_blob_id];
    end
end

%end addSegmentationError
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% addSelectionLayers.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function addSelectionLayers()
%helper function for the manual tracking review module. Used to overlay
%selection layers over the original image
global mtr_gui_struct;

selection_layers=mtr_gui_struct.SelectionLayers;
layers_nr=length(selection_layers);
if (layers_nr==0)
    return;
end
layer_transparency=1./(layers_nr+1);
ancestry_records=mtr_gui_struct.CellsAncestry;
ancestry_layout=mtr_gui_struct.AncestryLayout;
%sort frame ancestries and tracks so we can use ismember to pick instead of
%loops
frame_tracks=mtr_gui_struct.FrameTracks;
tracks_layout=mtr_gui_struct.TracksLayout;
[dummy sort_idx]=sort(frame_tracks(:,tracks_layout.TrackIDCol));
mtr_gui_struct.FrameTracks=frame_tracks(sort_idx,:);
%get the ancestry records for the cells in the frame
cell_ids=frame_tracks(:,tracks_layout.TrackIDCol);
%this works because both are sorted and unique
frame_ancestries_idx=ismember(ancestry_records(:,ancestry_layout.TrackIDCol),cell_ids);
frame_ancestries=ancestry_records(frame_ancestries_idx,:);
[dummy sort_idx]=sort(frame_ancestries(:,ancestry_layout.TrackIDCol));
mtr_gui_struct.FrameAncestries=frame_ancestries(sort_idx,:);
cell_speeds=mtr_gui_struct.CellSpeeds;
cur_speeds_idx=cell_speeds(:,3)==((mtr_gui_struct.CurFrame-1).*mtr_gui_struct.TimeFrame);
[dummy sort_idx]=sort(cell_speeds(cur_speeds_idx,1));
frame_speeds=cell_speeds(cur_speeds_idx,:);
mtr_gui_struct.FrameSpeeds=frame_speeds(sort_idx,2);
cell_sq_disps=mtr_gui_struct.CellSquareDisplacements;
cur_sq_disps_idx=cell_sq_disps(:,3)==((mtr_gui_struct.CurFrame-1).*mtr_gui_struct.TimeFrame);
[dummy sort_idx]=sort(cell_sq_disps(cur_sq_disps_idx,1));
frame_sq_disps=cell_sq_disps(cur_sq_disps_idx,:);
mtr_gui_struct.FrameSquareDisplacements=frame_sq_disps(sort_idx,2);
existing_red_color=[];
existing_green_color=[];
existing_blue_color=[];

for i=1:layers_nr
    [red_color green_color blue_color]=addSelectionLayer(selection_layers{i},...
        existing_red_color,existing_green_color,existing_blue_color);
    existing_red_color=red_color;
    existing_green_color=green_color;
    existing_blue_color=blue_color;
end

hold off;
mtr_gui_struct.ImageHandle=image(mtr_gui_struct.ImageData,'Parent',mtr_gui_struct.TracksHandle);
hold on;
mtr_gui_struct.ImageHandle=image(cat(3,red_color,green_color,blue_color),'Parent',mtr_gui_struct.
TracksHandle);
set(mtr_gui_struct.ImageHandle,'AlphaData',0.5);
%set the function handle for a mouse click in the objects image
set(mtr_gui_struct.ImageHandle,'buttondownfcn','mouseClickInTrackingFrame');
hold off;

%end addSelectionLayers
end

function [red_color green_color blue_color]=addSelectionLayer(selection_layer,...
    existing_red_color,existing_green_color,existing_blue_color)
global mtr_gui_struct;

selected_ids=getCellsInSelectionLayer(selection_layer);
label_ids=getSelectedCellsLabelIDs(selected_ids);
cells_lbl=mtr_gui_struct.CellsLabel;
lbl_sz=size(cells_lbl);
selected_cells_mask=ismember(cells_lbl,label_ids);
max_pxl=intmax('uint8');
rgb_triple=getRGBTriple(selection_layer.Color);
if isempty(existing_red_color)
    red_color=(max_pxl.*rgb_triple(1).*uint8(selected_cells_mask));
else
    red_color_mask=existing_red_color>0;
    red_color=zeros(lbl_sz,'uint8');
    intersect_mask=red_color_mask & selected_cells_mask;
```

```
    new_red_color=(max_pxl.*rgb_triple(1).*uint8(selected_cells_mask));

red_color(~intersect_mask)=existing_red_color(~intersect_mask)+new_red_color(~intersect_mask);

red_color(intersect_mask)=existing_red_color(intersect_mask)./2+new_red_color(intersect_mask)./2;
end
if isempty(existing_green_color)
    green_color=(max_pxl.*rgb_triple(2).*uint8(selected_cells_mask));
else
    green_color_mask=existing_green_color>0;
    green_color=zeros(lbl_sz,'uint8');
    intersect_mask=green_color_mask & selected_cells_mask;
    new_green_color=(max_pxl.*rgb_triple(2).*uint8(selected_cells_mask));

green_color(~intersect_mask)=existing_green_color(~intersect_mask)+new_green_color(~intersect_mas
k);

green_color(intersect_mask)=existing_green_color(intersect_mask)./2+new_green_color(intersect_mas
k)./2;
end
if isempty(existing_blue_color)
    blue_color=(max_pxl.*rgb_triple(3).*uint8(selected_cells_mask));
else
    blue_color_mask=existing_blue_color>0;
    blue_color=zeros(lbl_sz,'uint8');
    intersect_mask=blue_color_mask & selected_cells_mask;
    new_blue_color=(max_pxl.*rgb_triple(3).*uint8(selected_cells_mask));

blue_color(~intersect_mask)=existing_blue_color(~intersect_mask)+new_blue_color(~intersect_mask);

blue_color(intersect_mask)=existing_blue_color(intersect_mask)./2+new_blue_color(intersect_mask).
/2;
end

%end addSelectionLayer
end

function rgb_triple=getRGBTriple(color_string)

switch color_string
    case 'Aquamarine'
        rgb_triple=[0.4980;1.0000;0.8314];
    case 'Black'
        rgb_triple=[0.0;0.0;0.0];
    case 'Dark Brown'
        rgb_triple=[0.3608;0.2510;0.2000];
    case 'Blue'
        rgb_triple=[0.0;0.0;1.0];
    case 'Dark Green'
        rgb_triple=[0.0;0.3922;0];
    case 'Lime Green'
        rgb_triple=[0.1961;0.8039;0.1961];
    case 'Grey'
        rgb_triple=[0.3294;0.3294;0.3294];
    case 'Orange'
        rgb_triple=[1.0000;0.6471;0.0];
    case 'Pink'
        rgb_triple=[1.0000;0.7529;0.7961];
    case 'Purple'
        rgb_triple=[0.6275;0.1255;0.9412];
    case 'Red'
        rgb_triple=[1.0;0.0;0.0];
    case 'Sienna'
        rgb_triple=[0.6275;0.3216;0.1765];
    case 'Turqoise'
        rgb_triple=[0.2510;0.8784;0.8157];
    case 'Violet'
        rgb_triple=[0.9333;0.5098;0.9333];
    case 'Yellow'
        rgb_triple=[1.0;1.0;0.0];
end

%end getRGBTriple
end

function selected_ids=getCellsInSelectionLayer(selection_layer)
global mtr_gui_struct;

layer_conditions=selection_layer.Conditions;
frame_tracks=mtr_gui_struct.FrameTracks;
frame_ancestries=mtr_gui_struct.FrameAncestries;
```

```
frame_speeds=mtr_gui_struct.FrameSpeeds;
frame_sq_disps=mtr_gui_struct.FrameSquareDisplacements;
tracks_layout=mtr_gui_struct.TracksLayout;
ancestry_layout=mtr_gui_struct.AncestryLayout;
cell_ids=frame_tracks(:,tracks_layout.TrackIDCol);
selection_idx=true(length(cell_ids),1);

for i=1:length(layer_conditions)
    cur_condition=layer_conditions(i);
    switch(cur_condition.ComboCellProperty)
        case 'Area'
            property_vals=frame_tracks(:,tracks_layout.AreaCol);
        case 'Cell ID'
            property_vals=cell_ids;
        case 'Generation'
            property_vals=frame_ancestries(:,ancestry_layout.GenerationCol);
        case 'Eccentricity'
            property_vals=frame_tracks(:,tracks_layout.EccCol);
        case 'End Frame'
            property_vals=frame_ancestries(:,ancestry_layout.StopTimeCol);
        case 'Speed'
            property_vals=frame_speeds;
        case 'Start Frame'
            property_vals=frame_ancestries(:,ancestry_layout.StartTimeCol);
        case 'Parent ID'
            property_vals=frame_ancestries(:,ancestry_layout.ParentIDCol);
        case 'Perimeter'
            property_vals=frame_tracks(:,tracks_layout.PerCol);
        case 'RMS'
            property_vals=frame_sq_disps;
        case 'Solidity'
            property_vals=frame_tracks(:,tracks_layout.SolCol);
    end

    edit_val=cur_condition.EditValue;
    if(edit_val(end)=='%')
        b_pct=true;
        [dummy sort_idx]=sort(property_vals);
        nr_of_vals=length(property_vals);
        select_pct=str2num(edit_val(1:(end-1)))/100;
        new_selection_idx=false(nr_of_vals,1);
    else
        b_pct=false;
        threshold_val=str2double(edit_val);
    end

    switch cur_condition.ComboOperator
        case '='
            if (b_pct)
                selection_idx=ceil(select_pct.*nr_of_vals);
                new_selection_idx(sort_idx(selection_idx))=true;
            else
                new_selection_idx=property_vals==threshold_val;
                selected_ids=cell_ids(selection_idx);
            end
        case '>'
            if (b_pct)
                nr_selected_vals=ceil((1-select_pct).*nr_of_vals);
                new_selection_idx(sort_idx((end-nr_selected_vals+1):end))=true;
            else
                new_selection_idx=property_vals>threshold_val;
            end
        case '<'
            if (b_pct)
                nr_selected_vals=ceil((1-select_pct).*nr_of_vals);
                new_selection_idx(sort_idx(1:nr_selected_vals))=true;
            else
                new_selection_idx=property_vals<threshold_val;
            end
    end
    switch cur_condition.ComboLogicConnector
        case 'AND'
            selection_idx=selection_idx & new_selection_idx;
        case 'OR'
            selection_idx=selection_idx | new_selection_idx;
    end
end
selected_ids=cell_ids(selection_idx);

%end getSelectionID
end
```

```
function label_ids=getSelectedCellsLabelIDs(selected_ids)
global mtr_gui_struct;

if (isempty(selected_ids))
    label_ids=[];
    return;
end

tracks_layout=mtr_gui_struct.TracksLayout;
frame_tracks=mtr_gui_struct.FrameTracks;
selected_cells_nr=length(selected_ids);
label_ids=zeros(selected_cells_nr,1);

for i=1:length(selected_ids)
    cur_track_record=frame_tracks(frame_tracks(:,tracks_layout.TrackIDCol)==selected_ids(i),:);
    cell_centroid=cur_track_record(:,tracks_layout.Centroid1Col:tracks_layout.Centroid2Col);
    cur_centroids=mtr_gui_struct.CurCentroids;
    cur_dist=hypot(cell_centroid(1)-cur_centroids(:,1),cell_centroid(2)-cur_centroids(:,2));
    [dummy label_ids(i)]=min(cur_dist);
end

%end getSelectedCellsLabelIDs
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%% addSplit.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function addSplit()
%helper function for the manual tracking review module. used to initiate a
%track split
global mtr_gui_struct;

mtr_gui_struct.TrackToSplitID=mtr_gui_struct.SelectedCellID;
mtr_gui_struct.TrackToSplitRecord=mtr_gui_struct.CurrentTrackRecord;
mtr_gui_struct.TrackToSplitAncestry=mtr_gui_struct.CurrentAncestryRecord;
mtr_gui_struct.SplitTrack=true;

%end addSplit
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%% addToFunctionChain.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function function_chain=addToFunctionChain(function_chain,new_function)
%CellAnimation core function. Used to add a module to the function chain

function_chain=[function_chain;{new_function}];

%end addToFunctionChain
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%% addToMatchingGroups.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ranking_order matching_groups
group_idx]=addToMatchingGroups(matching_groups,cur_shape_params,...
    nearby_params,params_coeff_var,best_fit_idx,min_reliable_params,
track_ranks,relevant_params_idx,...
    max_angle_diff,min_second_distance,max_dist_ratio,front_params)
%helper function for the CA tracking module. used to create a new matching
%group if one is not found that matches the specific combination of
%parameters
%we need to rank parameters by how near they are to their former values
%then once a ranking order is determined assign it to a matching_group. if
%none exists with the same ranking order create a new one. the first two
%parameters (distance and direction) are treated differently in that they
%are either first rank or last rank predictors.
best_fit_params=nearby_params(best_fit_idx,:);
matched_params=[cur_shape_params; best_fit_params];
%calculate the sd of the matched params and compare it with the sd of the
%nearby_params. parameters for which the sd of the matched params is
%greater than or equal to the sd of nearby_params are unreliable for
%ranking
sd_best_fit_params=std(matched_params(:,3:end));
min_params=min(matched_params);
max_params=max(matched_params);
pct_diff=1-min_params./max_params; %this way i only get values from 0-100%
```

```
smallest_pct_change=pct_diff(3:end)./params_coeff_var;
[dummy ranking_order]=sort(smallest_pct_change);
ranking_order=ranking_order+2;

if (size(nearby_params,1)==1)
    reliable_params_col=[1 2 3 4 5 6 7 8 9];
    ranking_order=[1 2 ranking_order];
else
    sd_nearby_params=std(nearby_params(:,3:end));
    reliable_params_col=find(sd_best_fit_params<sd_nearby_params)+2;
    %determine how reliable distance and direction are
    [dummy closest_distance_idx]=min(track_ranks(:,1));
    [dummy closest_angle_idx]=min(track_ranks(:,2));
    if (closest_angle_idx==closest_distance_idx)
        %both nearest distance and nearest angle point to the same cell
        %both distance and angle are reliable for this cell
        reliable_params_col=[1 2 reliable_params_col];
        ranking_order=[1 2 ranking_order];
    else
        %nearest distance and angle point to different cells
        %use the other parameters to pick which one is right
        angle_score=sum(track_ranks(closest_angle_idx,:));
        dist_score=sum(track_ranks(closest_distance_idx,:));
        if (abs(angle_score-dist_score)>2)
            %we have a clear favorite
            if (angle_score<dist_score)
                %direction is most significant
                reliable_params_col=[2 reliable_params_col];
                ranking_order=[2 ranking_order 1];
            else
                %distance is most significant
                reliable_params_col=[1 reliable_params_col 2];
                ranking_order=[1 ranking_order 2];
            end
        else
            angle_diffs_sorted=sort(nearby_params(:,2));
            distances_sorted=sort(nearby_params(:,1));
            dist_ratio=distances_sorted(1)/distances_sorted(2);
            if ((angle_diffs_sorted(1)<max_angle_diff)&&(abs(angle_diffs_sorted(1)-
angle_diffs_sorted(2))>max_angle_diff))
                %direction is most significant
                reliable_params_col=[2 reliable_params_col];
                ranking_order=[2 ranking_order 1];
            elseif ((distances_sorted(2)>min_second_distance)&&(dist_ratio<max_dist_ratio))
                %distance is most significant
                reliable_params_col=[1 reliable_params_col 2];
                ranking_order=[1 ranking_order 2];
            else
                %can't determine which is more reliable use both with
                %slight pref for distance
                reliable_params_col=[1 2 reliable_params_col];
                ranking_order=[1 2 ranking_order];
            end
        end
    end
end

reliable_params_idx=ismember(ranking_order,reliable_params_col);
%put the reliable params first
ranking_order=[ranking_order(reliable_params_idx) ranking_order(~reliable_params_idx)];
%remove parameters that have been set as irrelevant by user
irrelevant_params_col=find(~relevant_params_idx);
irrelevant_params_idx=ismember(ranking_order,irrelevant_params_col);
ranking_order(irrelevant_params_idx)=[];
%add them back at the end
ranking_order=[ranking_order irrelevant_params_col];

%if we've been asked to push any parameters to the front push them
if (~isempty(front_params))
    front_params_idx=ismember(ranking_order,front_params);
    ranking_order(front_params_idx)=[];
    ranking_order=[front_params ranking_order];
end

if (length(reliable_params_col)<min_reliable_params)
    %not enough reliable params to create a matching group
    group_idx=0;
    return;
end
nr_groups=size(matching_groups,1);
if (nr_groups==0)
```

```
    matching_groups=ranking_order;
    group_idx=1;
    return;
end
ranking_diff=matching_groups-repmat(ranking_order,nr_groups,1);
ranking_fit=sum(abs(ranking_diff),2);
group_idx=find(ranking_fit==0);
if (isempty(group_idx))
    %doesn't match any of the existing groups - create a new one
    matching_groups=[matching_groups; ranking_order];
    group_idx=nr_groups+1;
end

%end addToMatchingGroups
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% aeMenuNewAssay.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function aeMenuNewAssay(hObject, eventdata, handles)
%helper function for assayEditorGUI. used to implement the "New Assay".
handles.CurrentAssay='';
handles.AssayDescription='';
handles.ScriptVariables={};
handles.ModulesList={};
handles.ModulesMap=java.util.HashMap;
set(handles.figure1,'Name','CellAnimation Assay Editor - Untitled');
set(handles.listboxCurrentAssay,'String','');
set(handles.listboxCurrentAssay,'Value',1);
guidata(handles.figure1,handles);

%end aeMenuNewAssay
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% aeMenuOpenAssay.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function aeMenuOpenAssay(hObject, eventdata, handles)
%helper function for assayEditorGUI. used to implement the "Open Assay"
%functionality
[dlg_ok, selected_assay]=openAssayGUI();
if (~dlg_ok)
    return;
end

handles.CurrentAssay=selected_assay;
handles.AssayPath=pwd;
handles.AssayDescription=getModuleDescription(selected_assay);
%read the script variables
handles.ScriptVariables=getScriptVariables(selected_assay);
[modules_list modules_map]=buildModulesList(selected_assay);
new_modules_list=cellfun(@(x) traceModuleArgs(x,modules_list,modules_map),
modules_list,'UniformOutput',false);
handles.ModulesList=new_modules_list;
handles.ModulesMap=modules_map;
guidata(hObject,handles);
%get level 1 modules
level1_idx=cellfun(@(x) x.Level==1,new_modules_list);
modules_list=new_modules_list(level1_idx);
module_strings=formatModuleStrings(modules_list);
set(handles.listboxCurrentAssay,'String',module_strings);
set(handles.listboxCurrentAssay,'Value',1);
set(handles.figure1,'Name',['CellAnimation Assay Editor - ' selected_assay(1:(end-2))]);

%end aeMenuOpenAssay
end

function [modules_list modules_map]=buildModulesList(selected_assay)
%build a list of structures containing all the modules in sequential order
%read the assay text
file_text=fileread(selected_assay);
%remove the comments
file_text=regexp(file_text,'\n\r\%[^\n\r]*[\n\r]*','split');
file_text=strcat(file_text{:});

%get the names of every module instance and their corresponding function
%chain
module_tokens=regexp(file_text,'addToFunctionChain\(\s*(\w+),\s*(\w+)\s*\)','tokens');
chains_list=cellfun(@(x) x{1},module_tokens,'UniformOutput',false);
```

```
chains_list=unique(chains_list);
modules_nr=length(module_tokens);
modules_list=cell(modules_nr,1);
main_chain='functions_list';
prev_chain=main_chain; %main chain is called functions_list
chains_map=java.util.HashMap;
parents_map=java.util.HashMap;
modules_map=java.util.HashMap;
cur_level=1;
parent_instance='';
chains_map.put(prev_chain,cur_level);
parents_map.put(prev_chain,parent_instance);

for i=1:length(module_tokens)
    cur_tokens=module_tokens{i};
    cur_chain=cur_tokens{1};
    module_var=cur_tokens{2};
    if ~strcmp(prev_chain,cur_chain)
        prev_chain=cur_chain;
        cur_level=chains_map.get(cur_chain);
        if isempty(cur_level)
            cur_level=getChainLevel(file_text,cur_chain,main_chain);
            %get the parent module of the current chain
            parent_var=regexp(file_text,['(\w+)\.\w+=\s*' cur_chain],'once','tokens');
            parent_instance=regexp(file_text,['[;\n\r]' parent_var{1}
'.InstanceName=.(\w*).;'],'once','tokens');
            parent_instance=parent_instance{1};
            chains_map.put(cur_chain,cur_level);
            parents_map.put(cur_chain,parent_instance);
        else
            parent_instance=parents_map.get(cur_chain);
        end
    end

modules_list{i}=extractModule(file_text,module_var,parent_instance,cur_level,cur_chain,modules_li
st,chains_list);
    modules_map.put(modules_list{i}.InstanceName,i);
end

%end buildModulesList
    end

function ml=getChainLevel(file_text,sub_chain,main_chain)
%get the level of the current chain
cur_chain=sub_chain;
ml=1;
while ~strcmp(cur_chain,main_chain)
    ml=ml+1;
    %get the parent module of the current chain
    parent_var=regexp(file_text,['(\w+)\.\w+=\s*' cur_chain],'once','tokens');
    %get the chain of the parent module
    cur_chain=regexp(file_text,['addToFunctionChain\(\s*(\w*)\s*,\s*' parent_var{1}
'\s*\)'],'once','tokens');
    cur_chain=cur_chain{1};
end

%end getChainLevel
end

function
module_struct=extractModule(file_text,module_var,parent_instance,module_level,module_chain,module
s_list,chains)
%extract module struct from module assay
%build module structure
module_struct.VarName=module_var;
module_struct.Parent=parent_instance;
module_struct.Level=module_level;
module_struct.ChainName=module_chain;
%get the module section
module_lines=regexp(file_text,['\<' module_var '[^\n\r]*[\n\r]*;'],'match');
module_text=strcat(module_lines{:});
%find the instance name
search_pattern=[module_var '.InstanceName=.(\w*).;'];
instance_name=regexp(module_text,search_pattern,'once','tokens');
module_struct.InstanceName=instance_name{1};
%find the module name
search_pattern=[module_var '.FunctionHandle *= *@(\w*);'];
module_name=regexp(module_text,search_pattern,'once','tokens');
module_struct.ModuleName=module_name{1};
%is this module a control module
is_parent_idx=strcmp(module_struct.ModuleName,{'if_statement','whileLoop','forLoop'});
```

```
module_struct.IsParent=max(is_parent_idx);
if (module_struct.IsParent)
    %get the chains
    switch(module_struct.ModuleName)
        case {'forLoop','whileLoop'}
            module_struct.ChainVars={'LoopFunctions'};
            search_pattern=[module_var '.LoopFunctions=(\w*);'];
            chains=regexp(file_text,search_pattern,'tokens','once');
            module_struct.Chains=chains;
        case 'if_statement'
            module_struct.ChainVars={'ElseFunctions','IfFunctions'};
            search_pattern=[module_var '.ElseFunctions=(\w*);'];
            chains=regexp(file_text,search_pattern,'tokens','once');
            if isempty(chains)
                module_struct.Chains{1}=lower([module_struct.InstanceName
module_struct.ChainVars{1}]);
            else
                module_struct.Chains(1)=chains;
            end
            search_pattern=[module_var '.IfFunctions=(\w*);'];
            chains=regexp(file_text,search_pattern,'tokens','once');
            if isempty(chains)
                module_struct.Chains{2}=[module_struct.InstanceName module_struct.ChainVars{2}];
            else
                module_struct.Chains(2)=chains;
            end
    end
else
    module_struct.Chains={};
    module_struct.ChainVars={};
end
%get the static args
search_pattern=[module_var '.FunctionArgs.(\w*).Value *= *([^;]+);'];
module_struct.StaticParameters=regexp(module_text,search_pattern,'tokens');
%get the output args
search_pattern=[module_var '.FunctionArgs.(\w*).FunctionInstance\d* *= *.(\w*).;' module_var
'.FunctionArgs.\w*.OutputArg\d* *= *([^;]+);'];
module_struct.OutputArgs=regexp(module_text,search_pattern,'tokens');
%get the input args
search_pattern=[module_var '.FunctionArgs.(\w*).FunctionInstance\d* *= *.(\w*).;' module_var
'.FunctionArgs.\w*.InputArg\d* *= *([^;]+);'];
module_struct.InputArgs=regexp(module_text,search_pattern,'tokens');
%get the output keep values
search_pattern=[module_var '.KeepValues.(\w*).FunctionInstance\d* *= *.(\w*).;' module_var
'.KeepValues.\w*.OutputArg\d* *= *([^;]+);'];
module_struct.KeepOutputArgs=regexp(module_text,search_pattern,'tokens');

%end extractModule
end

function ip=isParent(module_struct,instance_name)
if isempty(module_struct)
    ip=false;
else
    ip=strcmp(module_struct.Parent,instance_name);
end
%end moduleHasParent
end

function arg_belongs=argBelongsToControlModule(arg_name,control_module)
%test if this argument belongs to a control
%module or if it's just being held for other modules
switch(control_module.ModuleName)
    case 'forLoop'
        internal_args={'EndLoop','IncrementLoop','StartLoop'};
    case 'whileLoop'
        internal_args={'TestFunction'};
    case 'if_statement'
        internal_args={'TestVariable'};
end

arg_belongs=max(strcmp(internal_args,arg_name{1}));
%end
end

function new_struct=traceModuleArgs(module_struct, modules_list, modules_map)
%trace any arguments provided by a control module to the module that provide the actual
%output args
input_args=module_struct.InputArgs;
output_args=module_struct.OutputArgs;
new_output_args={};
```

```
static_args={};
for i=1:length(output_args)
    %most output args by control modules don't belong to them
    cur_arg=output_args{i};
    if (module_struct.IsParent&&(~argBelongsToControlModule(cur_arg,module_struct)))
        %this is an intermediary argument. it will be traced later from its
        %proper module
        continue;
    end
    arg_idx=modules_map.get(cur_arg{2});
    arg_struct=modules_list{arg_idx};
    if (arg_struct.IsParent)
        arg_name=cur_arg{3};
        arg_name=arg_name(2:(end-1));
        if strcmp(arg_name,'LoopCounter')
            %only existing true output arg of a control module
            if strcmp(arg_struct.ModuleName,'forLoop')
                %and this is a for loop so true output - skip this one
                new_output_args=[new_output_args {cur_arg}];
                continue;
            end
        end
        new_output_args=[new_output_args traceOutputArg(cur_arg{1}, cur_arg, arg_struct,
modules_list, modules_map)];
    else
        new_output_args=[new_output_args {cur_arg}];
    end
end

for i=1:length(input_args)
    cur_arg=input_args{i};
    if (module_struct.IsParent&&(~argBelongsToControlModule(cur_arg,module_struct)))
        %controle modules shouldn't modify arguments that don't really
        %belong to them
        continue;
    end
   [arg_structs arg_types]=traceInputArg(cur_arg, modules_list, modules_map);
   for j=1:length(arg_types)
       %replace the arg name with the original input name
       arg_structs{j}{1}=cur_arg{1};
       cur_type=arg_types{j};
       switch cur_type
           case 'output'
               new_output_args=[new_output_args arg_structs(j)];
           case 'static'
               static_args=[static_args arg_structs(j)];
       end
   end
end
module_struct.InputArgs={};
new_struct=module_struct;
new_struct.OutputArgs=new_output_args;
new_struct.StaticParameters=[module_struct.StaticParameters static_args];
%what output args to keep will be determined dynamically when the assay is
%saved
new_struct.KeepOutputArgs={};

%end traceModuleArgs
end

function output_args=traceOutputArg(arg_name, output_arg, arg_struct, modules_list, modules_map)
%trace an output module in a control module to its provider(s)
cur_arg=output_arg;
output_args={};
while arg_struct.IsParent
    new_args=arg_struct.KeepOutputArgs;
    new_arg_name=cur_arg{3};
    new_arg_name=new_arg_name(2:(end-1));
    new_arg_idx=cellfun(@(x) strcmp(x{1},new_arg_name),new_args);
    cur_args=new_args(new_arg_idx);
    if (length(cur_args)>1)
        for i=1:length(cur_args)
            cur_arg=cur_args{i};
            cur_arg_idx=modules_map.get(cur_arg{2});
            arg_struct=modules_list{cur_arg_idx};
            output_args=[output_args traceOutputArg(arg_name, cur_arg, arg_struct, modules_list,
modules_map)];
        end
        return;
    else
        cur_arg=cur_args{1};
```

```
    end
    cur_arg_idx=modules_map.get(cur_arg{2});
    arg_struct=modules_list{cur_arg_idx};
end

new_arg={{arg_name cur_arg{2} cur_arg{3}}};
output_args=[output_args new_arg];

%end traceOutputArg
end

function [arg_structs arg_types]=traceInputArg(input_arg, modules_list, modules_map)
%trace an input arg to the module(s) which provides the output value(s)
module_instance=input_arg{2};
module_idx=modules_map.get(module_instance);
module_struct=modules_list{module_idx};
module_name=module_struct.ModuleName;
%if the module is not a control module it is the module which provides the
%output value
switch module_name
    case {'forLoop','if_statement','whileLoop'}
        [arg_structs arg_types]=findArgStruct(input_arg, module_struct, modules_list,
modules_map);
    otherwise
        assert(false);
end
%end traceArg
end

function [arg_structs arg_types]=findArgStruct(input_arg, module_struct, modules_list,
modules_map)
%find which structure holds the reference to this input argument could be
%multiple
input_struct=module_struct.InputArgs;
arg_name=input_arg{3};
arg_name=arg_name(2:(end-1));
arg_structs={};
arg_types={};
if (~isempty(input_struct))
    struct_names=cellfun(@(x) x{1},input_struct,'UniformOutput',false);
    arg_idx=strcmp(arg_name,struct_names);
    if (max(arg_idx)==1)
        %this struct is not where the values come from originally
        [arg_structs arg_types]=traceInputArg(input_struct{arg_idx}, modules_list, modules_map);
    end
end

static_struct=module_struct.StaticParameters;
arg_name=input_arg{3};
arg_name=arg_name(2:(end-1));
if (~isempty(static_struct))
    struct_names=cellfun(@(x) x{1},static_struct,'UniformOutput',false);
    arg_idx=strcmp(arg_name,struct_names);
    if (max(arg_idx)==1)
        %this struct lists the values come from originally
        static_args=static_struct(arg_idx);
        %change the input name to the name expected by the destination
        %module
        nr_matches=sum(arg_idx);
        for i=1:nr_matches
            cur_arg=static_args{i};
            arg_structs=[arg_structs; {{input_arg{1} cur_arg{2}}}];
            arg_types=[arg_types; {'static'}];
        end
    end
end

output_struct=module_struct.OutputArgs;
arg_name=input_arg{3};
arg_name=arg_name(2:(end-1));
if (~isempty(output_struct))
    struct_names=cellfun(@(x) x{1},output_struct,'UniformOutput',false);
    arg_idx=strcmp(arg_name,struct_names);
    if (max(arg_idx)==1)
        %this struct lists the values come from originally
        output_args=output_struct(arg_idx);
        %change the input name to the name expected by the destination
        %module
        nr_matches=sum(arg_idx);
        for i=1:nr_matches
            cur_arg=output_args{i};
```

```
            arg_structs=[arg_structs; {{input_arg{1} cur_arg{2} cur_arg{3}}}];
            arg_types=[arg_types; {'output'}];
        end
    end
end

%end findArgStruct
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% areaFilterLabel.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=areaFilterLabel(input_args)
%Usage
%This module is used to remove objects below and/or above a certain area from a label matrix.
%
%Input Structure Members
%MaxArea - Objects with an area larger than this value will be removed.
%MinArea - Objects with an area smaller than this value will be removed.
%ObjectsLabel - The label matrix from which objects will be removed.
%
%Output Structure Members
%LabelMatrix - The filtered label matrix.

cells_lbl=input_args.ObjectsLabel.Value;
cells_props=regionprops(cells_lbl,'Area');
field_names=fieldnames(input_args);
if (max(strcmp(field_names,'MinArea')))
    b_min=true;
else
    b_min=false;
end
if (max(strcmp(field_names,'MaxArea')))
    b_max=true;
else
    b_max=false;
end
cells_area=[cells_props.Area];
cells_nr=length(cells_area);
valid_areas_idx=true(1,cells_nr);
if (b_min)
    valid_areas_idx=valid_areas_idx&(cells_area>=input_args.MinArea.Value);
end
if (b_max)
    valid_areas_idx=valid_areas_idx&(cells_area<=input_args.MaxArea.Value);
end
if (min(valid_areas_idx)==1)
    %no invalid objects return the same label back
    output_args.LabelMatrix=cells_lbl;
else
    valid_object_numbers=find(valid_areas_idx);
    new_object_numbers=1:length(valid_object_numbers);
    %we will replace valid numbers with new and everything else will be set to
    %zero
    object_idx=cells_lbl>0;
    new_object_index=zeros(max(cells_lbl(object_idx)),1);
    new_object_index(valid_object_numbers)=new_object_numbers;
    new_cells_lbl=cells_lbl;
    %replace the old object numbers to prevent skips in numbering
    object_idx=cells_lbl>0;
    new_cells_lbl(object_idx)=new_object_index(cells_lbl(object_idx));
    output_args.LabelMatrix=new_cells_lbl;
end

%end areaFilterLabel
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% areaOverPerimeterFilterLabel.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=areaOverPerimeterFilterLabel(input_args)
%Usage
%This module is used to remove objects below and/or above a certain area/perimeter ratio from a
%label matrix. Montages consisting of multiple images can exhibit noise at the points where the
%individual images are joined. The objects from this type of noise have higher area/perimeter
%ratio than true objects and can be removed using this filter.
%
%Input Structure Members
```

```
%MaxAreaOverPerimeter - Objects with an AOP ratio larger than this value will be removed.
%MinAreaOverPerimeter - Objects with an AOP ratio smaller than this value will be removed.
%ObjectsLabel - The label matrix from which objects will be removed.
%
%Output Structure Members
%LabelMatrix - The filtered label matrix.

cells_lbl=input_args.ObjectsLabel.Value;
cells_props=regionprops(cells_lbl,'Area','Perimeter');
field_names=fieldnames(input_args);
if (max(strcmp(field_names,'MinAreaOverPerimeter')))
    b_min=true;
else
    b_min=false;
end
if (max(strcmp(field_names,'MaxAreaOverPerimeter')))
    b_max=true;
else
    b_max=false;
end
cells_area=[cells_props.Area];
cells_perimeter=[cells_props.Perimeter];
cells_nr=length(cells_area);
a_over_p=cells_area./cells_perimeter;
valid_aoverp_idx=false(1,cells_nr);
if (b_min)
    valid_aoverp_idx=valid_aoverp_idx|(a_over_p>=input_args.MinAreaOverPerimeter.Value);
end
if (b_max)
    valid_aoverp_idx=valid_aoverp_idx|(a_over_p<=input_args.MaxAreaOverPerimeter.Value);
end
if (min(valid_aoverp_idx)==1)
    %no invalid objects return the same label back
    output_args.LabelMatrix=cells_lbl;
else
    valid_object_numbers=find(valid_aoverp_idx);
    new_object_numbers=1:length(valid_object_numbers);
    %we will replace valid numbers with new and everything else will be set to
    %zero
    object_idx=cells_lbl>0;
    new_object_index=zeros(max(cells_lbl(object_idx)),1);
    new_object_index(valid_object_numbers)=new_object_numbers;
    new_cells_lbl=cells_lbl;
    %replace the old object numbers to prevent skips in numbering
    new_cells_lbl(object_idx)=new_object_index(cells_lbl(object_idx));
    output_args.LabelMatrix=new_cells_lbl;
end

%end areaOverPerimeterFilterLabel
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayBrightFieldCytoCA.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayBrightFieldCytoCA()
%assayFluoNuclTestCA - This module is used to track cells that have been stained using a
%         nuclear stain  using a
%custom  CellAnimation  algorithm.  This  algorithm  is  more
%accurate   and   flexible than  the   nearest-neighbor algorithm
% at the   expense of   execution speed.   The
%assay   tracks  the  cells, detects    mitotic events
% and  records tracking   data,  cell shape   parameters
% and ancestry   information  to spreadsheets.    For
%each tracked image,   an   overlayed image is   saved
%  to disk that  displays the     detected cell
% outlines, cell    IDs and cell   generation. ImageFolder
%  - String variable  that specifies the     absolute
%location of  the directory which     contains  the
% time-lapse images. An   example  of  such a
%string variable would   be  'c:/sample  images/high-density'. ImageFilesRoot
%- String variable   specifying the root  image file
% name. The root   image file name   for
%   a set of images  is the image  file name
%      of any of the   images without
% the number   or     the file extension. For
% example, if the   file name
% is 'Experiment-0002_Position(8)_t021.tif' the root  image file name will   be 'Experiment-
0002_Position(8)_t'.
%  ImageExtension      - String variable specifying  the
```

```
%image file extension  including the preceding    dot.
%  For example  if  the  file name is  'image003.jpg'
%the image   extension is       '.jpg'. StartFrame
%- Number  specifying  the first image  in  the sequence
% to be  analyzed.     The minimum   value
% for  this variable depends  on  the numbering
%of the  image    sequence so if   the
% first  image in the  sequence  is 'image003.tif'
% then the  minimum value   is 3. FrameCount  - Number
%   specifying  how many images  from  the image
%    sequence should be processed. TimeFrame   - Number specifying
% the time     between  consecutive images in  minutes.
% FrameStep -    Number   specifying the step size when
%  reading images. Set   this variable     to
% 1  to read every   image in    the
%  sequence, 2 to read every    other image and
%  so    on. NumberFormat   - String value specifying
%the   number of    digits    in the
%image file names   in  the sequence.    For example
%   if the   image file name is  'image020.jpg'
% the value for       the NumberFormat  is
%'%03d', while   if the file name    is 'image000020.jpg'
%   the value  should  be '%06d'. MaxFramesMissing  -
%  Number specifying for   how    many frames
%a  cell may  be disappear before its     track
%is      ended. OutputFolder  - The folder where the
% overlayed    images and track data   will
% be saved.   By   default  this value is set
%to   a folder  named      'output' within
% the folder where  the   images    to be
%analyzed are  located. AncestryFolder     - The   folder
% where the  overlayed images and ancestry  data   will be
% saved.    By   default   this value is
%set to   a    folder named  'ancestry' within
%the output    folder. AncestrySpreadsheet -   The   path
%name  to the spreadsheet containing  the   ancestry data.
%By    default this  value   is set to
%a file   named    'ancestry.csv' within  the ancestry
% folder. ShapesSpreadsheet  -  The path name   to
%the  spreadsheet  containing the position and  shape   properties for
%      each cell in the timelapse   sequence
%at every   time point.   By    default this
% is  set   to to  a  file named 'shapes.csv'
%  within    the  ancestry  folder. TracksFolder - The
% folder where  the label     matrixes   containing
%the cell outlines are   saved.  By    default this
%  value is   set to   a folder named
% 'track' within  the  output folder. SegmentationFilesRoot    -
%  The  root file name of the label   matrixes containing
% the cell     outlines.   ImageFileBase  - The
%path name  to the  images. This   value is
% generated from   the ImageFolder    and the
% ImageFilesRoot and should not  be changed. BrightnessThresholdPct    -
%   Number  specifying the percentage threshold value for   the
%image  generated by the     generateBinImgUsingLocAvg
%filter. Any  pixel in  the original  image smaller   than
%the    threshold value times   the    corresponding
%value in     the local average image  below this value
% will  be      set  to  zero
%while the rest   will  be  set to one.
%  ObjectArea -  Number   specifying the  threshold area
%for the  clearSmallObjects, polygonalAssistedWatershed   and  areaFilterLabel    filters.
%Objects   below this value   will  be removed
%  from the filtered image.  Strel - String   variable
%specifying    the  type  of filter used  to generate
%the   local  average   image in
% generateBinImgUsingLocAvg. Currently 'disk' is   the only  value supported.
%StrelSize -  Number specifying   the  size    of
% the local neighborhood used to calculate   the  average
%    for each pixel in  the local average
%image generated  by    the  generateBinImgUsingLocAvg  module. ClearBorder -
% Boolean value  specifying   whether  objects  next to or
%   touching   the image border in    the
% binary images    generated by  the generateBinImgUsingLocAvg   module
% will  be  erased   (true) or not  (false). ClearBorderDist
%  -  Number specifying       how close
%to the border objects may be    and still  be
%    erased  if the ClearBorder   parameter is
%set  to true  in  the generateBinImgUsingLocAvg module.
%MedianFilterSize   - Number specifying the   size of   the
%median filter   used by  the distanceWatershed   module.
```

```
%  Setting  this   to a higher integer value will
% reduce the  number of       objects detected
%  by the module and can    be used  to
%prevent    oversegmentation.  MinSolidity  - Number specifying   a
%threshold   solidity  value for  the  solidityFilterLabelObjects filter. Objects
% whose   solidity    is below this   value
%will   be  removed from   the filtered  image. MinAreaOverPerimeter
%-  Number    specifying  a  threshold   AOP
%ratio value for the areaOverPerimeterFilterLabel    filter. Objects  with
%    an AOP ratio   smaller than this value will
%be    removed. ResizeImageScale  -  Number    specifying
% by what ratio the images   will  be   resized
% before they  are   processed. By    default
% this  value is set to 0.5.   Used  by
% the  resizeImage module.   ApproximationDistance -  Number specifying  how
%close the    convex  hull  in   the
%getConvexObjects module approximates the     object outline.  By default this
%     value is set   to 2.5. Setting it
%to  a     lower  value  will
% result  in convex hulls that more closely   resemble  the
%  object   outlines   however  this increases the
% chance of    detecting  insignificant concavities. MaxSearchRadius - Number
% specifying  the    absolute  lower bound   for
%the search radius to  prevent  selecting     too
%few  candidate   objects   for a track. Used by
% assignCellToTrackUsingAll module. MinSearchRadius  - Number specifying
%the absolute higher   bound for the search  radius  to
%prevent  selecting   too  many     candidate
%objects for a track. Used by   assignCellToTrackUsingAll  module.
%MinSecondDistance  - Number specifying  the  minimum significant distance
%between the    closest  candidate object to a  track
% and  the    second  closest. Used   to
% determine when distance should  be  used as
%a  ranking  parameter.   Used   by assignCellToTrackUsingAll module. MaxDistRatio
%-    Number specifying the  maximum allowed distance
%ratio    between the two   nearest candidate objects.
%If  the   ratio  is higher  than  this
%  value distance   ranking will not be used.
%Used  by     assignCellToTrackUsingAll  module. MaxAngleDiff -  Number
%specifying the maximum allowed     angle difference    between
% a track and a candidate     object.  If
% the angle  is larger   than   this value direction
%ranking  will  not be      used  for
%this object.   Used   by assignCellToTrackUsingAll module. NrParamsForSureMatch  -
%  Number specifying the  minimum number of     closest
% matches  between a candidate   object parameters  and
%a  track's   object  parameters that  make  the
% candidate  object a   sure match to  the track.
%  Used  by     assignCellToTrackUsingAll  module. SearchRadiusPct -
%Number specifying  the size of the     neighborhood from
%  which   candidate objects for matching  the   track
%are  selected.    It is  a multiple   of
%  the distance to  the  nearest   candidate  in
% the  current  frame.  Setting this   variable
%equal to 1 turns  this     module  into a
%nearest-neighbor algorithm (only     the   nearest  cell can
%be   a candidate). It does      not make
% sense  to have  a  value    lower
%than 1.   Used by assignCellToTrackUsingAll module. MaxMergeDistance -   Number
%specifying  the     maximum  distance that one  track
%may  be from another track      for the
%  duration   and still be considered for possible   merging
% with  the    other track.   Used by
%detectMergeCandidatesUsingDistance    module. MaxSplitArea -  Number  specifying  the maximum
% area   a  nucleus  may  be and  still
%be    considered  as a  part
%of a possible mitotic  event. Used by     detectMitoticEvents
%module. MaxSplitDistance  - Number  specifying   the maximum distance a
%  new  nucleus may   be  from another nucleus
% and    still   be considered   as part
%of a possible mitotic   event.   Used  by  detectMitoticEvents
%    module. MinSplitEccentricity -  Number  specifying  the minimum
%eccentricity a     new nucleus may     have
% and still be  considered as part   of   a
% possible    mitotic event.  Used by detectMitoticEvents module.  MaxSplitEccentricity
%   - Number specifying  the  maximum  eccentricity a
% new   nucleus  may  have  and still be
% considered    as part of a  possible
%  mitotic event. Used by  detectMitoticEvents module. MinTimeForSplit   -
%Number   specifying the  minimum time  in   minutes a
%track needs     to  exist before    it
```

```
%is considered for a   possible    mitotic   event.
%Used   by detectMitoticEvents  module. MinLifespan - Number specifying  the
% minimum  length  in    frames  a frame has
% to  be  to   not be removed
%by the removeShortTracks     module.   FrontParams - Numeric array
%specifying a set   of   column   indices from
% the shape  and motility  parameters  matrix.
%The parameters in  those  columns  will be   heavily weighted,
%  and  have  more   influence in determining
%the    best match for a  track from
%  a list of objects.   Used by   assignCellToTrackUsingAll
% module.   DefaultParamWeights -  Numeric array  specifying  a set
%of  weights that     is  assigned to
%each shape   and motility parameter   based on   its
%  prediction  power.   Parameters with  high prediction power are
%  assigned    high  weights  and  parameters
%with  low prediction power   are  assigned  lower weights.
%Used  by     assignCellToTrackUsingAll module. DistanceRankingOrder    -
%Numeric   array specifying the default  order of shape   and
%  motility   parameters for    slow  moving objects
%when it cannot  be      determined based on
%prediction  power.  Used by   assignCellToTrackUsingAll    module. DirectionRankingOrder
% -  Numeric  array specifying the default    order
%  of shape and motility  parameters for     fast
%  moving directional objects  when it cannot   be
% determined   based on  prediction power.  Used by  assignCellToTrackUsingAll
% module.  RelevantParametersIndex -     Boolean  array specifying
%  column indexes  in the shape and  motility matrix that
% have      been   determined to  be
%irrelevant for tracking.  This  indicates    to the module
%not  to     use the parameters   those
%columns  in computing   track assignment probabilities.  The   order
%of column      indexes   is provided in TracksLayout
% variable. Used by   assignCellToTrackUsingAll    module.  UnknownParamWeights -
%Numeric    array specifying a set  of weights to  be
%     assigned to shape   and  motility
%parameters when   the prediction   power of   the
%parameters  cannot be determined. Used    by
%assignCellToTrackUsingAll module. UnknownRankingOrder -  Numeric array     specifying
%the order of the  shape and motility  parameters when
% their    predictive  power cannot  be determined. If the
%  objects  cannot be     categorized as either slow-moving
%     or  fast directional the parameter  order provided
%in    this    variable  is used.  Used
%by    assignCellToTrackUsingAll module. Important  Modules  -
%areaFilterLabel, areaOverPerimeterFilterLabel, assignCellToTrackUsingAll,   clearSmallObjects,
detectMergeCandidatesUsingDistance,   detectMitoticEvents, distanceWatershed,
generateBinImgUsingLocAvg, getConvexObjects,
%polygonalAssistedWatershed,   removeShortTracks,   segmentObjectsUsingMarkers,
solidityFilterLabel, splitTracks.

global functions_list;
functions_list=[];
%script variables
ImageFolder='C:\cetres refdata b\image-contrast_1';
ImageFilesRoot='refdataB_C1_';
ImageExtension='.png';
StartFrame=1;
FrameCount=399;
TimeFrame=10;
FrameStep=1;
NumberFormat='%03d';
MaxFramesMissing=6;
OutputFolder=[ImageFolder '/output'];
AncestryFolder=[OutputFolder '/ancestry'];
AncestrySpreadsheet=[AncestryFolder 'ancestry.csv'];
ShapesSpreadsheet=[AncestryFolder 'shapes.csv'];
TracksFolder=[OutputFolder '/track'];
SegmentationFilesRoot=[TracksFolder '/grayscale'];
ImageFileBase=[ImageFolder '/' ImageFilesRoot];
BrightnessThresholdPct=1.1;
ObjectArea=30;
Strel='disk';
StrelSize=10;
ClearBorder=true;
ClearBorderDist=2;
MedianFilterSize=3;
MinSolidity=0.69;
MinAreaOverPerimeter=1.5;
ResizeImageScale=0.5;
ApproximationDistance=2.4;
```

```
MaxSearchRadius=Inf;
MinSearchRadius=0;
MinSecondDistance=5;
MaxDistRatio=0.6;
MaxAngleDiff=0.35;
NrParamsForSureMatch=5;
SearchRadiusPct=1.5;
MaxMergeDistance=23;
MaxSplitArea=400;
MaxSplitDistance=45;
MinSplitEccentricity=0;
MaxSplitEccentricity=0.95;
MinTimeForSplit=900;
MinLifespan=30;
FrontParams=[];
DefaultParamWeights=[34 21 13 8 5 3 2 2 2];
DistanceRankingOrder=[1 3 4 5 6 7 8 9 2];
DirectionRankingOrder=[2 3 4 5 6 7 8 9 1];
RelevantParametersIndex=[true true true false true false true true false];
UnknownParamWeights=[5 3 1 1 1 1 1 1 1];
UnknownRankingOrder=[1 2 3 4 5 6 7 8 9];
%end script variables

assign_cells_to_tracks_functions=[];
else_is_empty_cells_label_functions=[];
if_is_empty_cells_label_functions=[];
image_read_loop_functions=[];
image_overlay_loop_functions=[];

loadtrackslayout.InstanceName='LoadTracksLayout';
loadtrackslayout.FunctionHandle=@loadTracksLayout;
loadtrackslayout.FunctionArgs.FileName.Value='tracks_layout.mat';
functions_list=addToFunctionChain(functions_list,loadtrackslayout);

loadancestrylayout.InstanceName='LoadAncestryLayout';
loadancestrylayout.FunctionHandle=@loadAncestryLayout;
loadancestrylayout.FunctionArgs.FileName.Value='ancestry_layout.mat';
functions_list=addToFunctionChain(functions_list,loadancestrylayout);

displaycurframe.InstanceName='DisplayCurFrame';
displaycurframe.FunctionHandle=@displayVariable;
displaycurframe.FunctionArgs.VariableName.Value='Current Tracking Frame';
displaycurframe.FunctionArgs.Variable.FunctionInstance='SegmentationLoop';
displaycurframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,displaycurframe);

makeimagenamesinsegmentationloop.InstanceName='MakeImageNamesInSegmentationLoop';
makeimagenamesinsegmentationloop.FunctionHandle=@makeImgFileName;
makeimagenamesinsegmentationloop.FunctionArgs.FileBase.Value=ImageFileBase;
makeimagenamesinsegmentationloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinsegmentationloop.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makeimagenamesinsegmentati
onloop);

readimagesinsegmentationloop.InstanceName='ReadImagesInSegmentationLoop';
readimagesinsegmentationloop.FunctionHandle=@readImage;
readimagesinsegmentationloop.FunctionArgs.ImageChannel.Value='';
readimagesinsegmentationloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInSegmentatio
nLoop';
readimagesinsegmentationloop.FunctionArgs.ImageName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,readimagesinsegmentationlo
op);

gaussianblur.InstanceName='GaussianBlur';
gaussianblur.FunctionHandle=@gaussianFilter;
gaussianblur.FunctionArgs.KernelSize.Value=5;
gaussianblur.FunctionArgs.StandardDev.Value=2;
gaussianblur.FunctionArgs.Image.FunctionInstance='ReadImagesInSegmentationLoop';
gaussianblur.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,gaussianblur);

normalizeimageto16bit.InstanceName='NormalizeImageTo16Bit';
normalizeimageto16bit.FunctionHandle=@imNorm;
normalizeimageto16bit.FunctionArgs.IntegerClass.Value='uint16';
normalizeimageto16bit.FunctionArgs.RawImage.FunctionInstance='GaussianBlur';
normalizeimageto16bit.FunctionArgs.RawImage.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,normalizeimageto16bit);

resizeimage.InstanceName='ResizeImage';
```

```
resizeimage.FunctionHandle=@resizeImage;
resizeimage.FunctionArgs.Scale.Value=ResizeImageScale;
resizeimage.FunctionArgs.Method.Value='bicubic';
resizeimage.FunctionArgs.Image.FunctionInstance='NormalizeImageTo16Bit';
resizeimage.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,resizeimage);

cytobrightnesslocalaveragingfilter.InstanceName='CytoBrightnessLocalAveragingFilter';
cytobrightnesslocalaveragingfilter.FunctionHandle=@generateBinImgUsingLocAvg;
cytobrightnesslocalaveragingfilter.FunctionArgs.Strel.Value=Strel;
cytobrightnesslocalaveragingfilter.FunctionArgs.StrelSize.Value=StrelSize;
cytobrightnesslocalaveragingfilter.FunctionArgs.BrightnessThresholdPct.Value=BrightnessThresholdP
ct;
cytobrightnesslocalaveragingfilter.FunctionArgs.ClearBorder.Value=ClearBorder;
cytobrightnesslocalaveragingfilter.FunctionArgs.ClearBorderDist.Value=ClearBorderDist;
cytobrightnesslocalaveragingfilter.FunctionArgs.Image.FunctionInstance='ResizeImage';
cytobrightnesslocalaveragingfilter.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,cytobrightnesslocalaveragi
ngfilter);

fillholescytoplasmimages.InstanceName='FillHolesCytoplasmImages';
fillholescytoplasmimages.FunctionHandle=@fillHoles;
fillholescytoplasmimages.FunctionArgs.Image.FunctionInstance='CytoBrightnessLocalAveragingFilter'
;
fillholescytoplasmimages.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,fillholescytoplasmimages);

clearsmallcells.InstanceName='ClearSmallCells';
clearsmallcells.FunctionHandle=@clearSmallObjects;
clearsmallcells.FunctionArgs.MinObjectArea.Value=ObjectArea;
clearsmallcells.FunctionArgs.Image.FunctionInstance='FillHolesCytoplasmImages';
clearsmallcells.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,clearsmallcells);

nuclbrightnesslocalaveragingfilter.InstanceName='NuclBrightnessLocalAveragingFilter';
nuclbrightnesslocalaveragingfilter.FunctionHandle=@generateBinImgUsingLocAvg;
nuclbrightnesslocalaveragingfilter.FunctionArgs.Strel.Value=Strel;
nuclbrightnesslocalaveragingfilter.FunctionArgs.StrelSize.Value=StrelSize;
nuclbrightnesslocalaveragingfilter.FunctionArgs.BrightnessThresholdPct.Value=BrightnessThresholdP
ct;
nuclbrightnesslocalaveragingfilter.FunctionArgs.ClearBorder.Value=ClearBorder;
nuclbrightnesslocalaveragingfilter.FunctionArgs.ClearBorderDist.Value=ClearBorderDist;
nuclbrightnesslocalaveragingfilter.FunctionArgs.Image.FunctionInstance='ResizeImage';
nuclbrightnesslocalaveragingfilter.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,nuclbrightnesslocalaveragi
ngfilter);

fillholesnuclearimages.InstanceName='FillHolesNuclearImages';
fillholesnuclearimages.FunctionHandle=@fillHoles;
fillholesnuclearimages.FunctionArgs.Image.FunctionInstance='NuclBrightnessLocalAveragingFilter';
fillholesnuclearimages.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,fillholesnuclearimages);

clearsmallnuclei.InstanceName='ClearSmallNuclei';
clearsmallnuclei.FunctionHandle=@clearSmallObjects;
clearsmallnuclei.FunctionArgs.MinObjectArea.Value=ObjectArea;
clearsmallnuclei.FunctionArgs.Image.FunctionInstance='FillHolesNuclearImages';
clearsmallnuclei.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,clearsmallnuclei);

combinenuclearandcytoplasmimages.InstanceName='CombineNuclearAndCytoplasmImages';
combinenuclearandcytoplasmimages.FunctionHandle=@combineImages;
combinenuclearandcytoplasmimages.FunctionArgs.CombineOperation.Value='AND';
combinenuclearandcytoplasmimages.FunctionArgs.Image1.FunctionInstance='ClearSmallNuclei';
combinenuclearandcytoplasmimages.FunctionArgs.Image1.OutputArg='Image';
combinenuclearandcytoplasmimages.FunctionArgs.Image2.FunctionInstance='ClearSmallCells';
combinenuclearandcytoplasmimages.FunctionArgs.Image2.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,combinenuclearandcytoplasm
images);

reconstructcytoplasmimage.InstanceName='ReconstructCytoplasmImage';
reconstructcytoplasmimage.FunctionHandle=@reconstructObjects;
reconstructcytoplasmimage.FunctionArgs.GuideImage.FunctionInstance='CombineNuclearAndCytoplasmIma
ges';
reconstructcytoplasmimage.FunctionArgs.GuideImage.OutputArg='Image';
reconstructcytoplasmimage.FunctionArgs.ImageToReconstruct.FunctionInstance='ClearSmallNuclei';
reconstructcytoplasmimage.FunctionArgs.ImageToReconstruct.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,reconstructcytoplasmimage)
;

labelnuclei.InstanceName='LabelNuclei';
```

```
labelnuclei.FunctionHandle=@labelObjects;
labelnuclei.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
labelnuclei.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,labelnuclei);

labelcytoplasm.InstanceName='LabelCytoplasm';
labelcytoplasm.FunctionHandle=@labelObjects;
labelcytoplasm.FunctionArgs.Image.FunctionInstance='ReconstructCytoplasmImage';
labelcytoplasm.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,labelcytoplasm);

getconvexobjects.InstanceName='GetConvexObjects';
getconvexobjects.FunctionHandle=@getConvexObjects;
getconvexobjects.FunctionArgs.ApproximationDistance.Value=ApproximationDistance;
getconvexobjects.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
getconvexobjects.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getconvexobjects);

distancewatershed.InstanceName='DistanceWatershed';
distancewatershed.FunctionHandle=@distanceWatershed;
distancewatershed.FunctionArgs.MedianFilterNhood.Value=MedianFilterSize;
distancewatershed.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
distancewatershed.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,distancewatershed);

polygonalassistedwatershed.InstanceName='PolygonalAssistedWatershed';
polygonalassistedwatershed.FunctionHandle=@polygonalAssistedWatershed;
polygonalassistedwatershed.FunctionArgs.MinBlobArea.Value=ObjectArea;
polygonalassistedwatershed.FunctionArgs.ImageLabel.FunctionInstance='LabelNuclei';
polygonalassistedwatershed.FunctionArgs.ImageLabel.OutputArg='LabelMatrix';
polygonalassistedwatershed.FunctionArgs.WatershedLabel.FunctionInstance='DistanceWatershed';
polygonalassistedwatershed.FunctionArgs.WatershedLabel.OutputArg='LabelMatrix';
polygonalassistedwatershed.FunctionArgs.ConvexObjectsIndex.FunctionInstance='GetConvexObjects';
polygonalassistedwatershed.FunctionArgs.ConvexObjectsIndex.OutputArg='ConvexObjectsIndex';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,polygonalassistedwatershed
);

segmentobjectsusingmarkers.InstanceName='SegmentObjectsUsingMarkers';
segmentobjectsusingmarkers.FunctionHandle=@segmentObjectsUsingMarkers;
segmentobjectsusingmarkers.FunctionArgs.MarkersLabel.FunctionInstance='PolygonalAssistedWatershed
';
segmentobjectsusingmarkers.FunctionArgs.MarkersLabel.OutputArg='LabelMatrix';
segmentobjectsusingmarkers.FunctionArgs.ObjectsLabel.FunctionInstance='LabelCytoplasm';
segmentobjectsusingmarkers.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,segmentobjectsusingmarkers
);

areafilter.InstanceName='AreaFilter';
areafilter.FunctionHandle=@areaFilterLabel;
areafilter.FunctionArgs.MinArea.Value=ObjectArea;
areafilter.FunctionArgs.ObjectsLabel.FunctionInstance='SegmentObjectsUsingMarkers';
areafilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,areafilter);

solidityfilter.InstanceName='SolidityFilter';
solidityfilter.FunctionHandle=@solidityFilterLabel;
solidityfilter.FunctionArgs.MinSolidity.Value=MinSolidity;
solidityfilter.FunctionArgs.ObjectsLabel.FunctionInstance='AreaFilter';
solidityfilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,solidityfilter);

aoverpfilter.InstanceName='AOverPFilter';
aoverpfilter.FunctionHandle=@areaOverPerimeterFilterLabel;
aoverpfilter.FunctionArgs.MinAreaOverPerimeter.Value=MinAreaOverPerimeter;
aoverpfilter.FunctionArgs.ObjectsLabel.FunctionInstance='SolidityFilter';
aoverpfilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,aoverpfilter);

resizecytolabel.InstanceName='ResizeCytoLabel';
resizecytolabel.FunctionHandle=@resizeImage;
resizecytolabel.FunctionArgs.Scale.Value=1/ResizeImageScale;
resizecytolabel.FunctionArgs.Method.Value='nearest';
resizecytolabel.FunctionArgs.Image.FunctionInstance='AOverPFilter';
resizecytolabel.FunctionArgs.Image.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,resizecytolabel);

makeexportfilenames.InstanceName='MakeExportFileNames';
makeexportfilenames.FunctionHandle=@makeImgFileName;
makeexportfilenames.FunctionArgs.FileBase.Value='C:\darren\movie2\cellanimation output\exported
labels\CALabel';
makeexportfilenames.FunctionArgs.FileExt.Value='.tif';
```

```
makeexportfilenames.FunctionArgs.NumberFmt.Value=NumberFormat;
makeexportfilenames.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
makeexportfilenames.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makeexportfilenames);

exportlabeltoimage.InstanceName='ExportLabelToImage';
exportlabeltoimage.FunctionHandle=@exportLabelToImage;
exportlabeltoimage.FunctionArgs.Format.Value='tif';
exportlabeltoimage.FunctionArgs.Image.FunctionInstance='ResizeCytoLabel';
exportlabeltoimage.FunctionArgs.Image.OutputArg='Image';
exportlabeltoimage.FunctionArgs.FileName.FunctionInstance='MakeExportFileNames';
exportlabeltoimage.FunctionArgs.FileName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,exportlabeltoimage);

getshapeparameters.InstanceName='GetShapeParameters';
getshapeparameters.FunctionHandle=@getShapeParams;
getshapeparameters.FunctionArgs.LabelMatrix.FunctionInstance='ResizeCytoLabel';
getshapeparameters.FunctionArgs.LabelMatrix.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getshapeparameters);

isemptypreviouscellslabel.InstanceName='IsEmptyPreviousCellsLabel';
isemptypreviouscellslabel.FunctionHandle=@isEmptyFunction;
isemptypreviouscellslabel.FunctionArgs.TestVariable.Value=[];
isemptypreviouscellslabel.FunctionArgs.TestVariable.FunctionInstance='SaveCellsLabel';
isemptypreviouscellslabel.FunctionArgs.TestVariable.OutputArg='CellsLabel';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,isemptypreviouscellslabel)
;

starttracks.InstanceName='StartTracks';
starttracks.FunctionHandle=@startTracks;
starttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
starttracks.FunctionArgs.CellsLabel.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.CellsLabel.InputArg='ResizeCytoLabel_Image';
starttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
starttracks.FunctionArgs.ShapeParameters.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.ShapeParameters.InputArg='GetShapeParameters_ShapeParameters';
if_is_empty_cells_label_functions=addToFunctionChain(if_is_empty_cells_label_functions,starttrack
s);

getcurrenttracks.InstanceName='GetCurrentTracks';
getcurrenttracks.FunctionHandle=@getCurrentTracks;
getcurrenttracks.FunctionArgs.OffsetFrame.Value=-1;
getcurrenttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getcurrenttracks.FunctionArgs.TimeCol.Value=2;
getcurrenttracks.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getcurrenttracks.FunctionArgs.FrameStep.Value=FrameStep;
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg='Tracks';
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getcurrenttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getcurrenttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getcur
renttracks);

getprevioustracks.InstanceName='GetPreviousTracks';
getprevioustracks.FunctionHandle=@getCurrentTracks;
getprevioustracks.FunctionArgs.OffsetFrame.Value=-2;
getprevioustracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getprevioustracks.FunctionArgs.TimeCol.Value=2;
getprevioustracks.FunctionArgs.TrackIDCol.Value=1;
getprevioustracks.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getprevioustracks.FunctionArgs.FrameStep.Value=FrameStep;
getprevioustracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg='Tracks';
getprevioustracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getprevioustracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getprevioustracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getpre
vioustracks);

makeunassignedcellslist.InstanceName='MakeUnassignedCellsList';
makeunassignedcellslist.FunctionHandle=@makeUnassignedCellsList;
makeunassignedcellslist.FunctionArgs.CellsCentroids.FunctionInstance='IfIsEmptyPreviousCellsLabel
';
makeunassignedcellslist.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,makeun
assignedcellslist);
```

```
makeexcludedtrackslist.InstanceName='MakeExcludedTracksList';
makeexcludedtrackslist.FunctionHandle=@makeExcludedTracksList;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.FunctionInstance='MakeUnassignedCellsList'
;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.OutputArg='UnassignedCellsIDs';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,makeex
cludedtrackslist);

getcellsmeandisplacement.InstanceName='GetCellsMeanDisplacement';
getcellsmeandisplacement.FunctionHandle=@getObjectsMeanDisplacement;
getcellsmeandisplacement.FunctionArgs.Centroid1Col.Value=3;
getcellsmeandisplacement.FunctionArgs.Centroid2Col.Value=4;
getcellsmeandisplacement.FunctionArgs.CurrentTracks.FunctionInstance='GetCurrentTracks';
getcellsmeandisplacement.FunctionArgs.CurrentTracks.OutputArg='Tracks';
getcellsmeandisplacement.FunctionArgs.ObjectCentroids.FunctionInstance='IfIsEmptyPreviousCellsLab
el';
getcellsmeandisplacement.FunctionArgs.ObjectCentroids.InputArg='GetShapeParameters_Centroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getcel
lsmeandisplacement);

getparamscoefficientofvariation.InstanceName='GetParamsCoefficientOfVariation';
getparamscoefficientofvariation.FunctionHandle=@getParamsCoefficientOfVariation;
getparamscoefficientofvariation.FunctionArgs.AreaCol.Value=5;
getparamscoefficientofvariation.FunctionArgs.SolidityCol.Value=11;
getparamscoefficientofvariation.FunctionArgs.Params.FunctionInstance='IfIsEmptyPreviousCellsLabel
';
getparamscoefficientofvariation.FunctionArgs.Params.InputArg='GetShapeParameters_ShapeParameters'
;
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getpar
amscoefficientofvariation);

getmaxtrackid.InstanceName='GetMaxTrackID';
getmaxtrackid.FunctionHandle=@getMaxTrackID;
getmaxtrackid.FunctionArgs.TrackIDCol.Value=1;
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg='Tracks';
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg2='Tracks';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getmax
trackid);

isnotemptyunassignedcells.InstanceName='IsNotEmptyUnassignedCells';
isnotemptyunassignedcells.FunctionHandle=@isNotEmptyFunction;
isnotemptyunassignedcells.FunctionArgs.TestVariable.FunctionInstance='AssignCellToTrackUsingAll';
isnotemptyunassignedcells.FunctionArgs.TestVariable.OutputArg='UnassignedIDs';
isnotemptyunassignedcells.FunctionArgs.TestVariable.FunctionInstance2='AssignCellsToTracksLoop';
isnotemptyunassignedcells.FunctionArgs.TestVariable.InputArg2='MakeUnassignedCellsList_Unassigned
CellsIDs';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,isnotemptyun
assignedcells);

getcurrentunassignedcell.InstanceName='GetCurrentUnassignedCell';
getcurrentunassignedcell.FunctionHandle=@getCurrentUnassignedCell;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance='AssignCellToTrackUsingAll
';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTracksLoop'
;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassign
edCellsIDs';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,getcurrentun
assignedcell);

assigncelltotrackusingall.InstanceName='AssignCellToTrackUsingAll';
assigncelltotrackusingall.FunctionHandle=@assignCellToTrackUsingAll;
assigncelltotrackusingall.FunctionArgs.CheckCellPath.Value=true;
assigncelltotrackusingall.FunctionArgs.FrontParams.Value=FrontParams;
assigncelltotrackusingall.FunctionArgs.MaxSearchRadius.Value=MaxSearchRadius;
assigncelltotrackusingall.FunctionArgs.MinSearchRadius.Value=MinSearchRadius;
assigncelltotrackusingall.FunctionArgs.SearchRadiusPct.Value=SearchRadiusPct;
assigncelltotrackusingall.FunctionArgs.RelevantParametersIndex.Value=RelevantParametersIndex;
assigncelltotrackusingall.FunctionArgs.NrParamsForSureMatch.Value=NrParamsForSureMatch;
assigncelltotrackusingall.FunctionArgs.DefaultParamWeights.Value=DefaultParamWeights;
assigncelltotrackusingall.FunctionArgs.UnknownParamWeights.Value=UnknownParamWeights;
assigncelltotrackusingall.FunctionArgs.DistanceRankingOrder.Value=DistanceRankingOrder;
assigncelltotrackusingall.FunctionArgs.DirectionRankingOrder.Value=DirectionRankingOrder;
assigncelltotrackusingall.FunctionArgs.UnknownRankingOrder.Value=UnknownRankingOrder;
assigncelltotrackusingall.FunctionArgs.MinSecondDistance.Value=MinSecondDistance;
assigncelltotrackusingall.FunctionArgs.MaxDistRatio.Value=MaxDistRatio;
assigncelltotrackusingall.FunctionArgs.MaxAngleDiff.Value=MaxAngleDiff;
```

```
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.Value=[];
assigncelltotrackusingall.FunctionArgs.TrackAssignments.Value=[];
assigncelltotrackusingall.FunctionArgs.MatchingGroups.Value=[];
assigncelltotrackusingall.FunctionArgs.MatchingGroupsStats.Value=[];
assigncelltotrackusingall.FunctionArgs.UnassignedCells.FunctionInstance='AssignCellToTrackUsingAl
l';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.FunctionInstance='AssignCellToTrackUsingAll
';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.OutputArg='ExcludedTracks';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.FunctionInstance='SetMatchingGroupIndex';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.OutputArg='ShapeParameters';
assigncelltotrackusingall.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellToTrackUsingA
ll';
assigncelltotrackusingall.FunctionArgs.TrackAssignments.OutputArg='TrackAssignments';
assigncelltotrackusingall.FunctionArgs.MatchingGroups.FunctionInstance='AssignCellToTrackUsingAll
';
assigncelltotrackusingall.FunctionArgs.MatchingGroups.OutputArg='MatchingGroups';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTracksLoop
';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassig
nedCellsIDs';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.FunctionInstance2='AssignCellsToTracksLoop'
;
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.InputArg2='MakeExcludedTracksList_ExcludedT
racks';
assigncelltotrackusingall.FunctionArgs.CellsLabel.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CellsLabel.InputArg='ResizeCytoLabel_Image';
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.FunctionInstance='AssignCellsToTracksLo
op';
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.InputArg='SaveCellsLabel_CellsLabel';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.FunctionInstance2='AssignCellsToTracksLoop
';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.InputArg2='GetShapeParameters_ShapeParamet
ers';
assigncelltotrackusingall.FunctionArgs.CellsCentroids.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
assigncelltotrackusingall.FunctionArgs.CurrentTracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CurrentTracks.InputArg='GetCurrentTracks_Tracks';
assigncelltotrackusingall.FunctionArgs.MaxTrackID.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.MaxTrackID.InputArg='GetMaxTrackID_MaxTrackID';
assigncelltotrackusingall.FunctionArgs.Tracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.Tracks.InputArg='IfIsEmptyPreviousCellsLabel_Tracks';
assigncelltotrackusingall.FunctionArgs.MatchingGroupsStats.FunctionInstance='AssignCellsToTracksL
oop';
assigncelltotrackusingall.FunctionArgs.MatchingGroupsStats.InputArg='GetMatchingGroupMeans_Matchi
ngGroupStats';
assigncelltotrackusingall.FunctionArgs.ParamsCoeffOfVariation.FunctionInstance='AssignCellsToTrac
ksLoop';
assigncelltotrackusingall.FunctionArgs.ParamsCoeffOfVariation.InputArg='GetParamsCoefficientOfVar
iation_CoefficientOfVariation';
assigncelltotrackusingall.FunctionArgs.PreviousTracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.PreviousTracks.InputArg='GetPreviousTracks_Tracks';
assigncelltotrackusingall.FunctionArgs.TracksLayout.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
assigncelltotrackusingall.FunctionArgs.MatchingGroups.FunctionInstance2='AssignCellsToTracksLoop'
;
assigncelltotrackusingall.FunctionArgs.MatchingGroups.InputArg2='HoldMatchingGroups_ValueToHold';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,assigncellto
trackusingall);

setmatchinggroupindex.InstanceName='SetMatchingGroupIndex';
setmatchinggroupindex.FunctionHandle=@setGroupIndex;
setmatchinggroupindex.FunctionArgs.AreaCol.Value=5;
setmatchinggroupindex.FunctionArgs.GroupIDCol.Value=13;
setmatchinggroupindex.FunctionArgs.CellID.FunctionInstance='GetCurrentUnassignedCell';
setmatchinggroupindex.FunctionArgs.CellID.OutputArg='CellID';
setmatchinggroupindex.FunctionArgs.GroupIndex.FunctionInstance='AssignCellToTrackUsingAll';
setmatchinggroupindex.FunctionArgs.GroupIndex.OutputArg='GroupIndex';
setmatchinggroupindex.FunctionArgs.ShapeParameters.FunctionInstance='SetMatchingGroupIndex';
setmatchinggroupindex.FunctionArgs.ShapeParameters.OutputArg='ShapeParameters';
setmatchinggroupindex.FunctionArgs.ShapeParameters.FunctionInstance2='AssignCellsToTracksLoop';
setmatchinggroupindex.FunctionArgs.ShapeParameters.InputArg2='GetShapeParameters_ShapeParameters'
;
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,setmatchingg
roupindex);

assigncellstotracksloop.InstanceName='AssignCellsToTracksLoop';
assigncellstotracksloop.FunctionHandle=@whileLoop;
assigncellstotracksloop.FunctionArgs.TestFunction.FunctionInstance='IsNotEmptyUnassignedCells';
assigncellstotracksloop.FunctionArgs.TestFunction.OutputArg='Boolean';
```

```
assigncellstotracksloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.FunctionInstance=
'MakeUnassignedCellsList';
assigncellstotracksloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.OutputArg='Unassi
gnedCellsIDs';
assigncellstotracksloop.FunctionArgs.MakeExcludedTracksList_ExcludedTracks.FunctionInstance='Make
ExcludedTracksList';
assigncellstotracksloop.FunctionArgs.MakeExcludedTracksList_ExcludedTracks.OutputArg='ExcludedTra
cks';
assigncellstotracksloop.FunctionArgs.GetCurrentTracks_Tracks.FunctionInstance='GetCurrentTracks';
assigncellstotracksloop.FunctionArgs.GetCurrentTracks_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.GetMaxTrackID_MaxTrackID.FunctionInstance='GetMaxTrackID';
assigncellstotracksloop.FunctionArgs.GetMaxTrackID_MaxTrackID.OutputArg='MaxTrackID';
assigncellstotracksloop.FunctionArgs.IfIsEmptyPreviousCellsLabel_Tracks.FunctionInstance='IfIsEmp
tyPreviousCellsLabel';
assigncellstotracksloop.FunctionArgs.IfIsEmptyPreviousCellsLabel_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.GetMatchingGroupMeans_MatchingGroupStats.FunctionInstance='G
etMatchingGroupMeans';
assigncellstotracksloop.FunctionArgs.GetMatchingGroupMeans_MatchingGroupStats.OutputArg='Matching
GroupStats';
assigncellstotracksloop.FunctionArgs.GetParamsCoefficientOfVariation_CoefficientOfVariation.Funct
ionInstance='GetParamsCoefficientOfVariation';
assigncellstotracksloop.FunctionArgs.GetParamsCoefficientOfVariation_CoefficientOfVariation.Outpu
tArg='CoefficientOfVariation';
assigncellstotracksloop.FunctionArgs.GetPreviousTracks_Tracks.FunctionInstance='GetPreviousTracks
';
assigncellstotracksloop.FunctionArgs.GetPreviousTracks_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.ResizeCytoLabel_Image.FunctionInstance='IfIsEmptyPreviousCel
lsLabel';
assigncellstotracksloop.FunctionArgs.ResizeCytoLabel_Image.InputArg='ResizeCytoLabel_Image';
assigncellstotracksloop.FunctionArgs.SaveCellsLabel_CellsLabel.FunctionInstance='IfIsEmptyPreviou
sCellsLabel';
assigncellstotracksloop.FunctionArgs.SaveCellsLabel_CellsLabel.InputArg='SaveCellsLabel_CellsLabe
l';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_ShapeParameters.FunctionInstance='IfIsEmp
tyPreviousCellsLabel';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_ShapeParameters.InputArg='GetShapeParamet
ers_ShapeParameters';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_Centroids.FunctionInstance='IfIsEmptyPrev
iousCellsLabel';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_Centroids.InputArg='GetShapeParameters_Ce
ntroids';
assigncellstotracksloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='IfIsEmptyPre
viousCellsLabel';
assigncellstotracksloop.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout_Tra
cksLayout';
assigncellstotracksloop.FunctionArgs.HoldMatchingGroups_ValueToHold.FunctionInstance='IfIsEmptyPr
eviousCellsLabel';
assigncellstotracksloop.FunctionArgs.HoldMatchingGroups_ValueToHold.InputArg='HoldMatchingGroups_
ValueToHold';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_TrackAssignments.FunctionInstance='A
ssignCellToTrackUsingAll';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_TrackAssignments.OutputArg='TrackAss
ignments';
assigncellstotracksloop.KeepValues.SetMatchingGroupIndex_ShapeParameters.FunctionInstance='SetMat
chingGroupIndex';
assigncellstotracksloop.KeepValues.SetMatchingGroupIndex_ShapeParameters.OutputArg='ShapeParamete
rs';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.FunctionInstance='Ass
ignCellToTrackUsingAll';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.OutputArg='MatchingGr
oups';
assigncellstotracksloop.LoopFunctions=assign_cells_to_tracks_functions;
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,assign
cellstotracksloop);

continuetracks.InstanceName='ContinueTracks';
continuetracks.FunctionHandle=@continueTracks;
continuetracks.FunctionArgs.TimeFrame.Value=TimeFrame;
continuetracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
continuetracks.FunctionArgs.Tracks.OutputArg='Tracks';
continuetracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
continuetracks.FunctionArgs.Tracks.OutputArg2='Tracks';
continuetracks.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellsToTracksLoop';
continuetracks.FunctionArgs.TrackAssignments.OutputArg='AssignCellToTrackUsingAll_TrackAssignment
s';
continuetracks.FunctionArgs.ShapeParameters.FunctionInstance='AssignCellsToTracksLoop';
continuetracks.FunctionArgs.ShapeParameters.OutputArg='SetMatchingGroupIndex_ShapeParameters';
continuetracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
continuetracks.FunctionArgs.CellsCentroids.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
```

```
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,contin
uetracks);

getmatchinggroupmeans.InstanceName='GetMatchingGroupMeans';
getmatchinggroupmeans.FunctionHandle=@getMatchingGroupMeans;
getmatchinggroupmeans.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getmatchinggroupmeans.FunctionArgs.Tracks.OutputArg='Tracks';
getmatchinggroupmeans.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getmatchinggroupmeans.FunctionArgs.Tracks.OutputArg2='Tracks';
getmatchinggroupmeans.FunctionArgs.TracksLayout.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getmatchinggroupmeans.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getmat
chinggroupmeans);

ifisemptypreviouscellslabel.InstanceName='IfIsEmptyPreviousCellsLabel';
ifisemptypreviouscellslabel.FunctionHandle=@if_statement;
ifisemptypreviouscellslabel.FunctionArgs.PreviousCellsLabel.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.MatchingGroupsStats.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.TrackAssignments.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.TestVariable.FunctionInstance='IsEmptyPreviousCellsLabel
';
ifisemptypreviouscellslabel.FunctionArgs.TestVariable.OutputArg='Boolean';
ifisemptypreviouscellslabel.FunctionArgs.ResizeCytoLabel_Image.FunctionInstance='ResizeCytoLabel'
;
ifisemptypreviouscellslabel.FunctionArgs.ResizeCytoLabel_Image.OutputArg='Image';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_LoopCounter.FunctionInstance='Segmentat
ionLoop';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_LoopCounter.OutputArg='LoopCounter';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_ShapeParameters.FunctionInstance='Get
ShapeParameters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_ShapeParameters.OutputArg='ShapeParam
eters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_Centroids.FunctionInstance='GetShapeP
arameters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_Centroids.OutputArg='Centroids';
ifisemptypreviouscellslabel.FunctionArgs.SaveCellsLabel_CellsLabel.FunctionInstance='SaveCellsLab
el';
ifisemptypreviouscellslabel.FunctionArgs.SaveCellsLabel_CellsLabel.OutputArg='CellsLabel';
ifisemptypreviouscellslabel.FunctionArgs.HoldMatchingGroups_ValueToHold.FunctionInstance='HoldMat
chingGroups';
ifisemptypreviouscellslabel.FunctionArgs.HoldMatchingGroups_ValueToHold.OutputArg='ValueToHold';
ifisemptypreviouscellslabel.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='Segmenta
tionLoop';
ifisemptypreviouscellslabel.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout
_TracksLayout';
ifisemptypreviouscellslabel.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.FunctionInstance=
'AssignCellsToTracksLoop';
ifisemptypreviouscellslabel.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.OutputArg='Assign
CellToTrackUsingAll_MatchingGroups';
ifisemptypreviouscellslabel.KeepValues.ContinueTracks_Tracks.FunctionInstance='ContinueTracks';
ifisemptypreviouscellslabel.KeepValues.ContinueTracks_Tracks.OutputArg='Tracks';
ifisemptypreviouscellslabel.ElseFunctions=else_is_empty_cells_label_functions;
ifisemptypreviouscellslabel.IfFunctions=if_is_empty_cells_label_functions;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,ifisemptypreviouscellslabe
l);

savecellslabel.InstanceName='SaveCellsLabel';
savecellslabel.FunctionHandle=@saveCellsLabel;
savecellslabel.FunctionArgs.FileRoot.Value=SegmentationFilesRoot;
savecellslabel.FunctionArgs.NumberFormat.Value=NumberFormat;
savecellslabel.FunctionArgs.CellsLabel.FunctionInstance='ResizeCytoLabel';
savecellslabel.FunctionArgs.CellsLabel.OutputArg='Image';
savecellslabel.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
savecellslabel.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,savecellslabel);

holdmatchinggroups.InstanceName='HoldMatchingGroups';
holdmatchinggroups.FunctionHandle=@holdValue;
holdmatchinggroups.FunctionArgs.ValueToHold.Value=[];
holdmatchinggroups.FunctionArgs.ValueToHold.FunctionInstance='IfIsEmptyPreviousCellsLabel';
holdmatchinggroups.FunctionArgs.ValueToHold.OutputArg='AssignCellToTrackUsingAll_MatchingGroups';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,holdmatchinggroups);

segmentationloop.InstanceName='SegmentationLoop';
segmentationloop.FunctionHandle=@forLoop;
segmentationloop.FunctionArgs.StartLoop.Value=StartFrame;
segmentationloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
segmentationloop.FunctionArgs.IncrementLoop.Value=FrameStep;
segmentationloop.FunctionArgs.MatchingGroups.Value=[];
segmentationloop.FunctionArgs.Tracks.Value=[];
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='LoadTracksLayout';
```

```
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.OutputArg='TracksLayout';
segmentationloop.KeepValues.ContinueTracks_Tracks.FunctionInstance='IfIsEmptyPreviousCellsLabel';
segmentationloop.KeepValues.ContinueTracks_Tracks.OutputArg='ContinueTracks_Tracks';
segmentationloop.LoopFunctions=image_read_loop_functions;
functions_list=addToFunctionChain(functions_list,segmentationloop);

savetracks.InstanceName='SaveTracks';
savetracks.FunctionHandle=@saveTracks;
savetracks.FunctionArgs.TracksFileName.Value=[TracksFolder '/tracks.mat'];
savetracks.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
savetracks.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,savetracks);

gettrackids.InstanceName='GetTrackIDs';
gettrackids.FunctionHandle=@getTrackIDs;
gettrackids.FunctionArgs.TrackIDCol.Value=1;
gettrackids.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
gettrackids.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,gettrackids);

detectmergecandidates.InstanceName='DetectMergeCandidates';
detectmergecandidates.FunctionHandle=@detectMergeCandidatesUsingDistance;
detectmergecandidates.FunctionArgs.MaxMergeDistance.Value=MaxMergeDistance;
detectmergecandidates.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDs';
detectmergecandidates.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
detectmergecandidates.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
detectmergecandidates.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
detectmergecandidates.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
detectmergecandidates.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,detectmergecandidates);

mergetracks.InstanceName='MergeTracks';
mergetracks.FunctionHandle=@mergeTracks;
mergetracks.FunctionArgs.FrameCount.Value=FrameCount;
mergetracks.FunctionArgs.StartFrame.Value=StartFrame;
mergetracks.FunctionArgs.TimeFrame.Value=TimeFrame;
mergetracks.FunctionArgs.SegFileRoot.Value=SegmentationFilesRoot;
mergetracks.FunctionArgs.FrameStep.Value=FrameStep;
mergetracks.FunctionArgs.NumberFormat.Value=NumberFormat;
mergetracks.FunctionArgs.TracksToBeMerged.FunctionInstance='DetectMergeCandidates';
mergetracks.FunctionArgs.TracksToBeMerged.OutputArg='TracksToBeMerged';
mergetracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
mergetracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
mergetracks.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
mergetracks.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,mergetracks);

gettrackidsaftermerge.InstanceName='GetTrackIDsAfterMerge';
gettrackidsaftermerge.FunctionHandle=@getTrackIDs;
gettrackidsaftermerge.FunctionArgs.TrackIDCol.Value=1;
gettrackidsaftermerge.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
gettrackidsaftermerge.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,gettrackidsaftermerge);

makeancestryforfirstframecells.InstanceName='MakeAncestryForFirstFrameCells';
makeancestryforfirstframecells.FunctionHandle=@makeAncestryForFirstFrameCells;
makeancestryforfirstframecells.FunctionArgs.TimeCol.Value=2;
makeancestryforfirstframecells.FunctionArgs.TrackIDCol.Value=1;
makeancestryforfirstframecells.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
makeancestryforfirstframecells.FunctionArgs.Tracks.OutputArg='Tracks';
makeancestryforfirstframecells.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDsAfterMerge';
makeancestryforfirstframecells.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
functions_list=addToFunctionChain(functions_list,makeancestryforfirstframecells);

detectmitoticevents.InstanceName='DetectMitoticEvents';
detectmitoticevents.FunctionHandle=@detectMitoticEvents;
detectmitoticevents.FunctionArgs.MaxSplitArea.Value=MaxSplitArea;
detectmitoticevents.FunctionArgs.MinSplitEccentricity.Value=MinSplitEccentricity;
detectmitoticevents.FunctionArgs.MaxSplitEccentricity.Value=MaxSplitEccentricity;
detectmitoticevents.FunctionArgs.MaxSplitDistance.Value=MaxSplitDistance;
detectmitoticevents.FunctionArgs.MinTimeForSplit.Value=MinTimeForSplit;
detectmitoticevents.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
detectmitoticevents.FunctionArgs.Tracks.OutputArg='Tracks';
detectmitoticevents.FunctionArgs.UntestedIDs.FunctionInstance='MakeAncestryForFirstFrameCells';
detectmitoticevents.FunctionArgs.UntestedIDs.OutputArg='UntestedIDs';
detectmitoticevents.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
detectmitoticevents.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
functions_list=addToFunctionChain(functions_list,detectmitoticevents);

makeancestryforcellsenteringframes.InstanceName='MakeAncestryForCellsEnteringFrames';
makeancestryforcellsenteringframes.FunctionHandle=@makeAncestryForCellsEnteringFrames;
```

```
makeancestryforcellsenteringframes.FunctionArgs.TimeCol.Value=2;
makeancestryforcellsenteringframes.FunctionArgs.TrackIDCol.Value=1;
makeancestryforcellsenteringframes.FunctionArgs.SplitCells.FunctionInstance='DetectMitoticEvents'
;
makeancestryforcellsenteringframes.FunctionArgs.SplitCells.OutputArg='SplitCells';
makeancestryforcellsenteringframes.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDsAfterMerge'
;
makeancestryforcellsenteringframes.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
makeancestryforcellsenteringframes.FunctionArgs.FirstFrameIDs.FunctionInstance='MakeAncestryForFi
rstFrameCells';
makeancestryforcellsenteringframes.FunctionArgs.FirstFrameIDs.OutputArg='FirstFrameIDs';
makeancestryforcellsenteringframes.FunctionArgs.CellsAncestry.FunctionInstance='MakeAncestryForFi
rstFrameCells';
makeancestryforcellsenteringframes.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
makeancestryforcellsenteringframes.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
makeancestryforcellsenteringframes.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,makeancestryforcellsenteringframes);

splittracks.InstanceName='SplitTracks';
splittracks.FunctionHandle=@splitTracks;
splittracks.FunctionArgs.TimeFrame.Value=TimeFrame;
splittracks.FunctionArgs.SplitCells.FunctionInstance='DetectMitoticEvents';
splittracks.FunctionArgs.SplitCells.OutputArg='SplitCells';
splittracks.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
splittracks.FunctionArgs.Tracks.OutputArg='Tracks';
splittracks.FunctionArgs.CellsAncestry.FunctionInstance='MakeAncestryForCellsEnteringFrames';
splittracks.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
splittracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
splittracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
splittracks.FunctionArgs.AncestryLayout.FunctionInstance='LoadAncestryLayout';
splittracks.FunctionArgs.AncestryLayout.OutputArg='AncestryLayout';
functions_list=addToFunctionChain(functions_list,splittracks);

removeshorttracks.InstanceName='RemoveShortTracks';
removeshorttracks.FunctionHandle=@removeShortTracks;
removeshorttracks.FunctionArgs.MinLifespan.Value=MinLifespan;
removeshorttracks.FunctionArgs.Tracks.FunctionInstance='SplitTracks';
removeshorttracks.FunctionArgs.Tracks.OutputArg='Tracks';
removeshorttracks.FunctionArgs.CellsAncestry.FunctionInstance='SplitTracks';
removeshorttracks.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
removeshorttracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
removeshorttracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
removeshorttracks.FunctionArgs.AncestryLayout.FunctionInstance='LoadAncestryLayout';
removeshorttracks.FunctionArgs.AncestryLayout.OutputArg='AncestryLayout';
functions_list=addToFunctionChain(functions_list,removeshorttracks);

saveupdatedtracks.InstanceName='SaveUpdatedTracks';
saveupdatedtracks.FunctionHandle=@saveTracks;
saveupdatedtracks.FunctionArgs.TracksFileName.Value=[AncestryFolder '/tracks.mat'];
saveupdatedtracks.FunctionArgs.Tracks.FunctionInstance='RemoveShortTracks';
saveupdatedtracks.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,saveupdatedtracks);

saveancestry.InstanceName='SaveAncestry';
saveancestry.FunctionHandle=@saveAncestry;
saveancestry.FunctionArgs.AncestryFileName.Value=[AncestryFolder '/ancestry.mat'];
saveancestry.FunctionArgs.CellsAncestry.FunctionInstance='RemoveShortTracks';
saveancestry.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
functions_list=addToFunctionChain(functions_list,saveancestry);

makeimagenamesinoverlayloop.InstanceName='MakeImageNamesInOverlayLoop';
makeimagenamesinoverlayloop.FunctionHandle=@makeImgFileName;
makeimagenamesinoverlayloop.FunctionArgs.FileBase.Value=ImageFileBase;
makeimagenamesinoverlayloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinoverlayloop.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenamesinoverlayloop.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
makeimagenamesinoverlayloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,makeimagenamesinover
layloop);

readimagesinoverlayloop.InstanceName='ReadImagesInOverlayLoop';
readimagesinoverlayloop.FunctionHandle=@readImage;
readimagesinoverlayloop.FunctionArgs.ImageChannel.Value='';
readimagesinoverlayloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInOverlayLoop';
readimagesinoverlayloop.FunctionArgs.ImageName.OutputArg='FileName';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,readimagesinoverlayl
oop);

getcurrenttracks2.InstanceName='GetCurrentTracks2';
getcurrenttracks2.FunctionHandle=@getCurrentTracks;
getcurrenttracks2.FunctionArgs.OffsetFrame.Value=0;
```

```
getcurrenttracks2.FunctionArgs.TimeFrame.Value=TimeFrame;
getcurrenttracks2.FunctionArgs.TimeCol.Value=2;
getcurrenttracks2.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks2.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getcurrenttracks2.FunctionArgs.FrameStep.Value=FrameStep;
getcurrenttracks2.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
getcurrenttracks2.FunctionArgs.CurFrame.OutputArg='LoopCounter';
getcurrenttracks2.FunctionArgs.Tracks.FunctionInstance='ImageOverlayLoop';
getcurrenttracks2.FunctionArgs.Tracks.InputArg='RemoveShortTracks_Tracks';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,getcurrenttracks2);

makematnamesinoverlayloop.InstanceName='MakeMatNamesInOverlayLoop';
makematnamesinoverlayloop.FunctionHandle=@makeImgFileName;
makematnamesinoverlayloop.FunctionArgs.FileBase.Value=SegmentationFilesRoot;
makematnamesinoverlayloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makematnamesinoverlayloop.FunctionArgs.FileExt.Value='.mat';
makematnamesinoverlayloop.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
makematnamesinoverlayloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,makematnamesinoverla
yloop);

loadcellslabel.InstanceName='LoadCellsLabel';
loadcellslabel.FunctionHandle=@loadCellsLabel;
loadcellslabel.FunctionArgs.FileName.FunctionInstance='MakeMatNamesInOverlayLoop';
loadcellslabel.FunctionArgs.FileName.OutputArg='FileName';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,loadcellslabel);

loadcolormap.InstanceName='LoadColormap';
loadcolormap.FunctionHandle=@loadColormap;
loadcolormap.FunctionArgs.FileName.Value='colormap_lines';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,loadcolormap);

displayoverlayingframe.InstanceName='DisplayOverlayingFrame';
displayoverlayingframe.FunctionHandle=@displayVariable;
displayoverlayingframe.FunctionArgs.VariableName.Value='Overlaying Frame';
displayoverlayingframe.FunctionArgs.Variable.FunctionInstance='ImageOverlayLoop';
displayoverlayingframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,displayoverlayingfra
me);

displayancestry.InstanceName='DisplayAncestry';
displayancestry.FunctionHandle=@displayAncestryData;
displayancestry.FunctionArgs.NumberFormat.Value=NumberFormat;
displayancestry.FunctionArgs.ProlDir.Value=AncestryFolder;
displayancestry.FunctionArgs.ImageFileName.Value=ImageFilesRoot;
displayancestry.FunctionArgs.DS.Value='/';
displayancestry.FunctionArgs.LabelColorRGB.Value=[0 0 0];
displayancestry.FunctionArgs.Image.FunctionInstance='ReadImagesInOverlayLoop';
displayancestry.FunctionArgs.Image.OutputArg='Image';
displayancestry.FunctionArgs.CurrentTracks.FunctionInstance='GetCurrentTracks2';
displayancestry.FunctionArgs.CurrentTracks.OutputArg='Tracks';
displayancestry.FunctionArgs.CellsLabel.FunctionInstance='LoadCellsLabel';
displayancestry.FunctionArgs.CellsLabel.OutputArg='LabelMatrix';
displayancestry.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.CurFrame.OutputArg='LoopCounter';
displayancestry.FunctionArgs.ColorMap.FunctionInstance='LoadColormap';
displayancestry.FunctionArgs.ColorMap.OutputArg='Colormap';
displayancestry.FunctionArgs.CellsAncestry.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.CellsAncestry.InputArg='RemoveShortTracks_CellsAncestry';
displayancestry.FunctionArgs.TracksLayout.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
displayancestry.FunctionArgs.AncestryLayout.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.AncestryLayout.InputArg='LoadAncestryLayout_AncestryLayout';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,displayancestry);

imageoverlayloop.InstanceName='ImageOverlayLoop';
imageoverlayloop.FunctionHandle=@forLoop;
imageoverlayloop.FunctionArgs.StartLoop.Value=StartFrame;
imageoverlayloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
imageoverlayloop.FunctionArgs.IncrementLoop.Value=FrameStep;
imageoverlayloop.FunctionArgs.RemoveShortTracks_Tracks.FunctionInstance='RemoveShortTracks';
imageoverlayloop.FunctionArgs.RemoveShortTracks_Tracks.OutputArg='Tracks';
imageoverlayloop.FunctionArgs.RemoveShortTracks_CellsAncestry.FunctionInstance='RemoveShortTracks
';
imageoverlayloop.FunctionArgs.RemoveShortTracks_CellsAncestry.OutputArg='CellsAncestry';
imageoverlayloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='LoadTracksLayout';
imageoverlayloop.FunctionArgs.LoadTracksLayout_TracksLayout.OutputArg='TracksLayout';
imageoverlayloop.FunctionArgs.LoadAncestryLayout_AncestryLayout.FunctionInstance='LoadAncestryLay
out';
imageoverlayloop.FunctionArgs.LoadAncestryLayout_AncestryLayout.OutputArg='AncestryLayout';
imageoverlayloop.LoopFunctions=image_overlay_loop_functions;
```

```
functions_list=addToFunctionChain(functions_list,imageoverlayloop);

saveancestryspreadsheets.InstanceName='SaveAncestrySpreadsheets';
saveancestryspreadsheets.FunctionHandle=@saveAncestrySpreadsheets;
saveancestryspreadsheets.FunctionArgs.ShapesXlsFile.Value=ShapesSpreadsheet;
saveancestryspreadsheets.FunctionArgs.ProlXlsFile.Value=AncestrySpreadsheet;
saveancestryspreadsheets.FunctionArgs.Tracks.FunctionInstance='RemoveShortTracks';
saveancestryspreadsheets.FunctionArgs.Tracks.OutputArg='Tracks';
saveancestryspreadsheets.FunctionArgs.CellsAncestry.FunctionInstance='RemoveShortTracks';
saveancestryspreadsheets.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
saveancestryspreadsheets.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
saveancestryspreadsheets.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
functions_list=addToFunctionChain(functions_list,saveancestryspreadsheets);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayBrightFieldCytoTestNN.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayBrightFieldCytoTestNN()
%assayBrightFieldCytoTestNN - This assay has been optimized for tracking cells imaged using
bright-field microscopy using
%  a nearest-neighbor  algorithm. The assay tracks the cells, detects mitotic events
%and  records  tracking data, cell  shape parameters and ancestry information to
%spreadsheets. For  each tracked  image, an overlayed image  is saved to
%disk that displays  the detected cell  outlines, cell IDs and cell
%generation. ImageFolder  - String variable that  specifies the absolute location of the
%directory which contains  the  time-lapse images. An  example of such a
%string variable would be  'c:/sample images/high-density'. ImageFilesRoot - String variable
specifying the
%root image file name. The  root image file name  for a
%set of images is the image  file name of any of the
%images  without the number or the  file extension. For example, if the
%file name   is 'Experiment-0002_Position(8)_t021.tif' the root  image file name will be
%'Experiment-0002_Position(8)_t'. ImageExtension - String variable  specifying the image  file
extension including the
%preceding dot. For example  if the  file name  is 'image003.jpg' the
%image extension is '.jpg'. StartFrame - Number specifying the  first  image in
%the sequence to be analyzed. The minimum  value for this   variable
%depends on the numbering of the image sequence so if  the first
% image in the sequence is 'image003.tif' then the minimum value is 3. FrameCount
% -  Number specifying how many images from the image sequence should be
%processed.  TimeFrame -  Number specifying the time between consecutive images in minutes.
%FrameStep -  Number specifying the  step size when reading images. Set this
%variable to 1   to read every  image in the sequence, 2
%to read every other  image and  so on.  NumberFormat - String
%value specifying the number of  digits in the image file names  in
% the sequence. For example if  the image file name is 'image020.jpg' the
% value for  the NumberFormat is  '%03d', while if the file name
%is 'image000020.jpg'  the value should  be  '%06d'. MaxFramesMissing - Number specifying
%for how many frames  a cell may be  disappear before its
%track is ended. OutputFolder - The  folder where the  overlayed images and
%track data will be saved. By  default  this value  is set
%to a folder named 'output' within the folder where   the  images
%to be analyzed are located. AncestryFolder - The folder where the overlayed
%images and ancestry data will be saved. By  default this value is set
%  to a folder named 'ancestry' within the output folder. AncestrySpreadsheet - The
%path  name  to the spreadsheet containing the ancestry data. By default this
% value  is set  to a file named 'ancestry.csv' within the ancestry
%folder. ShapesSpreadsheet -  The path name  to the spreadsheet containing the position
%and shape properties for   each cell in  the timelapse sequence at
%every time point. By default  this is  set to  to a
%file named 'shapes.csv' within the ancestry  folder. TracksFolder - The folder where
%the label matrixes containing the cell outlines  are saved. By  default this
%value  is set to a folder named  'track' within the output folder.
%SegmentationFilesRoot - The  root file name of the  label matrixes containing the
%cell outlines. ImageFileBase - The  path name to the  images. This value
%is generated from the ImageFolder  and  the ImageFilesRoot and  should not
%be changed. GradientThreshold - Number specifying the threshold value  for the  image
%generated by the generateBinImgUsingGradient filter.  Any pixels in the gradient  image
%below this value will be set to zero  while the rest will
% be set to one. ObjectArea - Number specifying the threshold area for the
```

```
%areaFilterLabel,   clearSmallObjects, segmentObjectsUsingClusters filters. Objects below  this
value will be removed
%from the  filtered  image. ObjectReduce - Number specifying how much a cluster
%of objects should  be reduced  before it  is processed by the
%segmentObjectsUsingClusters module. By default  this value is  set to 1  and
%it should not be changed  unless segmentObjectsUsingClusters throws an  out-of-memory error. In
%that  case the value can  be reduced to something lower  than
%1. ClusterDist - Number specifying the height  at which the hierarchical cluster tree
% will be cut  to determine the  number of clusters. Setting this
%value higher  results in a cluster  being  split into fewer objects
%by the segmentObjectsUsingClusters module.  ClearBorder - Boolean value specifying  whether
objects next
%to or touching the image border  in  the binary  images generated
%by the generateBinImgUsingGradient module will be erased (true)  or not   (false).
%ClearBorderDist - Number specifying how close to the border objects  may be
%and still be  erased if the ClearBorder parameter is set to  true
% in the generateBinImgUsingGradient module. ResizeImageScale - Number specifying by what ratio
the images
%  will be resized before they are  processed. By default this value
%is set   to 0.5. Used by the resizeImage module. ApproximationDistance - Number
%specifying how close  the  convex hull in the getConvexObjects module approximates the
% object outline. By  default this  value is set to 2.5. Setting
%it to a  lower  value will result  in convex hulls that
%more closely resemble the object outlines   however this increases  the chance
%of detecting insignificant concavities. MaxMergeDistance - Number  specifying the maximum
distance that
%one track may be from another track   for the duration and still
%be  considered for possible merging with the other  track.  Used by
%detectMergeCandidatesUsingDistance module. MaxSplitArea  - Number specifying the maximum area  a
nucleus may
%be and still be considered   as a part of  a possible
%mitotic event. Used by detectMitoticEvents module. MaxSplitDistance  - Number specifying the
maximum
%distance a new nucleus may be from another nucleus   and still
%be considered as part of a possible mitotic event. Used by  detectMitoticEvents
% module. MinSplitEccentricity - Number specifying the minimum eccentricity a new nucleus may
%have  and still be  considered as part of a possible mitotic event.
%Used   by detectMitoticEvents module. MaxSplitEccentricity - Number specifying the maximum
eccentricity a
%new nucleus may   have and still be  considered as part of
%a possible mitotic event.  Used  by detectMitoticEvents module. MinTimeForSplit - Number
specifying
%the minimum time in minutes  a track  needs to exist before
%it is considered for a possible  mitotic event. Used  by detectMitoticEvents module.
%MinLifespan - Number specifying the minimum length  in frames a frame  has
%to be to  not be removed by  the removeShortTracks module.  Important
% Modules areaFilterLabel, assignCellToTrackUsingNN, clearSmallObjects,
detectMergeCandidatesUsingDistance, detectMitoticEvents, generateBinImgUsingGradient,
removeShortTracks,  segmentObjectsUsingClusters, segmentObjectsUsingMarkers, splitTracks.

global functions_list;
functions_list=[];
%script variables
ImageFolder='C:/kam/H929_No_Position(8)';
ImageFilesRoot='Experiment-0002_Position(8)_t';
ImageExtension='.jpg';
StartFrame=1;
FrameCount=10;
TimeFrame=15;
FrameStep=1;
NumberFormat='%03d';
MaxFramesMissing=0;
OutputFolder=[ImageFolder '/output'];
AncestryFolder=[OutputFolder '/ancestry'];
AncestrySpreadsheet=[AncestryFolder 'ancestry.csv'];
ShapesSpreadsheet=[AncestryFolder 'shapes.csv'];
TracksFolder=[OutputFolder '/track'];
SegmentationFilesRoot=[TracksFolder '/grayscale'];
ImageFileBase=[ImageFolder '/' ImageFilesRoot];
GradientThreshold=1500;
ObjectArea=200;
ObjectReduce=1;
ClusterDist=21.5;
ClearBorder=true;
ClearBorderDist=2;
ResizeImageScale=0.5;
ApproximationDistance=2.5;
MaxMergeDistance=23;
MaxSplitArea=400;
MaxSplitDistance=45;
MinSplitEccentricity=0.5;
```

```
MaxSplitEccentricity=0.95;
MinTimeForSplit=900;
MinLifespan=30;
%end script variables

assign_cells_to_tracks_functions=[];
else_is_empty_cells_label_functions=[];
if_is_empty_cells_label_functions=[];
image_read_loop_functions=[];
image_overlay_loop_functions=[];

loadtrackslayout.InstanceName='LoadTracksLayout';
loadtrackslayout.FunctionHandle=@loadTracksLayout;
loadtrackslayout.FunctionArgs.FileName.Value='tracks_layout.mat';
functions_list=addToFunctionChain(functions_list,loadtrackslayout);

loadancestrylayout.InstanceName='LoadAncestryLayout';
loadancestrylayout.FunctionHandle=@loadAncestryLayout;
loadancestrylayout.FunctionArgs.FileName.Value='ancestry_layout.mat';
functions_list=addToFunctionChain(functions_list,loadancestrylayout);

displaycurframe.InstanceName='DisplayCurFrame';
displaycurframe.FunctionHandle=@displayVariable;
displaycurframe.FunctionArgs.VariableName.Value='Current Tracking Frame';
displaycurframe.FunctionArgs.Variable.FunctionInstance='SegmentationLoop';
displaycurframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,displaycurframe);

makeimagenamesinsegmentationloop.InstanceName='MakeImageNamesInSegmentationLoop';
makeimagenamesinsegmentationloop.FunctionHandle=@makeImgFileName;
makeimagenamesinsegmentationloop.FunctionArgs.FileBase.Value=ImageFileBase;
makeimagenamesinsegmentationloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinsegmentationloop.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makeimagenamesinsegmentati
onloop);

readimagesinsegmentationloop.InstanceName='ReadImagesInSegmentationLoop';
readimagesinsegmentationloop.FunctionHandle=@readImage;
readimagesinsegmentationloop.FunctionArgs.ImageChannel.Value='';
readimagesinsegmentationloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInSegmentatio
nLoop';
readimagesinsegmentationloop.FunctionArgs.ImageName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,readimagesinsegmentationlo
op);

normalizeimageto16bit.InstanceName='NormalizeImageTo16Bit';
normalizeimageto16bit.FunctionHandle=@imNorm;
normalizeimageto16bit.FunctionArgs.IntegerClass.Value='uint16';
normalizeimageto16bit.FunctionArgs.RawImage.FunctionInstance='ReadImagesInSegmentationLoop';
normalizeimageto16bit.FunctionArgs.RawImage.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,normalizeimageto16bit);

resizeimage.InstanceName='ResizeImage';
resizeimage.FunctionHandle=@resizeImage;
resizeimage.FunctionArgs.Scale.Value=ResizeImageScale;
resizeimage.FunctionArgs.Method.Value='bicubic';
resizeimage.FunctionArgs.Image.FunctionInstance='NormalizeImageTo16Bit';
resizeimage.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,resizeimage);

cytogradientlocalfilter.InstanceName='CytoGradientLocalFilter';
cytogradientlocalfilter.FunctionHandle=@generateBinImgUsingGradient;
cytogradientlocalfilter.FunctionArgs.GradientThreshold.Value=GradientThreshold;
cytogradientlocalfilter.FunctionArgs.ClearBorder.Value=ClearBorder;
cytogradientlocalfilter.FunctionArgs.ClearBorderDist.Value=ClearBorderDist;
cytogradientlocalfilter.FunctionArgs.Image.FunctionInstance='ResizeImage';
cytogradientlocalfilter.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,cytogradientlocalfilter);

fillholescytoplasmimages.InstanceName='FillHolesCytoplasmImages';
fillholescytoplasmimages.FunctionHandle=@fillHoles;
fillholescytoplasmimages.FunctionArgs.Image.FunctionInstance='CytoGradientLocalFilter';
fillholescytoplasmimages.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,fillholescytoplasmimages);

clearsmallcells.InstanceName='ClearSmallCells';
clearsmallcells.FunctionHandle=@clearSmallObjects;
clearsmallcells.FunctionArgs.MinObjectArea.Value=ObjectArea;
clearsmallcells.FunctionArgs.Image.FunctionInstance='FillHolesCytoplasmImages';
```

```
clearsmallcells.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,clearsmallcells);

nuclgradientlocalfilter.InstanceName='NuclGradientLocalFilter';
nuclgradientlocalfilter.FunctionHandle=@generateBinImgUsingGradient;
nuclgradientlocalfilter.FunctionArgs.GradientThreshold.Value=GradientThreshold;
nuclgradientlocalfilter.FunctionArgs.ClearBorder.Value=ClearBorder;
nuclgradientlocalfilter.FunctionArgs.ClearBorderDist.Value=ClearBorderDist;
nuclgradientlocalfilter.FunctionArgs.Image.FunctionInstance='ResizeImage';
nuclgradientlocalfilter.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,nuclgradientlocalfilter);

fillholesnuclearimages.InstanceName='FillHolesNuclearImages';
fillholesnuclearimages.FunctionHandle=@fillHoles;
fillholesnuclearimages.FunctionArgs.Image.FunctionInstance='NuclGradientLocalFilter';
fillholesnuclearimages.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,fillholesnuclearimages);

clearsmallnuclei.InstanceName='ClearSmallNuclei';
clearsmallnuclei.FunctionHandle=@clearSmallObjects;
clearsmallnuclei.FunctionArgs.MinObjectArea.Value=ObjectArea;
clearsmallnuclei.FunctionArgs.Image.FunctionInstance='FillHolesNuclearImages';
clearsmallnuclei.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,clearsmallnuclei);

combinenuclearandcytoplasmimages.InstanceName='CombineNuclearAndCytoplasmImages';
combinenuclearandcytoplasmimages.FunctionHandle=@combineImages;
combinenuclearandcytoplasmimages.FunctionArgs.CombineOperation.Value='AND';
combinenuclearandcytoplasmimages.FunctionArgs.Image1.FunctionInstance='ClearSmallNuclei';
combinenuclearandcytoplasmimages.FunctionArgs.Image1.OutputArg='Image';
combinenuclearandcytoplasmimages.FunctionArgs.Image2.FunctionInstance='ClearSmallCells';
combinenuclearandcytoplasmimages.FunctionArgs.Image2.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,combinenuclearandcytoplasm
images);

reconstructcytoplasmimage.InstanceName='ReconstructCytoplasmImage';
reconstructcytoplasmimage.FunctionHandle=@reconstructObjects;
reconstructcytoplasmimage.FunctionArgs.GuideImage.FunctionInstance='CombineNuclearAndCytoplasmIma
ges';
reconstructcytoplasmimage.FunctionArgs.GuideImage.OutputArg='Image';
reconstructcytoplasmimage.FunctionArgs.ImageToReconstruct.FunctionInstance='ClearSmallNuclei';
reconstructcytoplasmimage.FunctionArgs.ImageToReconstruct.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,reconstructcytoplasmimage)
;

labelnuclei.InstanceName='LabelNuclei';
labelnuclei.FunctionHandle=@labelObjects;
labelnuclei.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
labelnuclei.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,labelnuclei);

labelcytoplasm.InstanceName='LabelCytoplasm';
labelcytoplasm.FunctionHandle=@labelObjects;
labelcytoplasm.FunctionArgs.Image.FunctionInstance='ReconstructCytoplasmImage';
labelcytoplasm.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,labelcytoplasm);

getconvexobjects.InstanceName='GetConvexObjects';
getconvexobjects.FunctionHandle=@getConvexObjects;
getconvexobjects.FunctionArgs.ApproximationDistance.Value=ApproximationDistance;
getconvexobjects.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
getconvexobjects.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getconvexobjects);

segmentobjectsusingclusters.InstanceName='SegmentObjectsUsingClusters';
segmentobjectsusingclusters.FunctionHandle=@segmentObjectsUsingClusters;
segmentobjectsusingclusters.FunctionArgs.ObjectReduce.Value=ObjectReduce;
segmentobjectsusingclusters.FunctionArgs.MinimumObjectArea.Value=ObjectArea;
segmentobjectsusingclusters.FunctionArgs.ClusterDistance.Value=ClusterDist;
segmentobjectsusingclusters.FunctionArgs.ObjectsLabel.FunctionInstance='LabelNuclei';
segmentobjectsusingclusters.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,segmentobjectsusingcluster
s);

segmentobjectsusingmarkers.InstanceName='SegmentObjectsUsingMarkers';
segmentobjectsusingmarkers.FunctionHandle=@segmentObjectsUsingMarkers;
segmentobjectsusingmarkers.FunctionArgs.MarkersLabel.FunctionInstance='SegmentObjectsUsingCluster
s';
segmentobjectsusingmarkers.FunctionArgs.MarkersLabel.OutputArg='LabelMatrix';
segmentobjectsusingmarkers.FunctionArgs.ObjectsLabel.FunctionInstance='LabelCytoplasm';
segmentobjectsusingmarkers.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
```

```
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,segmentobjectsusingmarkers
);

areafilter.InstanceName='AreaFilter';
areafilter.FunctionHandle=@areaFilterLabel;
areafilter.FunctionArgs.MinArea.Value=ObjectArea;
areafilter.FunctionArgs.ObjectsLabel.FunctionInstance='SegmentObjectsUsingMarkers';
areafilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,areafilter);

resizecytolabel.InstanceName='ResizeCytoLabel';
resizecytolabel.FunctionHandle=@resizeImage;
resizecytolabel.FunctionArgs.Scale.Value=1/ResizeImageScale;
resizecytolabel.FunctionArgs.Method.Value='nearest';
resizecytolabel.FunctionArgs.Image.FunctionInstance='AreaFilter';
resizecytolabel.FunctionArgs.Image.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,resizecytolabel);

getshapeparameters.InstanceName='GetShapeParameters';
getshapeparameters.FunctionHandle=@getShapeParams;
getshapeparameters.FunctionArgs.LabelMatrix.FunctionInstance='ResizeCytoLabel';
getshapeparameters.FunctionArgs.LabelMatrix.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getshapeparameters);

isemptypreviouscellslabel.InstanceName='IsEmptyPreviousCellsLabel';
isemptypreviouscellslabel.FunctionHandle=@isEmptyFunction;
isemptypreviouscellslabel.FunctionArgs.TestVariable.Value=[];
isemptypreviouscellslabel.FunctionArgs.TestVariable.FunctionInstance='SaveCellsLabel';
isemptypreviouscellslabel.FunctionArgs.TestVariable.OutputArg='CellsLabel';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,isemptypreviouscellslabel)
;

starttracks.InstanceName='StartTracks';
starttracks.FunctionHandle=@startTracks;
starttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
starttracks.FunctionArgs.ShapeParameters.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.ShapeParameters.InputArg='GetShapeParameters_ShapeParameters';
starttracks.FunctionArgs.CellsLabel.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.CellsLabel.InputArg='ResizeCytoLabel_Image';
starttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
if_is_empty_cells_label_functions=addToFunctionChain(if_is_empty_cells_label_functions,starttrack
s);

getcurrenttracks.InstanceName='GetCurrentTracks';
getcurrenttracks.FunctionHandle=@getCurrentTracks;
getcurrenttracks.FunctionArgs.OffsetFrame.Value=-1;
getcurrenttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getcurrenttracks.FunctionArgs.TimeCol.Value=2;
getcurrenttracks.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getcurrenttracks.FunctionArgs.FrameStep.Value=FrameStep;
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg='Tracks';
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getcurrenttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getcurrenttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getcur
renttracks);

getprevioustracks.InstanceName='GetPreviousTracks';
getprevioustracks.FunctionHandle=@getCurrentTracks;
getprevioustracks.FunctionArgs.OffsetFrame.Value=-2;
getprevioustracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getprevioustracks.FunctionArgs.TimeCol.Value=2;
getprevioustracks.FunctionArgs.TrackIDCol.Value=1;
getprevioustracks.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getprevioustracks.FunctionArgs.FrameStep.Value=FrameStep;
getprevioustracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg='Tracks';
getprevioustracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getprevioustracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getprevioustracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getpre
vioustracks);

makeunassignedcellslist.InstanceName='MakeUnassignedCellsList';
makeunassignedcellslist.FunctionHandle=@makeUnassignedCellsList;
```

```
makeunassignedcellslist.FunctionArgs.CellsCentroids.FunctionInstance='IfIsEmptyPreviousCellsLabel
';
makeunassignedcellslist.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,makeun
assignedcellslist);

makeexcludedtrackslist.InstanceName='MakeExcludedTracksList';
makeexcludedtrackslist.FunctionHandle=@makeExcludedTracksList;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.FunctionInstance='MakeUnassignedCellsList'
;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.OutputArg='UnassignedCellsIDs';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,makeex
cludedtrackslist);

getcellsmeandisplacement.InstanceName='GetCellsMeanDisplacement';
getcellsmeandisplacement.FunctionHandle=@getObjectsMeanDisplacement;
getcellsmeandisplacement.FunctionArgs.Centroid1Col.Value=3;
getcellsmeandisplacement.FunctionArgs.Centroid2Col.Value=4;
getcellsmeandisplacement.FunctionArgs.CurrentTracks.FunctionInstance='GetCurrentTracks';
getcellsmeandisplacement.FunctionArgs.CurrentTracks.OutputArg='Tracks';
getcellsmeandisplacement.FunctionArgs.ObjectCentroids.FunctionInstance='IfIsEmptyPreviousCellsLab
el';
getcellsmeandisplacement.FunctionArgs.ObjectCentroids.InputArg='GetShapeParameters_Centroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getcel
lsmeandisplacement);

getparamscoefficientofvariation.InstanceName='GetParamsCoefficientOfVariation';
getparamscoefficientofvariation.FunctionHandle=@getParamsCoefficientOfVariation;
getparamscoefficientofvariation.FunctionArgs.AreaCol.Value=5;
getparamscoefficientofvariation.FunctionArgs.SolidityCol.Value=11;
getparamscoefficientofvariation.FunctionArgs.Params.FunctionInstance='IfIsEmptyPreviousCellsLabel
';
getparamscoefficientofvariation.FunctionArgs.Params.InputArg='GetShapeParameters_ShapeParameters'
;
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getpar
amscoefficientofvariation);

getmaxtrackid.InstanceName='GetMaxTrackID';
getmaxtrackid.FunctionHandle=@getMaxTrackID;
getmaxtrackid.FunctionArgs.TrackIDCol.Value=1;
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg='Tracks';
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg2='Tracks';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getmax
trackid);

isnotemptyunassignedcells.InstanceName='IsNotEmptyUnassignedCells';
isnotemptyunassignedcells.FunctionHandle=@isNotEmptyFunction;
isnotemptyunassignedcells.FunctionArgs.TestVariable.FunctionInstance='AssignCellToTrackUsingAll';
isnotemptyunassignedcells.FunctionArgs.TestVariable.OutputArg='UnassignedIDs';
isnotemptyunassignedcells.FunctionArgs.TestVariable.FunctionInstance2='AssignCellsToTracksLoop';
isnotemptyunassignedcells.FunctionArgs.TestVariable.InputArg2='MakeUnassignedCellsList_Unassigned
CellsIDs';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,isnotemptyun
assignedcells);

getcurrentunassignedcell.InstanceName='GetCurrentUnassignedCell';
getcurrentunassignedcell.FunctionHandle=@getCurrentUnassignedCell;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance='AssignCellToTrackUsingAll
';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTracksLoop'
;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassign
edCellsIDs';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,getcurrentun
assignedcell);

assigncelltotrackusingall.InstanceName='AssignCellToTrackUsingAll';
assigncelltotrackusingall.FunctionHandle=@assignCellToTrackUsingNN;
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.Value=[];
assigncelltotrackusingall.FunctionArgs.TrackAssignments.Value=[];
assigncelltotrackusingall.FunctionArgs.Tracks.Value=[];
assigncelltotrackusingall.FunctionArgs.UnassignedCells.FunctionInstance='AssignCellToTrackUsingAl
l';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.FunctionInstance='AssignCellToTrackUsingAll
';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.OutputArg='ExcludedTracks';
```

```
assigncelltotrackusingall.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellToTrackUsingA
ll';
assigncelltotrackusingall.FunctionArgs.TrackAssignments.OutputArg='TrackAssignments';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTracksLoop
';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassig
nedCellsIDs';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.FunctionInstance2='AssignCellsToTracksLoop'
;
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.InputArg2='MakeExcludedTracksList_ExcludedT
racks';
assigncelltotrackusingall.FunctionArgs.CellsLabel.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CellsLabel.InputArg='ResizeCytoLabel_Image';
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.FunctionInstance='AssignCellsToTracksLo
op';
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.InputArg='SaveCellsLabel_CellsLabel';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.FunctionInstance='AssignCellsToTracksLoop'
;
assigncelltotrackusingall.FunctionArgs.ShapeParameters.InputArg='GetShapeParameters_ShapeParamete
rs';
assigncelltotrackusingall.FunctionArgs.CellsCentroids.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
assigncelltotrackusingall.FunctionArgs.CurrentTracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CurrentTracks.InputArg='GetCurrentTracks_Tracks';
assigncelltotrackusingall.FunctionArgs.MaxTrackID.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.MaxTrackID.InputArg='GetMaxTrackID_MaxTrackID';
assigncelltotrackusingall.FunctionArgs.Tracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.Tracks.InputArg='SegmentationLoop_Tracks';
assigncelltotrackusingall.FunctionArgs.PreviousTracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.PreviousTracks.InputArg='GetPreviousTracks_Tracks';
assigncelltotrackusingall.FunctionArgs.TracksLayout.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,assigncellto
trackusingall);

assigncellstotracksloop.InstanceName='AssignCellsToTracksLoop';
assigncellstotracksloop.FunctionHandle=@whileLoop;
assigncellstotracksloop.FunctionArgs.TestFunction.FunctionInstance='IsNotEmptyUnassignedCells';
assigncellstotracksloop.FunctionArgs.TestFunction.OutputArg='Boolean';
assigncellstotracksloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.FunctionInstance=
'MakeUnassignedCellsList';
assigncellstotracksloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.OutputArg='Unassi
gnedCellsIDs';
assigncellstotracksloop.FunctionArgs.MakeExcludedTracksList_ExcludedTracks.FunctionInstance='Make
ExcludedTracksList';
assigncellstotracksloop.FunctionArgs.MakeExcludedTracksList_ExcludedTracks.OutputArg='ExcludedTra
cks';
assigncellstotracksloop.FunctionArgs.GetCurrentTracks_Tracks.FunctionInstance='GetCurrentTracks';
assigncellstotracksloop.FunctionArgs.GetCurrentTracks_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.GetMaxTrackID_MaxTrackID.FunctionInstance='GetMaxTrackID';
assigncellstotracksloop.FunctionArgs.GetMaxTrackID_MaxTrackID.OutputArg='MaxTrackID';
assigncellstotracksloop.FunctionArgs.GetPreviousTracks_Tracks.FunctionInstance='GetPreviousTracks
';
assigncellstotracksloop.FunctionArgs.GetPreviousTracks_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.ResizeCytoLabel_Image.FunctionInstance='IfIsEmptyPreviousCel
lsLabel';
assigncellstotracksloop.FunctionArgs.ResizeCytoLabel_Image.InputArg='ResizeCytoLabel_Image';
assigncellstotracksloop.FunctionArgs.SaveCellsLabel_CellsLabel.FunctionInstance='IfIsEmptyPreviou
sCellsLabel';
assigncellstotracksloop.FunctionArgs.SaveCellsLabel_CellsLabel.InputArg='SaveCellsLabel_CellsLabe
l';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_ShapeParameters.FunctionInstance='IfIsEmp
tyPreviousCellsLabel';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_ShapeParameters.InputArg='GetShapeParamet
ers_ShapeParameters';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_Centroids.FunctionInstance='IfIsEmptyPrev
iousCellsLabel';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_Centroids.InputArg='GetShapeParameters_Ce
ntroids';
assigncellstotracksloop.FunctionArgs.SegmentationLoop_Tracks.FunctionInstance='IfIsEmptyPreviousC
ellsLabel';
assigncellstotracksloop.FunctionArgs.SegmentationLoop_Tracks.InputArg='SegmentationLoop_Tracks';
assigncellstotracksloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='IfIsEmptyPre
viousCellsLabel';
assigncellstotracksloop.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout_Tra
cksLayout';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_TrackAssignments.FunctionInstance='A
ssignCellToTrackUsingAll';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_TrackAssignments.OutputArg='TrackAss
ignments';
assigncellstotracksloop.LoopFunctions=assign_cells_to_tracks_functions;
```

```
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,assign
cellstotracksloop);

continuetracks.InstanceName='ContinueTracks';
continuetracks.FunctionHandle=@continueTracks;
continuetracks.FunctionArgs.TimeFrame.Value=TimeFrame;
continuetracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
continuetracks.FunctionArgs.Tracks.OutputArg='Tracks';
continuetracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
continuetracks.FunctionArgs.Tracks.OutputArg2='Tracks';
continuetracks.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellsToTracksLoop';
continuetracks.FunctionArgs.TrackAssignments.OutputArg='AssignCellToTrackUsingAll_TrackAssignment
s';
continuetracks.FunctionArgs.CellsCentroids.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
continuetracks.FunctionArgs.ShapeParameters.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.ShapeParameters.InputArg='GetShapeParameters_ShapeParameters';
continuetracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,contin
uetracks);

ifisemptypreviouscellslabel.InstanceName='IfIsEmptyPreviousCellsLabel';
ifisemptypreviouscellslabel.FunctionHandle=@if_statement;
ifisemptypreviouscellslabel.FunctionArgs.PreviousCellsLabel.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.Tracks.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.TrackAssignments.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.TestVariable.FunctionInstance='IsEmptyPreviousCellsLabel
';
ifisemptypreviouscellslabel.FunctionArgs.TestVariable.OutputArg='Boolean';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_ShapeParameters.FunctionInstance='Get
ShapeParameters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_ShapeParameters.OutputArg='ShapeParam
eters';
ifisemptypreviouscellslabel.FunctionArgs.ResizeCytoLabel_Image.FunctionInstance='ResizeCytoLabel'
;
ifisemptypreviouscellslabel.FunctionArgs.ResizeCytoLabel_Image.OutputArg='Image';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_LoopCounter.FunctionInstance='Segmentat
ionLoop';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_LoopCounter.OutputArg='LoopCounter';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_Centroids.FunctionInstance='GetShapeP
arameters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_Centroids.OutputArg='Centroids';
ifisemptypreviouscellslabel.FunctionArgs.SaveCellsLabel_CellsLabel.FunctionInstance='SaveCellsLab
el';
ifisemptypreviouscellslabel.FunctionArgs.SaveCellsLabel_CellsLabel.OutputArg='CellsLabel';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_Tracks.FunctionInstance='SegmentationLo
op';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_Tracks.OutputArg='Tracks';
ifisemptypreviouscellslabel.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='Segmenta
tionLoop';
ifisemptypreviouscellslabel.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout
_TracksLayout';
ifisemptypreviouscellslabel.KeepValues.ContinueTracks_Tracks.FunctionInstance='ContinueTracks';
ifisemptypreviouscellslabel.KeepValues.ContinueTracks_Tracks.OutputArg='Tracks';
ifisemptypreviouscellslabel.ElseFunctions=else_is_empty_cells_label_functions;
ifisemptypreviouscellslabel.IfFunctions=if_is_empty_cells_label_functions;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,ifisemptypreviouscellslabe
l);

savecellslabel.InstanceName='SaveCellsLabel';
savecellslabel.FunctionHandle=@saveCellsLabel;
savecellslabel.FunctionArgs.FileRoot.Value=SegmentationFilesRoot;
savecellslabel.FunctionArgs.NumberFormat.Value=NumberFormat;
savecellslabel.FunctionArgs.CellsLabel.FunctionInstance='ResizeCytoLabel';
savecellslabel.FunctionArgs.CellsLabel.OutputArg='Image';
savecellslabel.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
savecellslabel.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,savecellslabel);

segmentationloop.InstanceName='SegmentationLoop';
segmentationloop.FunctionHandle=@forLoop;
segmentationloop.FunctionArgs.StartLoop.Value=StartFrame;
segmentationloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
segmentationloop.FunctionArgs.IncrementLoop.Value=FrameStep;
segmentationloop.FunctionArgs.Tracks.Value=[];
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='LoadTracksLayout';
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.OutputArg='TracksLayout';
segmentationloop.KeepValues.ContinueTracks_Tracks.FunctionInstance='IfIsEmptyPreviousCellsLabel';
segmentationloop.KeepValues.ContinueTracks_Tracks.OutputArg='ContinueTracks_Tracks';
segmentationloop.LoopFunctions=image_read_loop_functions;
```

```
functions_list=addToFunctionChain(functions_list,segmentationloop);

savetracks.InstanceName='SaveTracks';
savetracks.FunctionHandle=@saveTracks;
savetracks.FunctionArgs.TracksFileName.Value=[TracksFolder '/tracks.mat'];
savetracks.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
savetracks.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,savetracks);

gettrackids.InstanceName='GetTrackIDs';
gettrackids.FunctionHandle=@getTrackIDs;
gettrackids.FunctionArgs.TrackIDCol.Value=1;
gettrackids.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
gettrackids.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,gettrackids);

detectmergecandidates.InstanceName='DetectMergeCandidates';
detectmergecandidates.FunctionHandle=@detectMergeCandidatesUsingDistance;
detectmergecandidates.FunctionArgs.MaxMergeDistance.Value=MaxMergeDistance;
detectmergecandidates.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDs';
detectmergecandidates.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
detectmergecandidates.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
detectmergecandidates.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
detectmergecandidates.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
detectmergecandidates.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,detectmergecandidates);

mergetracks.InstanceName='MergeTracks';
mergetracks.FunctionHandle=@mergeTracks;
mergetracks.FunctionArgs.FrameCount.Value=FrameCount;
mergetracks.FunctionArgs.StartFrame.Value=StartFrame;
mergetracks.FunctionArgs.TimeFrame.Value=TimeFrame;
mergetracks.FunctionArgs.SegFileRoot.Value=SegmentationFilesRoot;
mergetracks.FunctionArgs.FrameStep.Value=FrameStep;
mergetracks.FunctionArgs.NumberFormat.Value=NumberFormat;
mergetracks.FunctionArgs.TracksToBeMerged.FunctionInstance='DetectMergeCandidates';
mergetracks.FunctionArgs.TracksToBeMerged.OutputArg='TracksToBeMerged';
mergetracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
mergetracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
mergetracks.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
mergetracks.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,mergetracks);

gettrackidsaftermerge.InstanceName='GetTrackIDsAfterMerge';
gettrackidsaftermerge.FunctionHandle=@getTrackIDs;
gettrackidsaftermerge.FunctionArgs.TrackIDCol.Value=1;
gettrackidsaftermerge.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
gettrackidsaftermerge.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,gettrackidsaftermerge);

makeancestryforfirstframecells.InstanceName='MakeAncestryForFirstFrameCells';
makeancestryforfirstframecells.FunctionHandle=@makeAncestryForFirstFrameCells;
makeancestryforfirstframecells.FunctionArgs.TimeCol.Value=2;
makeancestryforfirstframecells.FunctionArgs.TrackIDCol.Value=1;
makeancestryforfirstframecells.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
makeancestryforfirstframecells.FunctionArgs.Tracks.OutputArg='Tracks';
makeancestryforfirstframecells.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDsAfterMerge';
makeancestryforfirstframecells.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
functions_list=addToFunctionChain(functions_list,makeancestryforfirstframecells);

detectmitoticevents.InstanceName='DetectMitoticEvents';
detectmitoticevents.FunctionHandle=@detectMitoticEvents;
detectmitoticevents.FunctionArgs.MaxSplitArea.Value=MaxSplitArea;
detectmitoticevents.FunctionArgs.MinSplitEccentricity.Value=MinSplitEccentricity;
detectmitoticevents.FunctionArgs.MaxSplitEccentricity.Value=MaxSplitEccentricity;
detectmitoticevents.FunctionArgs.MaxSplitDistance.Value=MaxSplitDistance;
detectmitoticevents.FunctionArgs.MinTimeForSplit.Value=MinTimeForSplit;
detectmitoticevents.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
detectmitoticevents.FunctionArgs.Tracks.OutputArg='Tracks';
detectmitoticevents.FunctionArgs.UntestedIDs.FunctionInstance='MakeAncestryForFirstFrameCells';
detectmitoticevents.FunctionArgs.UntestedIDs.OutputArg='UntestedIDs';
detectmitoticevents.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
detectmitoticevents.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
functions_list=addToFunctionChain(functions_list,detectmitoticevents);

makeancestryforcellsenteringframes.InstanceName='MakeAncestryForCellsEnteringFrames';
makeancestryforcellsenteringframes.FunctionHandle=@makeAncestryForCellsEnteringFrames;
makeancestryforcellsenteringframes.FunctionArgs.TimeCol.Value=2;
makeancestryforcellsenteringframes.FunctionArgs.TrackIDCol.Value=1;
makeancestryforcellsenteringframes.FunctionArgs.SplitCells.FunctionInstance='DetectMitoticEvents'
;
```

```
makeancestryforcellsenteringframes.FunctionArgs.SplitCells.OutputArg='SplitCells';
makeancestryforcellsenteringframes.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDsAfterMerge'
;
makeancestryforcellsenteringframes.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
makeancestryforcellsenteringframes.FunctionArgs.FirstFrameIDs.FunctionInstance='MakeAncestryForFi
rstFrameCells';
makeancestryforcellsenteringframes.FunctionArgs.FirstFrameIDs.OutputArg='FirstFrameIDs';
makeancestryforcellsenteringframes.FunctionArgs.CellsAncestry.FunctionInstance='MakeAncestryForFi
rstFrameCells';
makeancestryforcellsenteringframes.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
makeancestryforcellsenteringframes.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
makeancestryforcellsenteringframes.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,makeancestryforcellsenteringframes);

splittracks.InstanceName='SplitTracks';
splittracks.FunctionHandle=@splitTracks;
splittracks.FunctionArgs.TimeFrame.Value=TimeFrame;
splittracks.FunctionArgs.SplitCells.FunctionInstance='DetectMitoticEvents';
splittracks.FunctionArgs.SplitCells.OutputArg='SplitCells';
splittracks.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
splittracks.FunctionArgs.Tracks.OutputArg='Tracks';
splittracks.FunctionArgs.CellsAncestry.FunctionInstance='MakeAncestryForCellsEnteringFrames';
splittracks.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
splittracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
splittracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
splittracks.FunctionArgs.AncestryLayout.FunctionInstance='LoadAncestryLayout';
splittracks.FunctionArgs.AncestryLayout.OutputArg='AncestryLayout';
functions_list=addToFunctionChain(functions_list,splittracks);

removeshorttracks.InstanceName='RemoveShortTracks';
removeshorttracks.FunctionHandle=@removeShortTracks;
removeshorttracks.FunctionArgs.MinLifespan.Value=MinLifespan;
removeshorttracks.FunctionArgs.Tracks.FunctionInstance='SplitTracks';
removeshorttracks.FunctionArgs.Tracks.OutputArg='Tracks';
removeshorttracks.FunctionArgs.CellsAncestry.FunctionInstance='SplitTracks';
removeshorttracks.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
removeshorttracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
removeshorttracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
removeshorttracks.FunctionArgs.AncestryLayout.FunctionInstance='LoadAncestryLayout';
removeshorttracks.FunctionArgs.AncestryLayout.OutputArg='AncestryLayout';
functions_list=addToFunctionChain(functions_list,removeshorttracks);

saveupdatedtracks.InstanceName='SaveUpdatedTracks';
saveupdatedtracks.FunctionHandle=@saveTracks;
saveupdatedtracks.FunctionArgs.TracksFileName.Value=[AncestryFolder '/tracks.mat'];
saveupdatedtracks.FunctionArgs.Tracks.FunctionInstance='RemoveShortTracks';
saveupdatedtracks.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,saveupdatedtracks);

saveancestry.InstanceName='SaveAncestry';
saveancestry.FunctionHandle=@saveAncestry;
saveancestry.FunctionArgs.AncestryFileName.Value=[AncestryFolder '/ancestry.mat'];
saveancestry.FunctionArgs.CellsAncestry.FunctionInstance='RemoveShortTracks';
saveancestry.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
functions_list=addToFunctionChain(functions_list,saveancestry);

makeimagenamesinoverlayloop.InstanceName='MakeImageNamesInOverlayLoop';
makeimagenamesinoverlayloop.FunctionHandle=@makeImgFileName;
makeimagenamesinoverlayloop.FunctionArgs.FileBase.Value=ImageFileBase;
makeimagenamesinoverlayloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinoverlayloop.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenamesinoverlayloop.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
makeimagenamesinoverlayloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,makeimagenamesinover
layloop);

readimagesinoverlayloop.InstanceName='ReadImagesInOverlayLoop';
readimagesinoverlayloop.FunctionHandle=@readImage;
readimagesinoverlayloop.FunctionArgs.ImageChannel.Value='';
readimagesinoverlayloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInOverlayLoop';
readimagesinoverlayloop.FunctionArgs.ImageName.OutputArg='FileName';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,readimagesinoverlayl
oop);

getcurrenttracks2.InstanceName='GetCurrentTracks2';
getcurrenttracks2.FunctionHandle=@getCurrentTracks;
getcurrenttracks2.FunctionArgs.OffsetFrame.Value=0;
getcurrenttracks2.FunctionArgs.TimeFrame.Value=TimeFrame;
getcurrenttracks2.FunctionArgs.TimeCol.Value=2;
getcurrenttracks2.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks2.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
```

```
getcurrenttracks2.FunctionArgs.FrameStep.Value=FrameStep;
getcurrenttracks2.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
getcurrenttracks2.FunctionArgs.CurFrame.OutputArg='LoopCounter';
getcurrenttracks2.FunctionArgs.Tracks.FunctionInstance='ImageOverlayLoop';
getcurrenttracks2.FunctionArgs.Tracks.InputArg='RemoveShortTracks_Tracks';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,getcurrenttracks2);

makematnamesinoverlayloop.InstanceName='MakeMatNamesInOverlayLoop';
makematnamesinoverlayloop.FunctionHandle=@makeImgFileName;
makematnamesinoverlayloop.FunctionArgs.FileBase.Value=SegmentationFilesRoot;
makematnamesinoverlayloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makematnamesinoverlayloop.FunctionArgs.FileExt.Value='.mat';
makematnamesinoverlayloop.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
makematnamesinoverlayloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,makematnamesinoverla
yloop);

loadcellslabel.InstanceName='LoadCellsLabel';
loadcellslabel.FunctionHandle=@loadCellsLabel;
loadcellslabel.FunctionArgs.FileName.FunctionInstance='MakeMatNamesInOverlayLoop';
loadcellslabel.FunctionArgs.FileName.OutputArg='FileName';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,loadcellslabel);

loadcolormap.InstanceName='LoadColormap';
loadcolormap.FunctionHandle=@loadColormap;
loadcolormap.FunctionArgs.FileName.Value='colormap_lines';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,loadcolormap);

displayoverlayingframe.InstanceName='DisplayOverlayingFrame';
displayoverlayingframe.FunctionHandle=@displayVariable;
displayoverlayingframe.FunctionArgs.VariableName.Value='Overlaying Frame';
displayoverlayingframe.FunctionArgs.Variable.FunctionInstance='ImageOverlayLoop';
displayoverlayingframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,displayoverlayingfra
me);

displayancestry.InstanceName='DisplayAncestry';
displayancestry.FunctionHandle=@displayAncestryData;
displayancestry.FunctionArgs.NumberFormat.Value=NumberFormat;
displayancestry.FunctionArgs.ProlDir.Value=AncestryFolder;
displayancestry.FunctionArgs.ImageFileName.Value=ImageFilesRoot;
displayancestry.FunctionArgs.DS.Value='/';
displayancestry.FunctionArgs.Image.FunctionInstance='ReadImagesInOverlayLoop';
displayancestry.FunctionArgs.Image.OutputArg='Image';
displayancestry.FunctionArgs.CurrentTracks.FunctionInstance='GetCurrentTracks2';
displayancestry.FunctionArgs.CurrentTracks.OutputArg='Tracks';
displayancestry.FunctionArgs.CellsLabel.FunctionInstance='LoadCellsLabel';
displayancestry.FunctionArgs.CellsLabel.OutputArg='LabelMatrix';
displayancestry.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.CurFrame.OutputArg='LoopCounter';
displayancestry.FunctionArgs.ColorMap.FunctionInstance='LoadColormap';
displayancestry.FunctionArgs.ColorMap.OutputArg='Colormap';
displayancestry.FunctionArgs.CellsAncestry.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.CellsAncestry.InputArg='RemoveShortTracks_CellsAncestry';
displayancestry.FunctionArgs.TracksLayout.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
displayancestry.FunctionArgs.AncestryLayout.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.AncestryLayout.InputArg='LoadAncestryLayout_AncestryLayout';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,displayancestry);

imageoverlayloop.InstanceName='ImageOverlayLoop';
imageoverlayloop.FunctionHandle=@forLoop;
imageoverlayloop.FunctionArgs.StartLoop.Value=StartFrame;
imageoverlayloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
imageoverlayloop.FunctionArgs.IncrementLoop.Value=FrameStep;
imageoverlayloop.FunctionArgs.RemoveShortTracks_Tracks.FunctionInstance='RemoveShortTracks';
imageoverlayloop.FunctionArgs.RemoveShortTracks_Tracks.OutputArg='Tracks';
imageoverlayloop.FunctionArgs.RemoveShortTracks_CellsAncestry.FunctionInstance='RemoveShortTracks
';
imageoverlayloop.FunctionArgs.RemoveShortTracks_CellsAncestry.OutputArg='CellsAncestry';
imageoverlayloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='LoadTracksLayout';
imageoverlayloop.FunctionArgs.LoadTracksLayout_TracksLayout.OutputArg='TracksLayout';
imageoverlayloop.FunctionArgs.LoadAncestryLayout_AncestryLayout.FunctionInstance='LoadAncestryLay
out';
imageoverlayloop.FunctionArgs.LoadAncestryLayout_AncestryLayout.OutputArg='AncestryLayout';
imageoverlayloop.LoopFunctions=image_overlay_loop_functions;
functions_list=addToFunctionChain(functions_list,imageoverlayloop);

saveancestryspreadsheets.InstanceName='SaveAncestrySpreadsheets';
saveancestryspreadsheets.FunctionHandle=@saveAncestrySpreadsheets;
saveancestryspreadsheets.FunctionArgs.ShapesXlsFile.Value=ShapesSpreadsheet;
```

```
saveancestryspreadsheets.FunctionArgs.ProlXlsFile.Value=AncestrySpreadsheet;
saveancestryspreadsheets.FunctionArgs.Tracks.FunctionInstance='RemoveShortTracks';
saveancestryspreadsheets.FunctionArgs.Tracks.OutputArg='Tracks';
saveancestryspreadsheets.FunctionArgs.CellsAncestry.FunctionInstance='RemoveShortTracks';
saveancestryspreadsheets.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
saveancestryspreadsheets.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
saveancestryspreadsheets.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
functions_list=addToFunctionChain(functions_list,saveancestryspreadsheets);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% assayBrightFieldCytoTestWG.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayBrightFieldCytoTestWG()
%assayBrightFieldCytoTestWG - This assay has been optimized for tracking cells imaged using
bright-field microscopy using
% a custom  algorithm. This algorithm is more accurate and flexible than the
%nearest-neighbor  algorithm at the  expense of execution speed. The assay tracks the
%cells, detects  mitotic events and records  tracking data, cell shape parameters and
%ancestry information to  spreadsheets. For each tracked image,  an overlayed image is
%saved to disk that  displays the detected cell outlines, cell  IDs and
%cell generation. ImageFolder  - String variable that specifies the absolute location of the
%directory which contains the   time-lapse images. An example of such a string
%variable would be 'c:/sample images/high-density'.  ImageFilesRoot - String variable specifying
the root image
%file name. The root image file  name  for a set of images
%is the image file name of any  of the  images without the
%number or the file extension. For example, if  the file name  is
%'Experiment-0002_Position(8)_t021.tif' the root image file name will be 'Experiment-
0002_Position(8)_t'.  ImageExtension - String variable
%specifying the image file extension including the preceding dot. For  example  if
%the file name is 'image003.jpg' the image extension is '.jpg'. StartFrame  - Number
%specifying the first image in the sequence to be analyzed. The minimum
%value for this variable depends on the numbering of the image sequence so
%if  the first image in the sequence is 'image003.tif' then the minimum value
% is 3. FrameCount - Number specifying how many images from the image sequence
%should  be processed. TimeFrame - Number specifying the time between consecutive images in
%minutes. FrameStep  - Number specifying the step size when reading images. Set this
%variable to 1   to read every image in the sequence, 2 to
%read every other image  and  so on. NumberFormat - String value specifying
%the number of digits in  the image file names in  the sequence.
%For example if the image file  name is 'image020.jpg' the value for
%the NumberFormat is '%03d', while if the  file name is 'image000020.jpg' the value
%should  be '%06d'. MaxFramesMissing - Number specifying  for how many frames a
%cell may be disappear before its  track is  ended. OutputFolder - The
%folder where the overlayed images and track data will be  saved. By
%default this value is set to a folder named 'output' within  the folder
%where  the images to be analyzed are located. AncestryFolder - The  folder
%where the overlayed images and ancestry data will be saved. By  default
%this value is set to a folder named 'ancestry' within the output folder. AncestrySpreadsheet
% - The path name to the spreadsheet containing the ancestry data. By default
%this   value is set to a file named 'ancestry.csv' within the ancestry
%folder. ShapesSpreadsheet  - The path name to the spreadsheet containing the position and
%shape properties for   each cell in the timelapse sequence at every time
%point. By default this  is  set to to a file named 'shapes.csv'
%within the ancestry folder. TracksFolder  - The folder where the label matrixes containing
%the cell outlines are saved. By   default this value is set to
%a folder named 'track' within the output  folder. SegmentationFilesRoot - The root file
%name of the label matrixes containing the cell  outlines. ImageFileBase - The path
%name to the images. This value is generated from  the ImageFolder  and
%the ImageFilesRoot and should not be changed. GradientThreshold - Number  specifying the
threshold
%value for the image generated by the generateBinImgUsingGradient filter.  Any  pixels in
%the gradient image below this value will be set to zero   while
%the rest will be set to one. ObjectArea - Number specifying the threshold
%area for the areaFilterLabel, clearSmallObjects, segmentObjectsUsingClusters filters. Objects
below  this value will be
% removed from the filtered image. ObjectReduce - Number specifying how much a cluster
%of  objects should be reduced before it  is processed by the segmentObjectsUsingClusters
%module. By  default this value is set to 1  and it should
%not be changed  unless segmentObjectsUsingClusters throws an out-of-memory error. In that  case
%the value can be  reduced to something lower than 1. ClusterDist - Number
```

```
%specifying the height at which  the hierarchical cluster tree will be cut
%to determine the number of clusters.  Setting this value higher results in a
%cluster  being split into fewer objects  by the segmentObjectsUsingClusters module. ClearBorder
-
%Boolean value specifying whether objects next to or  touching the image border in
% the binary images generated by the generateBinImgUsingGradient module  will be erased (true)
%or not  (false). ClearBorderDist - Number specifying how close  to the border
%objects may be and still be  erased if the ClearBorder  parameter is
%set to true in the generateBinImgUsingGradient module. ResizeImageScale - Number specifying by
what
%ratio the images will be resized before they are  processed. By default
%this value is set to 0.5. Used by the resizeImage module. ApproximationDistance - Number
% specifying how close the convex hull in the getConvexObjects module approximates the
%object  outline. By default this value is set to 2.5. Setting it to
%a   lower value will result in convex hulls that more closely resemble
%the object outlines   however this increases the chance of detecting insignificant concavities.
%MaxMergeDistance - Number specifying  the maximum distance that one track may be from
%another track  for the  duration and still be considered for possible merging
%with the other track.  Used  by detectMergeCandidatesUsingDistance module. MaxSearchRadius -
Number specifying
%the absolute lower bound for the search  radius to prevent selecting  too
%few candidate objects for a track. Used by  assignCellToTrackUsingAll module. MinSearchRadius -
Number
%specifying the absolute higher bound for the search radius  to prevent selecting
%too many candidate objects for a track. Used by assignCellToTrackUsingAll  module.
MinSecondDistance -
%Number specifying the minimum significant distance between the closest candidate object  to a
% track and the second closest. Used to determine when distance should  be
%used as  a ranking parameter. Used by assignCellToTrackUsingAll module. MaxDistRatio - Number
%specifying the maximum allowed distance ratio between the two nearest candidate objects.  If
% the ratio is higher than this value distance ranking will not be used.
%  Used by assignCellToTrackUsingAll module. MaxAngleDiff - Number specifying the maximum allowed
angle
%difference between  a track and a candidate  object. If the angle is
%larger than this  value direction ranking will not be  used for this
%object. Used by assignCellToTrackUsingAll  module. NrParamsForSureMatch - Number specifying the
minimum number of
%closest matches between a candidate  object parameters  and a track's object parameters
%that make the candidate object a  sure match to  the track. Used
%by assignCellToTrackUsingAll module. SearchRadiusPct - Number specifying  the size of the
neighborhood from
%which candidate objects for matching  the track  are selected. It is a
%multiple of the distance to the nearest  candidate  in the current frame.
%Setting this variable equal to 1 turns this module   into a nearest-neighbor
%algorithm (only the nearest cell can be a candidate). It does   not
%make sense to have a value lower than 1. Used by assignCellToTrackUsingAll  module.
%MaxSplitArea - Number specifying the maximum area a nucleus may be and still
%be considered  as a part of a possible mitotic event. Used by detectMitoticEvents
% module. MaxSplitDistance - Number specifying the maximum distance a new nucleus may be
%from  another nucleus  and still be considered as part of a possible
%mitotic event.  Used by detectMitoticEvents  module. MinSplitEccentricity - Number specifying
the minimum
%eccentricity a new  nucleus may have and still be  considered as part
%of a possible mitotic  event. Used by detectMitoticEvents module. MaxSplitEccentricity - Number
specifying
%the maximum eccentricity a new  nucleus may have and still be  considered
%as part of a possible mitotic  event. Used by detectMitoticEvents module. MinTimeForSplit -
%Number specifying the minimum time in minutes  a track needs to exist before
% it is considered for a possible mitotic  event. Used by detectMitoticEvents module.
%MinLifespan - Number specifying the minimum length in frames  a frame has to
%be to  not be removed by the removeShortTracks module.  FrontParams - Numeric
%array specifying a set of column indices from the shape and  motility
%parameters matrix. The parameters in those columns will be heavily weighted, and  have
%more  influence in determining the best match for a track from a
%list of objects.  Used by assignCellToTrackUsingAll module. DefaultParamWeights - Numeric array
specifying a
% set of weights that is assigned to each shape  and motility parameter
%based  on its prediction power. Parameters with high prediction power are  assigned
%high weights  and parameters with low prediction power are assigned lower weights. Used
% by assignCellToTrackUsingAll  module. DistanceRankingOrder - Numeric array specifying the
default order of
%shape and motility parameters  for slow  moving objects when it cannot be
%determined based on prediction power.  Used by assignCellToTrackUsingAll  module.
DirectionRankingOrder - Numeric
%array specifying the default order of  shape and motility parameters for fast
%moving directional objects when it cannot be  determined based on prediction power. Used
%by  assignCellToTrackUsingAll module. RelevantParametersIndex - Boolean array  specifying column
indexes in the
%shape and motility matrix that have  been determined  to be irrelevant for
%tracking. This indicates to the module not to  use  the parameters those
%columns in computing track assignment probabilities. The order of column   indexes is
```

```
%provided in TracksLayout variable. Used by assignCellToTrackUsingAll module. UnknownParamWeights
- Numeric array  specifying
%a set of weights to be assigned to shape and  motility parameters
%when the prediction power of the parameters cannot be determined. Used by
assignCellToTrackUsingAll
% module. UnknownRankingOrder - Numeric array specifying the order of the shape and motility
%parameters  when their  predictive power cannot be determined. If the objects cannot
%be categorized  as either slow-moving  or fast directional the parameter order provided
%in this variable  is used. Used by  assignCellToTrackUsingAll module. Important Modules
areaFilterLabel,
%assignCellToTrackUsingAll, clearSmallObjects, detectMergeCandidatesUsingDistance,
detectMitoticEvents,  generateBinImgUsingGradient, removeShortTracks,
segmentObjectsUsingClusters, segmentObjectsUsingMarkers, splitTracks.

global functions_list;
functions_list=[];
%script variables
ImageFolder='C:/kam/H929_No_Position(8)';
ImageFilesRoot='Experiment-0002_Position(8)_t';
ImageExtension='.jpg';
StartFrame=1;
FrameCount=10;
TimeFrame=15;
FrameStep=1;
NumberFormat='%03d';
MaxFramesMissing=0;
OutputFolder=[ImageFolder '/output'];
AncestryFolder=[OutputFolder '/ancestry'];
AncestrySpreadsheet=[AncestryFolder 'ancestry.csv'];
ShapesSpreadsheet=[AncestryFolder 'shapes.csv'];
TracksFolder=[OutputFolder '/track'];
SegmentationFilesRoot=[TracksFolder '/grayscale'];
ImageFileBase=[ImageFolder '/' ImageFilesRoot];
GradientThreshold=1500;
ObjectArea=200;
ObjectReduce=1;
ClusterDist=21.5;
ClearBorder=true;
ClearBorderDist=2;
ResizeImageScale=0.5;
ApproximationDistance=2.5;
MaxSearchRadius=200;
MinSearchRadius=50;
MinSecondDistance=5;
MaxDistRatio=0.6;
MaxAngleDiff=0.35;
NrParamsForSureMatch=5;
SearchRadiusPct=3;
MaxMergeDistance=23;
MaxSplitArea=400;
MaxSplitDistance=45;
MinSplitEccentricity=0.5;
MaxSplitEccentricity=0.95;
MinTimeForSplit=900;
MinLifespan=30;
FrontParams=[2 3];
DefaultParamWeights=[3 2 0 0 0 0 0 0 0];
DistanceRankingOrder=[2 3 4 5 6 7 8 9 1];
DirectionRankingOrder=[2 3 4 5 6 7 8 9 1];
RelevantParametersIndex=[false true true true true true true true true];
UnknownParamWeights=[3 2 0 0 0 0 0 0 0];
UnknownRankingOrder=[2 3 4 5 6 7 8 9 1];
%end script variables

assign_cells_to_tracks_functions=[];
else_is_empty_cells_label_functions=[];
if_is_empty_cells_label_functions=[];
image_read_loop_functions=[];
image_overlay_loop_functions=[];

loadtrackslayout.InstanceName='LoadTracksLayout';
loadtrackslayout.FunctionHandle=@loadTracksLayout;
loadtrackslayout.FunctionArgs.FileName.Value='tracks_layout.mat';
functions_list=addToFunctionChain(functions_list,loadtrackslayout);

loadancestrylayout.InstanceName='LoadAncestryLayout';
loadancestrylayout.FunctionHandle=@loadAncestryLayout;
loadancestrylayout.FunctionArgs.FileName.Value='ancestry_layout.mat';
functions_list=addToFunctionChain(functions_list,loadancestrylayout);

displaycurframe.InstanceName='DisplayCurFrame';
```

```
displaycurframe.FunctionHandle=@displayVariable;
displaycurframe.FunctionArgs.VariableName.Value='Current Tracking Frame';
displaycurframe.FunctionArgs.Variable.FunctionInstance='SegmentationLoop';
displaycurframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,displaycurframe);

makeimagenamesinsegmentationloop.InstanceName='MakeImageNamesInSegmentationLoop';
makeimagenamesinsegmentationloop.FunctionHandle=@makeImgFileName;
makeimagenamesinsegmentationloop.FunctionArgs.FileBase.Value=ImageFileBase;
makeimagenamesinsegmentationloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinsegmentationloop.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makeimagenamesinsegmentati
onloop);

readimagesinsegmentationloop.InstanceName='ReadImagesInSegmentationLoop';
readimagesinsegmentationloop.FunctionHandle=@readImage;
readimagesinsegmentationloop.FunctionArgs.ImageChannel.Value='';
readimagesinsegmentationloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInSegmentatio
nLoop';
readimagesinsegmentationloop.FunctionArgs.ImageName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,readimagesinsegmentationlo
op);

normalizeimageto16bit.InstanceName='NormalizeImageTo16Bit';
normalizeimageto16bit.FunctionHandle=@imNorm;
normalizeimageto16bit.FunctionArgs.IntegerClass.Value='uint16';
normalizeimageto16bit.FunctionArgs.RawImage.FunctionInstance='ReadImagesInSegmentationLoop';
normalizeimageto16bit.FunctionArgs.RawImage.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,normalizeimageto16bit);

resizeimage.InstanceName='ResizeImage';
resizeimage.FunctionHandle=@resizeImage;
resizeimage.FunctionArgs.Scale.Value=ResizeImageScale;
resizeimage.FunctionArgs.Method.Value='bicubic';
resizeimage.FunctionArgs.Image.FunctionInstance='NormalizeImageTo16Bit';
resizeimage.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,resizeimage);

cytogradientlocalfilter.InstanceName='CytoGradientLocalFilter';
cytogradientlocalfilter.FunctionHandle=@generateBinImgUsingGradient;
cytogradientlocalfilter.FunctionArgs.GradientThreshold.Value=GradientThreshold;
cytogradientlocalfilter.FunctionArgs.ClearBorder.Value=ClearBorder;
cytogradientlocalfilter.FunctionArgs.ClearBorderDist.Value=ClearBorderDist;
cytogradientlocalfilter.FunctionArgs.Image.FunctionInstance='ResizeImage';
cytogradientlocalfilter.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,cytogradientlocalfilter);

fillholescytoplasmimages.InstanceName='FillHolesCytoplasmImages';
fillholescytoplasmimages.FunctionHandle=@fillHoles;
fillholescytoplasmimages.FunctionArgs.Image.FunctionInstance='CytoGradientLocalFilter';
fillholescytoplasmimages.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,fillholescytoplasmimages);

clearsmallcells.InstanceName='ClearSmallCells';
clearsmallcells.FunctionHandle=@clearSmallObjects;
clearsmallcells.FunctionArgs.MinObjectArea.Value=ObjectArea;
clearsmallcells.FunctionArgs.Image.FunctionInstance='FillHolesCytoplasmImages';
clearsmallcells.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,clearsmallcells);

nuclgradientlocalfilter.InstanceName='NuclGradientLocalFilter';
nuclgradientlocalfilter.FunctionHandle=@generateBinImgUsingGradient;
nuclgradientlocalfilter.FunctionArgs.GradientThreshold.Value=GradientThreshold;
nuclgradientlocalfilter.FunctionArgs.ClearBorder.Value=ClearBorder;
nuclgradientlocalfilter.FunctionArgs.ClearBorderDist.Value=ClearBorderDist;
nuclgradientlocalfilter.FunctionArgs.Image.FunctionInstance='ResizeImage';
nuclgradientlocalfilter.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,nuclgradientlocalfilter);

fillholesnuclearimages.InstanceName='FillHolesNuclearImages';
fillholesnuclearimages.FunctionHandle=@fillHoles;
fillholesnuclearimages.FunctionArgs.Image.FunctionInstance='NuclGradientLocalFilter';
fillholesnuclearimages.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,fillholesnuclearimages);

clearsmallnuclei.InstanceName='ClearSmallNuclei';
clearsmallnuclei.FunctionHandle=@clearSmallObjects;
clearsmallnuclei.FunctionArgs.MinObjectArea.Value=ObjectArea;
clearsmallnuclei.FunctionArgs.Image.FunctionInstance='FillHolesNuclearImages';
```

```
clearsmallnuclei.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,clearsmallnuclei);

combinenuclearandcytoplasmimages.InstanceName='CombineNuclearAndCytoplasmImages';
combinenuclearandcytoplasmimages.FunctionHandle=@combineImages;
combinenuclearandcytoplasmimages.FunctionArgs.CombineOperation.Value='AND';
combinenuclearandcytoplasmimages.FunctionArgs.Image1.FunctionInstance='ClearSmallNuclei';
combinenuclearandcytoplasmimages.FunctionArgs.Image1.OutputArg='Image';
combinenuclearandcytoplasmimages.FunctionArgs.Image2.FunctionInstance='ClearSmallCells';
combinenuclearandcytoplasmimages.FunctionArgs.Image2.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,combinenuclearandcytoplasm
images);

reconstructcytoplasmimage.InstanceName='ReconstructCytoplasmImage';
reconstructcytoplasmimage.FunctionHandle=@reconstructObjects;
reconstructcytoplasmimage.FunctionArgs.GuideImage.FunctionInstance='CombineNuclearAndCytoplasmIma
ges';
reconstructcytoplasmimage.FunctionArgs.GuideImage.OutputArg='Image';
reconstructcytoplasmimage.FunctionArgs.ImageToReconstruct.FunctionInstance='ClearSmallNuclei';
reconstructcytoplasmimage.FunctionArgs.ImageToReconstruct.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,reconstructcytoplasmimage)
;

labelnuclei.InstanceName='LabelNuclei';
labelnuclei.FunctionHandle=@labelObjects;
labelnuclei.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
labelnuclei.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,labelnuclei);

labelcytoplasm.InstanceName='LabelCytoplasm';
labelcytoplasm.FunctionHandle=@labelObjects;
labelcytoplasm.FunctionArgs.Image.FunctionInstance='ReconstructCytoplasmImage';
labelcytoplasm.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,labelcytoplasm);

getconvexobjects.InstanceName='GetConvexObjects';
getconvexobjects.FunctionHandle=@getConvexObjects;
getconvexobjects.FunctionArgs.ApproximationDistance.Value=ApproximationDistance;
getconvexobjects.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
getconvexobjects.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getconvexobjects);

segmentobjectsusingclusters.InstanceName='SegmentObjectsUsingClusters';
segmentobjectsusingclusters.FunctionHandle=@segmentObjectsUsingClusters;
segmentobjectsusingclusters.FunctionArgs.ObjectReduce.Value=ObjectReduce;
segmentobjectsusingclusters.FunctionArgs.MinimumObjectArea.Value=ObjectArea;
segmentobjectsusingclusters.FunctionArgs.ClusterDistance.Value=ClusterDist;
segmentobjectsusingclusters.FunctionArgs.ObjectsLabel.FunctionInstance='LabelNuclei';
segmentobjectsusingclusters.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,segmentobjectsusingcluster
s);

segmentobjectsusingmarkers.InstanceName='SegmentObjectsUsingMarkers';
segmentobjectsusingmarkers.FunctionHandle=@segmentObjectsUsingMarkers;
segmentobjectsusingmarkers.FunctionArgs.MarkersLabel.FunctionInstance='SegmentObjectsUsingCluster
s';
segmentobjectsusingmarkers.FunctionArgs.MarkersLabel.OutputArg='LabelMatrix';
segmentobjectsusingmarkers.FunctionArgs.ObjectsLabel.FunctionInstance='LabelCytoplasm';
segmentobjectsusingmarkers.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,segmentobjectsusingmarkers
);

areafilter.InstanceName='AreaFilter';
areafilter.FunctionHandle=@areaFilterLabel;
areafilter.FunctionArgs.MinArea.Value=ObjectArea;
areafilter.FunctionArgs.ObjectsLabel.FunctionInstance='SegmentObjectsUsingMarkers';
areafilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,areafilter);

resizecytolabel.InstanceName='ResizeCytoLabel';
resizecytolabel.FunctionHandle=@resizeImage;
resizecytolabel.FunctionArgs.Scale.Value=1/ResizeImageScale;
resizecytolabel.FunctionArgs.Method.Value='nearest';
resizecytolabel.FunctionArgs.Image.FunctionInstance='AreaFilter';
resizecytolabel.FunctionArgs.Image.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,resizecytolabel);

getshapeparameters.InstanceName='GetShapeParameters';
getshapeparameters.FunctionHandle=@getShapeParams;
getshapeparameters.FunctionArgs.LabelMatrix.FunctionInstance='ResizeCytoLabel';
getshapeparameters.FunctionArgs.LabelMatrix.OutputArg='Image';
```

```
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getshapeparameters);

isemptypreviouscellslabel.InstanceName='IsEmptyPreviousCellsLabel';
isemptypreviouscellslabel.FunctionHandle=@isEmptyFunction;
isemptypreviouscellslabel.FunctionArgs.TestVariable.Value=[];
isemptypreviouscellslabel.FunctionArgs.TestVariable.FunctionInstance='SaveCellsLabel';
isemptypreviouscellslabel.FunctionArgs.TestVariable.OutputArg='CellsLabel';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,isemptypreviouscellslabel)
;

starttracks.InstanceName='StartTracks';
starttracks.FunctionHandle=@startTracks;
starttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
starttracks.FunctionArgs.ShapeParameters.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.ShapeParameters.InputArg='GetShapeParameters_ShapeParameters';
starttracks.FunctionArgs.CellsLabel.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.CellsLabel.InputArg='ResizeCytoLabel_Image';
starttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
if_is_empty_cells_label_functions=addToFunctionChain(if_is_empty_cells_label_functions,starttrack
s);

getcurrenttracks.InstanceName='GetCurrentTracks';
getcurrenttracks.FunctionHandle=@getCurrentTracks;
getcurrenttracks.FunctionArgs.OffsetFrame.Value=-1;
getcurrenttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getcurrenttracks.FunctionArgs.TimeCol.Value=2;
getcurrenttracks.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getcurrenttracks.FunctionArgs.FrameStep.Value=FrameStep;
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg='Tracks';
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getcurrenttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getcurrenttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getcur
renttracks);

getprevioustracks.InstanceName='GetPreviousTracks';
getprevioustracks.FunctionHandle=@getCurrentTracks;
getprevioustracks.FunctionArgs.OffsetFrame.Value=-2;
getprevioustracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getprevioustracks.FunctionArgs.TimeCol.Value=2;
getprevioustracks.FunctionArgs.TrackIDCol.Value=1;
getprevioustracks.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getprevioustracks.FunctionArgs.FrameStep.Value=FrameStep;
getprevioustracks.FunctionArgs.Tracks.Value=[];
getprevioustracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg='CellsLabel';
getprevioustracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getprevioustracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getprevioustracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getpre
vioustracks);

makeunassignedcellslist.InstanceName='MakeUnassignedCellsList';
makeunassignedcellslist.FunctionHandle=@makeUnassignedCellsList;
makeunassignedcellslist.FunctionArgs.CellsCentroids.FunctionInstance='IfIsEmptyPreviousCellsLabel
';
makeunassignedcellslist.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,makeun
assignedcellslist);

makeexcludedtrackslist.InstanceName='MakeExcludedTracksList';
makeexcludedtrackslist.FunctionHandle=@makeExcludedTracksList;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.FunctionInstance='MakeUnassignedCellsList'
;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.OutputArg='UnassignedCellsIDs';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,makeex
cludedtrackslist);

getcellsmeandisplacement.InstanceName='GetCellsMeanDisplacement';
getcellsmeandisplacement.FunctionHandle=@getObjectsMeanDisplacement;
getcellsmeandisplacement.FunctionArgs.Centroid1Col.Value=3;
getcellsmeandisplacement.FunctionArgs.Centroid2Col.Value=4;
getcellsmeandisplacement.FunctionArgs.CurrentTracks.FunctionInstance='GetCurrentTracks';
getcellsmeandisplacement.FunctionArgs.CurrentTracks.OutputArg='Tracks';
getcellsmeandisplacement.FunctionArgs.ObjectCentroids.FunctionInstance='IfIsEmptyPreviousCellsLab
el';
```

```
getcellsmeandisplacement.FunctionArgs.ObjectCentroids.InputArg='GetShapeParameters_Centroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getcel
lsmeandisplacement);

getparamscoefficientofvariation.InstanceName='GetParamsCoefficientOfVariation';
getparamscoefficientofvariation.FunctionHandle=@getParamsCoefficientOfVariation;
getparamscoefficientofvariation.FunctionArgs.AreaCol.Value=5;
getparamscoefficientofvariation.FunctionArgs.SolidityCol.Value=11;
getparamscoefficientofvariation.FunctionArgs.Params.FunctionInstance='IfIsEmptyPreviousCellsLabel
';
getparamscoefficientofvariation.FunctionArgs.Params.InputArg='GetShapeParameters_ShapeParameters'
;
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getpar
amscoefficientofvariation);

getmaxtrackid.InstanceName='GetMaxTrackID';
getmaxtrackid.FunctionHandle=@getMaxTrackID;
getmaxtrackid.FunctionArgs.TrackIDCol.Value=1;
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg='Tracks';
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg2='Tracks';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getmax
trackid);

isnotemptyunassignedcells.InstanceName='IsNotEmptyUnassignedCells';
isnotemptyunassignedcells.FunctionHandle=@isNotEmptyFunction;
isnotemptyunassignedcells.FunctionArgs.TestVariable.FunctionInstance='AssignCellToTrackUsingAll';
isnotemptyunassignedcells.FunctionArgs.TestVariable.OutputArg='UnassignedIDs';
isnotemptyunassignedcells.FunctionArgs.TestVariable.FunctionInstance2='AssignCellsToTracksLoop';
isnotemptyunassignedcells.FunctionArgs.TestVariable.InputArg2='MakeUnassignedCellsList_Unassigned
CellsIDs';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,isnotemptyun
assignedcells);

getcurrentunassignedcell.InstanceName='GetCurrentUnassignedCell';
getcurrentunassignedcell.FunctionHandle=@getCurrentUnassignedCell;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance='AssignCellToTrackUsingAll
';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTracksLoop'
;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassign
edCellsIDs';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,getcurrentun
assignedcell);

assigncelltotrackusingall.InstanceName='AssignCellToTrackUsingAll';
assigncelltotrackusingall.FunctionHandle=@assignCellToTrackUsingAll;
assigncelltotrackusingall.FunctionArgs.CheckCellPath.Value=false;
assigncelltotrackusingall.FunctionArgs.FrontParams.Value=FrontParams;
assigncelltotrackusingall.FunctionArgs.SearchRadiusPct.Value=SearchRadiusPct;
assigncelltotrackusingall.FunctionArgs.MaxSearchRadius.Value=MaxSearchRadius;
assigncelltotrackusingall.FunctionArgs.MinSearchRadius.Value=MinSearchRadius;
assigncelltotrackusingall.FunctionArgs.RelevantParametersIndex.Value=RelevantParametersIndex;
assigncelltotrackusingall.FunctionArgs.NrParamsForSureMatch.Value=NrParamsForSureMatch;
assigncelltotrackusingall.FunctionArgs.DefaultParamWeights.Value=DefaultParamWeights;
assigncelltotrackusingall.FunctionArgs.UnknownParamWeights.Value=UnknownParamWeights;
assigncelltotrackusingall.FunctionArgs.DistanceRankingOrder.Value=DistanceRankingOrder;
assigncelltotrackusingall.FunctionArgs.DirectionRankingOrder.Value=DirectionRankingOrder;
assigncelltotrackusingall.FunctionArgs.UnknownRankingOrder.Value=UnknownRankingOrder;
assigncelltotrackusingall.FunctionArgs.MinSecondDistance.Value=MinSecondDistance;
assigncelltotrackusingall.FunctionArgs.MaxDistRatio.Value=MaxDistRatio;
assigncelltotrackusingall.FunctionArgs.MaxAngleDiff.Value=MaxAngleDiff;
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.Value=[];
assigncelltotrackusingall.FunctionArgs.TrackAssignments.Value=[];
assigncelltotrackusingall.FunctionArgs.Tracks.Value=[];
assigncelltotrackusingall.FunctionArgs.MatchingGroups.Value=[];
assigncelltotrackusingall.FunctionArgs.MatchingGroupsStats.Value=[];
assigncelltotrackusingall.FunctionArgs.UnassignedCells.FunctionInstance='AssignCellToTrackUsingAl
l';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.FunctionInstance='AssignCellToTrackUsingAll
';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.OutputArg='ExcludedTracks';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.FunctionInstance='SetMatchingGroupIndex';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.OutputArg='ShapeParameters';
assigncelltotrackusingall.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellToTrackUsingA
ll';
assigncelltotrackusingall.FunctionArgs.TrackAssignments.OutputArg='TrackAssignments';
```

```
assigncelltotrackusingall.FunctionArgs.MatchingGroups.FunctionInstance='AssignCellToTrackUsingAll
';
assigncelltotrackusingall.FunctionArgs.MatchingGroups.OutputArg='MatchingGroups';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTracksLoop
';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassig
nedCellsIDs';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.FunctionInstance2='AssignCellsToTracksLoop'
;
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.InputArg2='MakeExcludedTracksList_ExcludedT
racks';
assigncelltotrackusingall.FunctionArgs.CellsLabel.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CellsLabel.InputArg='ResizeCytoLabel_Image';
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.FunctionInstance='AssignCellsToTracksLo
op';
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.InputArg='SaveCellsLabel_CellsLabel';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.FunctionInstance2='AssignCellsToTracksLoop
';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.InputArg2='GetShapeParameters_ShapeParamet
ers';
assigncelltotrackusingall.FunctionArgs.CellsCentroids.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
assigncelltotrackusingall.FunctionArgs.CurrentTracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CurrentTracks.InputArg='GetCurrentTracks_Tracks';
assigncelltotrackusingall.FunctionArgs.MaxTrackID.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.MaxTrackID.InputArg='GetMaxTrackID_MaxTrackID';
assigncelltotrackusingall.FunctionArgs.Tracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.Tracks.InputArg='SegmentationLoop_Tracks';
assigncelltotrackusingall.FunctionArgs.MatchingGroupsStats.FunctionInstance='AssignCellsToTracksL
oop';
assigncelltotrackusingall.FunctionArgs.MatchingGroupsStats.InputArg='GetMatchingGroupMeans_Matchi
ngGroupStats';
assigncelltotrackusingall.FunctionArgs.ParamsCoeffOfVariation.FunctionInstance='AssignCellsToTrac
ksLoop';
assigncelltotrackusingall.FunctionArgs.ParamsCoeffOfVariation.InputArg='GetParamsCoefficientOfVar
iation_CoefficientOfVariation';
assigncelltotrackusingall.FunctionArgs.PreviousTracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.PreviousTracks.InputArg='GetPreviousTracks_Tracks';
assigncelltotrackusingall.FunctionArgs.TracksLayout.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
assigncelltotrackusingall.FunctionArgs.MatchingGroups.FunctionInstance2='AssignCellsToTracksLoop'
;
assigncelltotrackusingall.FunctionArgs.MatchingGroups.InputArg2='HoldMatchingGroups_ValueToHold';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,assigncellto
trackusingall);

setmatchinggroupindex.InstanceName='SetMatchingGroupIndex';
setmatchinggroupindex.FunctionHandle=@setGroupIndex;
setmatchinggroupindex.FunctionArgs.AreaCol.Value=5;
setmatchinggroupindex.FunctionArgs.GroupIDCol.Value=13;
setmatchinggroupindex.FunctionArgs.CellID.FunctionInstance='GetCurrentUnassignedCell';
setmatchinggroupindex.FunctionArgs.CellID.OutputArg='CellID';
setmatchinggroupindex.FunctionArgs.GroupIndex.FunctionInstance='AssignCellToTrackUsingAll';
setmatchinggroupindex.FunctionArgs.GroupIndex.OutputArg='GroupIndex';
setmatchinggroupindex.FunctionArgs.ShapeParameters.FunctionInstance='SetMatchingGroupIndex';
setmatchinggroupindex.FunctionArgs.ShapeParameters.OutputArg='ShapeParameters';
setmatchinggroupindex.FunctionArgs.ShapeParameters.FunctionInstance2='AssignCellsToTracksLoop';
setmatchinggroupindex.FunctionArgs.ShapeParameters.InputArg2='GetShapeParameters_ShapeParameters'
;
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,setmatchingg
roupindex);

assigncellstotracksloop.InstanceName='AssignCellsToTracksLoop';
assigncellstotracksloop.FunctionHandle=@whileLoop;
assigncellstotracksloop.FunctionArgs.TestFunction.FunctionInstance='IsNotEmptyUnassignedCells';
assigncellstotracksloop.FunctionArgs.TestFunction.OutputArg='Boolean';
assigncellstotracksloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.FunctionInstance=
'MakeUnassignedCellsList';
assigncellstotracksloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.OutputArg='Unassi
gnedCellsIDs';
assigncellstotracksloop.FunctionArgs.MakeExcludedTracksList_ExcludedTracks.FunctionInstance='Make
ExcludedTracksList';
assigncellstotracksloop.FunctionArgs.MakeExcludedTracksList_ExcludedTracks.OutputArg='ExcludedTra
cks';
assigncellstotracksloop.FunctionArgs.GetCurrentTracks_Tracks.FunctionInstance='GetCurrentTracks';
assigncellstotracksloop.FunctionArgs.GetCurrentTracks_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.GetMaxTrackID_MaxTrackID.FunctionInstance='GetMaxTrackID';
assigncellstotracksloop.FunctionArgs.GetMaxTrackID_MaxTrackID.OutputArg='MaxTrackID';
assigncellstotracksloop.FunctionArgs.GetMatchingGroupMeans_MatchingGroupStats.FunctionInstance='G
etMatchingGroupMeans';
```

```
assigncellstotracksloop.FunctionArgs.GetMatchingGroupMeans_MatchingGroupStats.OutputArg='Matching
GroupStats';
assigncellstotracksloop.FunctionArgs.GetParamsCoefficientOfVariation_CoefficientOfVariation.Funct
ionInstance='GetParamsCoefficientOfVariation';
assigncellstotracksloop.FunctionArgs.GetParamsCoefficientOfVariation_CoefficientOfVariation.Outpu
tArg='CoefficientOfVariation';
assigncellstotracksloop.FunctionArgs.GetPreviousTracks_Tracks.FunctionInstance='GetPreviousTracks
';
assigncellstotracksloop.FunctionArgs.GetPreviousTracks_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.ResizeCytoLabel_Image.FunctionInstance='IfIsEmptyPreviousCel
lsLabel';
assigncellstotracksloop.FunctionArgs.ResizeCytoLabel_Image.InputArg='ResizeCytoLabel_Image';
assigncellstotracksloop.FunctionArgs.SaveCellsLabel_CellsLabel.FunctionInstance='IfIsEmptyPreviou
sCellsLabel';
assigncellstotracksloop.FunctionArgs.SaveCellsLabel_CellsLabel.InputArg='SaveCellsLabel_CellsLabe
l';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_ShapeParameters.FunctionInstance='IfIsEmp
tyPreviousCellsLabel';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_ShapeParameters.InputArg='GetShapeParamet
ers_ShapeParameters';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_Centroids.FunctionInstance='IfIsEmptyPrev
iousCellsLabel';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_Centroids.InputArg='GetShapeParameters_Ce
ntroids';
assigncellstotracksloop.FunctionArgs.SegmentationLoop_Tracks.FunctionInstance='IfIsEmptyPreviousC
ellsLabel';
assigncellstotracksloop.FunctionArgs.SegmentationLoop_Tracks.InputArg='SegmentationLoop_Tracks';
assigncellstotracksloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='IfIsEmptyPre
viousCellsLabel';
assigncellstotracksloop.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout_Tra
cksLayout';
assigncellstotracksloop.FunctionArgs.HoldMatchingGroups_ValueToHold.FunctionInstance='IfIsEmptyPr
eviousCellsLabel';
assigncellstotracksloop.FunctionArgs.HoldMatchingGroups_ValueToHold.InputArg='HoldMatchingGroups_
ValueToHold';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_TrackAssignments.FunctionInstance='A
ssignCellToTrackUsingAll';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_TrackAssignments.OutputArg='TrackAss
ignments';
assigncellstotracksloop.KeepValues.SetMatchingGroupIndex_ShapeParameters.FunctionInstance='SetMat
chingGroupIndex';
assigncellstotracksloop.KeepValues.SetMatchingGroupIndex_ShapeParameters.OutputArg='ShapeParamete
rs';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.FunctionInstance='Ass
ignCellToTrackUsingAll';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.OutputArg='MatchingGr
oups';
assigncellstotracksloop.LoopFunctions=assign_cells_to_tracks_functions;
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,assign
cellstotracksloop);

continuetracks.InstanceName='ContinueTracks';
continuetracks.FunctionHandle=@continueTracks;
continuetracks.FunctionArgs.TimeFrame.Value=TimeFrame;
continuetracks.FunctionArgs.Tracks.Value=[];
continuetracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
continuetracks.FunctionArgs.Tracks.OutputArg='Tracks';
continuetracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
continuetracks.FunctionArgs.Tracks.OutputArg2='Tracks';
continuetracks.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellsToTracksLoop';
continuetracks.FunctionArgs.TrackAssignments.OutputArg='AssignCellToTrackUsingAll_TrackAssignment
s';
continuetracks.FunctionArgs.ShapeParameters.FunctionInstance='AssignCellsToTracksLoop';
continuetracks.FunctionArgs.ShapeParameters.OutputArg='SetMatchingGroupIndex_ShapeParameters';
continuetracks.FunctionArgs.CellsCentroids.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
continuetracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,contin
uetracks);

getmatchinggroupmeans.InstanceName='GetMatchingGroupMeans';
getmatchinggroupmeans.FunctionHandle=@getMatchingGroupMeans;
getmatchinggroupmeans.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getmatchinggroupmeans.FunctionArgs.Tracks.OutputArg='Tracks';
getmatchinggroupmeans.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getmatchinggroupmeans.FunctionArgs.Tracks.OutputArg2='Tracks';
getmatchinggroupmeans.FunctionArgs.TracksLayout.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getmatchinggroupmeans.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getmat
chinggroupmeans);
```

```
ifisemptypreviouscellslabel.InstanceName='IfIsEmptyPreviousCellsLabel';
ifisemptypreviouscellslabel.FunctionHandle=@if_statement;
ifisemptypreviouscellslabel.FunctionArgs.PreviousCellsLabel.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.Tracks.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.MatchingGroupsStats.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.TrackAssignments.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.TestVariable.FunctionInstance='IsEmptyPreviousCellsLabel
';
ifisemptypreviouscellslabel.FunctionArgs.TestVariable.OutputArg='Boolean';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_ShapeParameters.FunctionInstance='Get
ShapeParameters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_ShapeParameters.OutputArg='ShapeParam
eters';
ifisemptypreviouscellslabel.FunctionArgs.ResizeCytoLabel_Image.FunctionInstance='ResizeCytoLabel'
;
ifisemptypreviouscellslabel.FunctionArgs.ResizeCytoLabel_Image.OutputArg='Image';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_LoopCounter.FunctionInstance='Segmentat
ionLoop';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_LoopCounter.OutputArg='LoopCounter';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_Centroids.FunctionInstance='GetShapeP
arameters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_Centroids.OutputArg='Centroids';
ifisemptypreviouscellslabel.FunctionArgs.SaveCellsLabel_CellsLabel.FunctionInstance='SaveCellsLab
el';
ifisemptypreviouscellslabel.FunctionArgs.SaveCellsLabel_CellsLabel.OutputArg='CellsLabel';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_Tracks.FunctionInstance='SegmentationLo
op';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_Tracks.OutputArg='Tracks';
ifisemptypreviouscellslabel.FunctionArgs.HoldMatchingGroups_ValueToHold.FunctionInstance='HoldMat
chingGroups';
ifisemptypreviouscellslabel.FunctionArgs.HoldMatchingGroups_ValueToHold.OutputArg='ValueToHold';
ifisemptypreviouscellslabel.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='Segmenta
tionLoop';
ifisemptypreviouscellslabel.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout
_TracksLayout';
ifisemptypreviouscellslabel.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.FunctionInstance=
'AssignCellsToTracksLoop';
ifisemptypreviouscellslabel.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.OutputArg='Assign
CellToTrackUsingAll_MatchingGroups';
ifisemptypreviouscellslabel.KeepValues.ContinueTracks_Tracks.FunctionInstance='ContinueTracks';
ifisemptypreviouscellslabel.KeepValues.ContinueTracks_Tracks.OutputArg='Tracks';
ifisemptypreviouscellslabel.ElseFunctions=else_is_empty_cells_label_functions;
ifisemptypreviouscellslabel.IfFunctions=if_is_empty_cells_label_functions;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,ifisemptypreviouscellslabe
l);

holdmatchinggroups.InstanceName='HoldMatchingGroups';
holdmatchinggroups.FunctionHandle=@holdValue;
holdmatchinggroups.FunctionArgs.ValueToHold.Value=[];
holdmatchinggroups.FunctionArgs.ValueToHold.FunctionInstance='IfIsEmptyPreviousCellsLabel';
holdmatchinggroups.FunctionArgs.ValueToHold.OutputArg='AssignCellToTrackUsingAll_MatchingGroups';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,holdmatchinggroups);

savecellslabel.InstanceName='SaveCellsLabel';
savecellslabel.FunctionHandle=@saveCellsLabel;
savecellslabel.FunctionArgs.FileRoot.Value=SegmentationFilesRoot;
savecellslabel.FunctionArgs.NumberFormat.Value=NumberFormat;
savecellslabel.FunctionArgs.CellsLabel.FunctionInstance='ResizeCytoLabel';
savecellslabel.FunctionArgs.CellsLabel.OutputArg='Image';
savecellslabel.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
savecellslabel.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,savecellslabel);

segmentationloop.InstanceName='SegmentationLoop';
segmentationloop.FunctionHandle=@forLoop;
segmentationloop.FunctionArgs.StartLoop.Value=StartFrame;
segmentationloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
segmentationloop.FunctionArgs.IncrementLoop.Value=FrameStep;
segmentationloop.FunctionArgs.MatchingGroups.Value=[];
segmentationloop.FunctionArgs.Tracks.Value=[];
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='LoadTracksLayout';
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.OutputArg='TracksLayout';
segmentationloop.KeepValues.ContinueTracks_Tracks.FunctionInstance='IfIsEmptyPreviousCellsLabel';
segmentationloop.KeepValues.ContinueTracks_Tracks.OutputArg='ContinueTracks_Tracks';
segmentationloop.LoopFunctions=image_read_loop_functions;
functions_list=addToFunctionChain(functions_list,segmentationloop);

savetracks.InstanceName='SaveTracks';
savetracks.FunctionHandle=@saveTracks;
savetracks.FunctionArgs.TracksFileName.Value=[TracksFolder '/tracks.mat'];
```

```
savetracks.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
savetracks.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,savetracks);

gettrackids.InstanceName='GetTrackIDs';
gettrackids.FunctionHandle=@getTrackIDs;
gettrackids.FunctionArgs.TrackIDCol.Value=1;
gettrackids.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
gettrackids.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,gettrackids);

detectmergecandidates.InstanceName='DetectMergeCandidates';
detectmergecandidates.FunctionHandle=@detectMergeCandidatesUsingDistance;
detectmergecandidates.FunctionArgs.MaxMergeDistance.Value=MaxMergeDistance;
detectmergecandidates.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDs';
detectmergecandidates.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
detectmergecandidates.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
detectmergecandidates.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
detectmergecandidates.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
detectmergecandidates.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,detectmergecandidates);

mergetracks.InstanceName='MergeTracks';
mergetracks.FunctionHandle=@mergeTracks;
mergetracks.FunctionArgs.FrameCount.Value=FrameCount;
mergetracks.FunctionArgs.StartFrame.Value=StartFrame;
mergetracks.FunctionArgs.TimeFrame.Value=TimeFrame;
mergetracks.FunctionArgs.SegFileRoot.Value=SegmentationFilesRoot;
mergetracks.FunctionArgs.FrameStep.Value=FrameStep;
mergetracks.FunctionArgs.NumberFormat.Value=NumberFormat;
mergetracks.FunctionArgs.TracksToBeMerged.FunctionInstance='DetectMergeCandidates';
mergetracks.FunctionArgs.TracksToBeMerged.OutputArg='TracksToBeMerged';
mergetracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
mergetracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
mergetracks.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
mergetracks.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,mergetracks);

gettrackidsaftermerge.InstanceName='GetTrackIDsAfterMerge';
gettrackidsaftermerge.FunctionHandle=@getTrackIDs;
gettrackidsaftermerge.FunctionArgs.TrackIDCol.Value=1;
gettrackidsaftermerge.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
gettrackidsaftermerge.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,gettrackidsaftermerge);

makeancestryforfirstframecells.InstanceName='MakeAncestryForFirstFrameCells';
makeancestryforfirstframecells.FunctionHandle=@makeAncestryForFirstFrameCells;
makeancestryforfirstframecells.FunctionArgs.TimeCol.Value=2;
makeancestryforfirstframecells.FunctionArgs.TrackIDCol.Value=1;
makeancestryforfirstframecells.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
makeancestryforfirstframecells.FunctionArgs.Tracks.OutputArg='Tracks';
makeancestryforfirstframecells.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDsAfterMerge';
makeancestryforfirstframecells.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
functions_list=addToFunctionChain(functions_list,makeancestryforfirstframecells);

detectmitoticevents.InstanceName='DetectMitoticEvents';
detectmitoticevents.FunctionHandle=@detectMitoticEvents;
detectmitoticevents.FunctionArgs.MaxSplitArea.Value=MaxSplitArea;
detectmitoticevents.FunctionArgs.MinSplitEccentricity.Value=MinSplitEccentricity;
detectmitoticevents.FunctionArgs.MaxSplitEccentricity.Value=MaxSplitEccentricity;
detectmitoticevents.FunctionArgs.MaxSplitDistance.Value=MaxSplitDistance;
detectmitoticevents.FunctionArgs.MinTimeForSplit.Value=MinTimeForSplit;
detectmitoticevents.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
detectmitoticevents.FunctionArgs.Tracks.OutputArg='Tracks';
detectmitoticevents.FunctionArgs.UntestedIDs.FunctionInstance='MakeAncestryForFirstFrameCells';
detectmitoticevents.FunctionArgs.UntestedIDs.OutputArg='UntestedIDs';
detectmitoticevents.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
detectmitoticevents.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
functions_list=addToFunctionChain(functions_list,detectmitoticevents);

makeancestryforcellsenteringframes.InstanceName='MakeAncestryForCellsEnteringFrames';
makeancestryforcellsenteringframes.FunctionHandle=@makeAncestryForCellsEnteringFrames;
makeancestryforcellsenteringframes.FunctionArgs.TimeCol.Value=2;
makeancestryforcellsenteringframes.FunctionArgs.TrackIDCol.Value=1;
makeancestryforcellsenteringframes.FunctionArgs.SplitCells.FunctionInstance='DetectMitoticEvents'
;
makeancestryforcellsenteringframes.FunctionArgs.SplitCells.OutputArg='SplitCells';
makeancestryforcellsenteringframes.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDsAfterMerge'
;
makeancestryforcellsenteringframes.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
```

```
makeancestryforcellsenteringframes.FunctionArgs.FirstFrameIDs.FunctionInstance='MakeAncestryForFi
rstFrameCells';
makeancestryforcellsenteringframes.FunctionArgs.FirstFrameIDs.OutputArg='FirstFrameIDs';
makeancestryforcellsenteringframes.FunctionArgs.CellsAncestry.FunctionInstance='MakeAncestryForFi
rstFrameCells';
makeancestryforcellsenteringframes.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
makeancestryforcellsenteringframes.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
makeancestryforcellsenteringframes.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,makeancestryforcellsenteringframes);

splittracks.InstanceName='SplitTracks';
splittracks.FunctionHandle=@splitTracks;
splittracks.FunctionArgs.TimeFrame.Value=TimeFrame;
splittracks.FunctionArgs.SplitCells.FunctionInstance='DetectMitoticEvents';
splittracks.FunctionArgs.SplitCells.OutputArg='SplitCells';
splittracks.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
splittracks.FunctionArgs.Tracks.OutputArg='Tracks';
splittracks.FunctionArgs.CellsAncestry.FunctionInstance='MakeAncestryForCellsEnteringFrames';
splittracks.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
splittracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
splittracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
splittracks.FunctionArgs.AncestryLayout.FunctionInstance='LoadAncestryLayout';
splittracks.FunctionArgs.AncestryLayout.OutputArg='AncestryLayout';
functions_list=addToFunctionChain(functions_list,splittracks);

removeshorttracks.InstanceName='RemoveShortTracks';
removeshorttracks.FunctionHandle=@removeShortTracks;
removeshorttracks.FunctionArgs.MinLifespan.Value=MinLifespan;
removeshorttracks.FunctionArgs.Tracks.FunctionInstance='SplitTracks';
removeshorttracks.FunctionArgs.Tracks.OutputArg='Tracks';
removeshorttracks.FunctionArgs.CellsAncestry.FunctionInstance='SplitTracks';
removeshorttracks.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
removeshorttracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
removeshorttracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
removeshorttracks.FunctionArgs.AncestryLayout.FunctionInstance='LoadAncestryLayout';
removeshorttracks.FunctionArgs.AncestryLayout.OutputArg='AncestryLayout';
functions_list=addToFunctionChain(functions_list,removeshorttracks);

saveupdatedtracks.InstanceName='SaveUpdatedTracks';
saveupdatedtracks.FunctionHandle=@saveTracks;
saveupdatedtracks.FunctionArgs.TracksFileName.Value=[AncestryFolder '/tracks.mat'];
saveupdatedtracks.FunctionArgs.Tracks.FunctionInstance='RemoveShortTracks';
saveupdatedtracks.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,saveupdatedtracks);

saveancestry.InstanceName='SaveAncestry';
saveancestry.FunctionHandle=@saveAncestry;
saveancestry.FunctionArgs.AncestryFileName.Value=[AncestryFolder '/ancestry.mat'];
saveancestry.FunctionArgs.CellsAncestry.FunctionInstance='RemoveShortTracks';
saveancestry.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
functions_list=addToFunctionChain(functions_list,saveancestry);

makeimagenamesinoverlayloop.InstanceName='MakeImageNamesInOverlayLoop';
makeimagenamesinoverlayloop.FunctionHandle=@makeImgFileName;
makeimagenamesinoverlayloop.FunctionArgs.FileBase.Value=ImageFileBase;
makeimagenamesinoverlayloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinoverlayloop.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenamesinoverlayloop.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
makeimagenamesinoverlayloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,makeimagenamesinover
layloop);

readimagesinoverlayloop.InstanceName='ReadImagesInOverlayLoop';
readimagesinoverlayloop.FunctionHandle=@readImage;
readimagesinoverlayloop.FunctionArgs.ImageChannel.Value='';
readimagesinoverlayloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInOverlayLoop';
readimagesinoverlayloop.FunctionArgs.ImageName.OutputArg='FileName';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,readimagesinoverlayl
oop);

getcurrenttracks2.InstanceName='GetCurrentTracks2';
getcurrenttracks2.FunctionHandle=@getCurrentTracks;
getcurrenttracks2.FunctionArgs.OffsetFrame.Value=0;
getcurrenttracks2.FunctionArgs.TimeFrame.Value=TimeFrame;
getcurrenttracks2.FunctionArgs.TimeCol.Value=2;
getcurrenttracks2.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks2.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getcurrenttracks2.FunctionArgs.FrameStep.Value=FrameStep;
getcurrenttracks2.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
getcurrenttracks2.FunctionArgs.CurFrame.OutputArg='LoopCounter';
getcurrenttracks2.FunctionArgs.Tracks.FunctionInstance='ImageOverlayLoop';
```

```
getcurrenttracks2.FunctionArgs.Tracks.InputArg='RemoveShortTracks_Tracks';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,getcurrenttracks2);

makematnamesinoverlayloop.InstanceName='MakeMatNamesInOverlayLoop';
makematnamesinoverlayloop.FunctionHandle=@makeImgFileName;
makematnamesinoverlayloop.FunctionArgs.FileBase.Value=SegmentationFilesRoot;
makematnamesinoverlayloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makematnamesinoverlayloop.FunctionArgs.FileExt.Value='.mat';
makematnamesinoverlayloop.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
makematnamesinoverlayloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,makematnamesinoverla
yloop);

loadcellslabel.InstanceName='LoadCellsLabel';
loadcellslabel.FunctionHandle=@loadCellsLabel;
loadcellslabel.FunctionArgs.FileName.FunctionInstance='MakeMatNamesInOverlayLoop';
loadcellslabel.FunctionArgs.FileName.OutputArg='FileName';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,loadcellslabel);

loadcolormap.InstanceName='LoadColormap';
loadcolormap.FunctionHandle=@loadColormap;
loadcolormap.FunctionArgs.FileName.Value='colormap_lines';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,loadcolormap);

displayoverlayingframe.InstanceName='DisplayOverlayingFrame';
displayoverlayingframe.FunctionHandle=@displayVariable;
displayoverlayingframe.FunctionArgs.VariableName.Value='Overlaying Frame';
displayoverlayingframe.FunctionArgs.Variable.FunctionInstance='ImageOverlayLoop';
displayoverlayingframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,displayoverlayingfra
me);

displayancestry.InstanceName='DisplayAncestry';
displayancestry.FunctionHandle=@displayAncestryData;
displayancestry.FunctionArgs.NumberFormat.Value=NumberFormat;
displayancestry.FunctionArgs.ProlDir.Value=AncestryFolder;
displayancestry.FunctionArgs.ImageFileName.Value=ImageFilesRoot;
displayancestry.FunctionArgs.DS.Value='/';
displayancestry.FunctionArgs.Image.FunctionInstance='ReadImagesInOverlayLoop';
displayancestry.FunctionArgs.Image.OutputArg='Image';
displayancestry.FunctionArgs.CurrentTracks.FunctionInstance='GetCurrentTracks2';
displayancestry.FunctionArgs.CurrentTracks.OutputArg='Tracks';
displayancestry.FunctionArgs.CellsLabel.FunctionInstance='LoadCellsLabel';
displayancestry.FunctionArgs.CellsLabel.OutputArg='LabelMatrix';
displayancestry.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.CurFrame.OutputArg='LoopCounter';
displayancestry.FunctionArgs.ColorMap.FunctionInstance='LoadColormap';
displayancestry.FunctionArgs.ColorMap.OutputArg='Colormap';
displayancestry.FunctionArgs.CellsAncestry.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.CellsAncestry.InputArg='RemoveShortTracks_CellsAncestry';
displayancestry.FunctionArgs.TracksLayout.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
displayancestry.FunctionArgs.AncestryLayout.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.AncestryLayout.InputArg='LoadAncestryLayout_AncestryLayout';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,displayancestry);

imageoverlayloop.InstanceName='ImageOverlayLoop';
imageoverlayloop.FunctionHandle=@forLoop;
imageoverlayloop.FunctionArgs.StartLoop.Value=StartFrame;
imageoverlayloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
imageoverlayloop.FunctionArgs.IncrementLoop.Value=FrameStep;
imageoverlayloop.FunctionArgs.RemoveShortTracks_Tracks.FunctionInstance='RemoveShortTracks';
imageoverlayloop.FunctionArgs.RemoveShortTracks_Tracks.OutputArg='Tracks';
imageoverlayloop.FunctionArgs.RemoveShortTracks_CellsAncestry.FunctionInstance='RemoveShortTracks
';
imageoverlayloop.FunctionArgs.RemoveShortTracks_CellsAncestry.OutputArg='CellsAncestry';
imageoverlayloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='LoadTracksLayout';
imageoverlayloop.FunctionArgs.LoadTracksLayout_TracksLayout.OutputArg='TracksLayout';
imageoverlayloop.FunctionArgs.LoadAncestryLayout_AncestryLayout.FunctionInstance='LoadAncestryLay
out';
imageoverlayloop.FunctionArgs.LoadAncestryLayout_AncestryLayout.OutputArg='AncestryLayout';
imageoverlayloop.LoopFunctions=image_overlay_loop_functions;
functions_list=addToFunctionChain(functions_list,imageoverlayloop);

saveancestryspreadsheets.InstanceName='SaveAncestrySpreadsheets';
saveancestryspreadsheets.FunctionHandle=@saveAncestrySpreadsheets;
saveancestryspreadsheets.FunctionArgs.ShapesXlsFile.Value=ShapesSpreadsheet;
saveancestryspreadsheets.FunctionArgs.ProlXlsFile.Value=AncestrySpreadsheet;
saveancestryspreadsheets.FunctionArgs.Tracks.FunctionInstance='RemoveShortTracks';
saveancestryspreadsheets.FunctionArgs.Tracks.OutputArg='Tracks';
saveancestryspreadsheets.FunctionArgs.CellsAncestry.FunctionInstance='RemoveShortTracks';
```

```
saveancestryspreadsheets.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
saveancestryspreadsheets.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
saveancestryspreadsheets.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
functions_list=addToFunctionChain(functions_list,saveancestryspreadsheets);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% assayCellCoverage.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayCellCoverage()
%assayCellCoverage - This assay is used to determine what percentage of an image is occupied
% by objects.  ImageFileName - String variable specifying the absolute image file name
%of  the image to be  analyzed. ImageDirectory - String variable specifying the
%directory where  the image to be analyzed is located. MaskFileName - String variable
%specifying the file  name of the resulting binary image from which  the
%object percentage is calculated.  Important Modules - manualSegmentationReview.

global functions_list;
functions_list=[];
%script variables
ImageFileName='C:/walter/20071104/20071104 ha ht-1080 af488coll4 10ugperml  then af488coll4
lugperml milk 35 mm bac pd_t001.TIF';
ImageDirectory='C:/walter/20071104/';
MaskFileName=[ImageDirectory 'mask.mat'];
%end script variables


readimage.InstanceName='ReadImage';
readimage.FunctionHandle=@readImage;
readimage.FunctionArgs.ImageName.Value=ImageFileName;
readimage.FunctionArgs.ImageChannel.Value='';
functions_list=addToFunctionChain(functions_list,readimage);

normalizeimageto16bit.InstanceName='NormalizeImageTo16Bit';
normalizeimageto16bit.FunctionHandle=@imNorm;
normalizeimageto16bit.FunctionArgs.IntegerClass.Value='uint16';
normalizeimageto16bit.FunctionArgs.RawImage.FunctionInstance='ReadImage';
normalizeimageto16bit.FunctionArgs.RawImage.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,normalizeimageto16bit);

displaynormalizedimage.InstanceName='DisplayNormalizedImage';
displaynormalizedimage.FunctionHandle=@displayImage;
displaynormalizedimage.FunctionArgs.FigureNr.Value=1;
displaynormalizedimage.FunctionArgs.Image.FunctionInstance='NormalizeImageTo16Bit';
displaynormalizedimage.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,displaynormalizedimage);

imagetobw.InstanceName='ImageToBW';
imagetobw.FunctionHandle=@im2bw_Wrapper;
imagetobw.FunctionArgs.Image.FunctionInstance='NormalizeImageTo16Bit';
imagetobw.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,imagetobw);

negativeimage.InstanceName='NegativeImage';
negativeimage.FunctionHandle=@negativeImage;
negativeimage.FunctionArgs.Image.FunctionInstance='ImageToBW';
negativeimage.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,negativeimage);

displaynegativeimage.InstanceName='DisplayNegativeImage';
displaynegativeimage.FunctionHandle=@displayImage;
displaynegativeimage.FunctionArgs.FigureNr.Value=2;
displaynegativeimage.FunctionArgs.Image.FunctionInstance='NegativeImage';
displaynegativeimage.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,displaynegativeimage);

labelobjects.InstanceName='LabelObjects';
labelobjects.FunctionHandle=@labelObjects;
labelobjects.FunctionArgs.Image.FunctionInstance='NegativeImage';
labelobjects.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,labelobjects);
```

```
reviewsegmentation.InstanceName='ReviewSegmentation';
reviewsegmentation.FunctionHandle=@manualSegmentationReview;
reviewsegmentation.FunctionArgs.PreviousLabel.Value=[];
reviewsegmentation.FunctionArgs.ObjectsLabel.FunctionInstance='LabelObjects';
reviewsegmentation.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
reviewsegmentation.FunctionArgs.RawLabel.FunctionInstance='LabelObjects';
reviewsegmentation.FunctionArgs.RawLabel.OutputArg='LabelMatrix';
reviewsegmentation.FunctionArgs.Image.FunctionInstance='NormalizeImageTo16Bit';
reviewsegmentation.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,reviewsegmentation);

labeltobw.InstanceName='LabelToBW';
labeltobw.FunctionHandle=@compareValues;
labeltobw.FunctionArgs.Operation.Value='>';
labeltobw.FunctionArgs.Arg2.Value=0;
labeltobw.FunctionArgs.Arg1.FunctionInstance='ReviewSegmentation';
labeltobw.FunctionArgs.Arg1.OutputArg='LabelMatrix';
functions_list=addToFunctionChain(functions_list,labeltobw);

savebwimage.InstanceName='SaveBWImage';
savebwimage.FunctionHandle=@saveWrapper;
savebwimage.FunctionArgs.FileName.Value=MaskFileName;
savebwimage.FunctionArgs.SaveData.FunctionInstance='LabelToBW';
savebwimage.FunctionArgs.SaveData.OutputArg='BooleanOut';
functions_list=addToFunctionChain(functions_list,savebwimage);

percentageforeground.InstanceName='PercentageForeground';
percentageforeground.FunctionHandle=@percentageForeground;
percentageforeground.FunctionArgs.Image.FunctionInstance='LabelToBW';
percentageforeground.FunctionArgs.Image.OutputArg='BooleanOut';
functions_list=addToFunctionChain(functions_list,percentageforeground);

displaycellcoverage.InstanceName='DisplayCellCoverage';
displaycellcoverage.FunctionHandle=@displayVariable;
displaycellcoverage.FunctionArgs.VariableName.Value='Percentage of Cells';
displaycellcoverage.FunctionArgs.Variable.FunctionInstance='PercentageForeground';
displaycellcoverage.FunctionArgs.Variable.OutputArg='PercentageForeground';
functions_list=addToFunctionChain(functions_list,displaycellcoverage);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayCellPropsSingleImage.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayCellPropsSingleImage()
%assayCellPropsSingleImage - This assay is used to segment objects in an image subject to manual
%review then  extract their shape properties.  ImageDirName - String variable specifying where
%the directory where the image to be analyzed is  located. An example of
%such a variable would be 'c:/test/'. FileRoot - String variable specifying the image file
%name without the extension. For example, if  the file name is 'Experiment-
0002_Position(8)_t021.tif' the
%root file name will be 'Experiment-0002_Position(8)_t021'. ImageExt - String variable specifying
the extension of
%the image file name. For example, if  the file name is 'Experiment-0002_Position(8)_t021.tif'
the
%root file name will be '.tif'. ImageFileName - String variable that is automatically generated
%from the ImageDirName, FileRoot and ImageExt. OutputDir - String variable specifying the
directory where
%the spreadsheet containing the shape properties and  the label matrix containing the cell
%outlines will be saved. SpreadsheetFileName - String variable specifying the file name of the
%spreadsheet containing the shape properties  of the objects in the original image. This
%variable is automatically generated from the  OutputDir and FileRoot variables. LabelFileName -
String
%variable specifying the file name of the label matrix containing the detected  objects
%in the original image. This variable is automatically generated from the OutputDir and
%FileRoot variables. BrightnessGradientThreshold - Number specifying the threshold value for the
image generated by
%the generateBinImgUsingLocAvg filter.  Any pixels in the gradient image below this value will
%be set to zero  while the rest will be set to one. ClearBorder
%- Boolean value specifying whether objects next to or touching the image border in
% the binary images generated by the generateBinImgUsingGradient module will be erased (true) or
%not  (false). ClearBorderDist - Number specifying how close to the border objects may
%be and still be  erased if the ClearBorder parameter is set to true
```

```
%in the generateBinImgUsingGradient module. MedianFilterSize - Number specifying the size of the
median filter
%used by the distanceWatershed module.  Setting this to a higher integer value will
%reduce the number of objects detected  by the module and can be used
%to prevent oversegmentation. ObjectArea - Number specifying the threshold area for the
clearSmallObjects filter.
%Objects below this value  will be removed from the filtered image. Strel -
%String variable specifying the type of filter used to generate the local average
%image in generateBinImgUsingLocAvg. Currently 'disk' is the only value supported. StrelSize -
Number specifying
%the size of the local neighborhood used to calculate the average  for each
%pixel in the local average image generated by the generateBinImgUsingLocAvg module.  Important
Modules
%- clearSmallObjects, distanceWatershed, generateBinImgUsingLocAvg, manualSegmentationReview,
segmentObjectsUsingMarkers.

global functions_list;
functions_list=[];
%script variables
ImageDirName='C:/walter/20071104/';
FileRoot='20071104 ha ht-1080 af488coll4 10ugperml  then af488coll4 1ugperml milk 35 mm bac
pd_t001';
ImageExt='.TIF';
ImageFileName=[ImageDirName FileRoot ImageExt];
OutputDir='c:/walter/';
SpreadsheetFileName=[OutputDir FileRoot '.csv'];
LabelFileName=[OutputDir FileRoot '.mat'];
BrightnessThresholdPct=1.2;
ClearBorder=false;
ClearBorderDist=0;
MedianFilterSize=3;
ObjectArea=30;
Strel='disk';
StrelSize=10;
%end script variables


getfileinfo.InstanceName='GetFileInfo';
getfileinfo.FunctionHandle=@getFileInfo;
getfileinfo.FunctionArgs.DirSep.Value='/';
getfileinfo.FunctionArgs.PathName.Value=ImageFileName;
functions_list=addToFunctionChain(functions_list,getfileinfo);

readimage.InstanceName='ReadImage';
readimage.FunctionHandle=@readImage;
readimage.FunctionArgs.ImageName.Value=ImageFileName;
readimage.FunctionArgs.ImageChannel.Value='';
functions_list=addToFunctionChain(functions_list,readimage);

normalizeimageto16bit.InstanceName='NormalizeImageTo16Bit';
normalizeimageto16bit.FunctionHandle=@imNorm;
normalizeimageto16bit.FunctionArgs.IntegerClass.Value='uint16';
normalizeimageto16bit.FunctionArgs.RawImage.FunctionInstance='ReadImage';
normalizeimageto16bit.FunctionArgs.RawImage.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,normalizeimageto16bit);

displaynormalizedimage.InstanceName='DisplayNormalizedImage';
displaynormalizedimage.FunctionHandle=@displayImage;
displaynormalizedimage.FunctionArgs.FigureNr.Value=1;
displaynormalizedimage.FunctionArgs.Image.FunctionInstance='NormalizeImageTo16Bit';
displaynormalizedimage.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,displaynormalizedimage);

negativeimage.InstanceName='NegativeImage';
negativeimage.FunctionHandle=@imcomplementWrapper;
negativeimage.FunctionArgs.Image.FunctionInstance='NormalizeImageTo16Bit';
negativeimage.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,negativeimage);

localaveragingfilter.InstanceName='LocalAveragingFilter';
localaveragingfilter.FunctionHandle=@generateBinImgUsingLocAvg;
localaveragingfilter.FunctionArgs.Strel.Value=Strel;
localaveragingfilter.FunctionArgs.StrelSize.Value=StrelSize;
localaveragingfilter.FunctionArgs.BrightnessThresholdPct.Value=BrightnessThresholdPct;
localaveragingfilter.FunctionArgs.ClearBorder.Value=ClearBorder;
localaveragingfilter.FunctionArgs.ClearBorderDist.Value=ClearBorderDist;
localaveragingfilter.FunctionArgs.Image.FunctionInstance='NegativeImage';
localaveragingfilter.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,localaveragingfilter);

fillholesimage.InstanceName='FillHolesImage';
```

```
fillholesimage.FunctionHandle=@fillHoles;
fillholesimage.FunctionArgs.Image.FunctionInstance='LocalAveragingFilter';
fillholesimage.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,fillholesimage);

clearsmallobjects.InstanceName='ClearSmallObjects';
clearsmallobjects.FunctionHandle=@clearSmallObjects;
clearsmallobjects.FunctionArgs.MinObjectArea.Value=ObjectArea;
clearsmallobjects.FunctionArgs.Image.FunctionInstance='FillHolesImage';
clearsmallobjects.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,clearsmallobjects);

displaythresholdedimage.InstanceName='DisplayThresholdedImage';
displaythresholdedimage.FunctionHandle=@displayImage;
displaythresholdedimage.FunctionArgs.FigureNr.Value=2;
displaythresholdedimage.FunctionArgs.Image.FunctionInstance='ClearSmallObjects';
displaythresholdedimage.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,displaythresholdedimage);

labelobjects.InstanceName='LabelObjects';
labelobjects.FunctionHandle=@labelObjects;
labelobjects.FunctionArgs.Image.FunctionInstance='ClearSmallObjects';
labelobjects.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,labelobjects);

distancewatershed.InstanceName='DistanceWatershed';
distancewatershed.FunctionHandle=@distanceWatershed;
distancewatershed.FunctionArgs.MedianFilterNhood.Value=MedianFilterSize;
distancewatershed.FunctionArgs.Image.FunctionInstance='ClearSmallObjects';
distancewatershed.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,distancewatershed);

segmentobjectsusingmarkers.InstanceName='SegmentObjectsUsingMarkers';
segmentobjectsusingmarkers.FunctionHandle=@segmentObjectsUsingMarkers;
segmentobjectsusingmarkers.FunctionArgs.MarkersLabel.FunctionInstance='DistanceWatershed';
segmentobjectsusingmarkers.FunctionArgs.MarkersLabel.OutputArg='LabelMatrix';
segmentobjectsusingmarkers.FunctionArgs.ObjectsLabel.FunctionInstance='LabelObjects';
segmentobjectsusingmarkers.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
functions_list=addToFunctionChain(functions_list,segmentobjectsusingmarkers);

reviewsegmentation.InstanceName='ReviewSegmentation';
reviewsegmentation.FunctionHandle=@manualSegmentationReview;
reviewsegmentation.FunctionArgs.PreviousLabel.Value=[];
reviewsegmentation.FunctionArgs.ObjectsLabel.FunctionInstance='SegmentObjectsUsingMarkers';
reviewsegmentation.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
reviewsegmentation.FunctionArgs.RawLabel.FunctionInstance='LabelObjects';
reviewsegmentation.FunctionArgs.RawLabel.OutputArg='LabelMatrix';
reviewsegmentation.FunctionArgs.Image.FunctionInstance='NormalizeImageTo16Bit';
reviewsegmentation.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,reviewsegmentation);

getregionprops.InstanceName='GetRegionProps';
getregionprops.FunctionHandle=@getRegionProps;
getregionprops.FunctionArgs.LabelMatrix.FunctionInstance='ReviewSegmentation';
getregionprops.FunctionArgs.LabelMatrix.OutputArg='LabelMatrix';
functions_list=addToFunctionChain(functions_list,getregionprops);

saveregionprops.InstanceName='SaveRegionProps';
saveregionprops.FunctionHandle=@saveRegionPropsSpreadsheets;
saveregionprops.FunctionArgs.SpreadsheetFileName.Value=SpreadsheetFileName;
saveregionprops.FunctionArgs.RegionProps.FunctionInstance='GetRegionProps';
saveregionprops.FunctionArgs.RegionProps.OutputArg='RegionProps';
functions_list=addToFunctionChain(functions_list,saveregionprops);

savelabel.InstanceName='SaveLabel';
savelabel.FunctionHandle=@saveWrapper;
savelabel.FunctionArgs.FileName.Value=LabelFileName;
savelabel.FunctionArgs.SaveData.FunctionInstance='ReviewSegmentation';
savelabel.FunctionArgs.SaveData.OutputArg='LabelMatrix';
functions_list=addToFunctionChain(functions_list,savelabel);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayDescriptionGUI.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function varargout = assayDescriptionGUI(varargin)
% ASSAYDESCRIPTIONGUI M-file for assayDescriptionGUI.fig
%      ASSAYDESCRIPTIONGUI, by itself, creates a new ASSAYDESCRIPTIONGUI or raises the existing
%      singleton*.
%
%      H = ASSAYDESCRIPTIONGUI returns the handle to a new ASSAYDESCRIPTIONGUI or the handle to
%      the existing singleton*.
%
%      ASSAYDESCRIPTIONGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ASSAYDESCRIPTIONGUI.M with the given input arguments.
%
%      ASSAYDESCRIPTIONGUI('Property','Value',...) creates a new ASSAYDESCRIPTIONGUI or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before assayDescriptionGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to assayDescriptionGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help assayDescriptionGUI

% Last Modified by GUIDE v2.5 28-Sep-2011 16:55:20

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @assayDescriptionGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @assayDescriptionGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before assayDescriptionGUI is made visible.
function assayDescriptionGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to assayDescriptionGUI (see VARARGIN)

% Choose default command line output for assayDescriptionGUI
handles.output = hObject;
handles.OK=false;
mt_idx=find(strcmp(varargin, 'AssayDescription'))+1;
assay_description=varargin{mt_idx};
set(handles.editDescription,'String',assay_description);

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes assayDescriptionGUI wait for user response (see UIRESUME)
uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = assayDescriptionGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

varargout{1}=handles.OK;
if handles.OK
```

```
    varargout{2}=get(handles.editDescription,'String');
else
    varargout{2}='';
end
delete(hObject);


% --- Executes on button press in pushbuttonOK.
function pushbuttonOK_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonOK (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.OK=true;
guidata(handles.figure1,handles);
uiresume(handles.figure1);


% --- Executes on button press in pushbuttonCancel.
function pushbuttonCancel_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonCancel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
uiresume(handles.figure1);


function editDescription_Callback(hObject, eventdata, handles)
% hObject    handle to editDescription (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editDescription as text
%        str2double(get(hObject,'String')) returns contents of editDescription as a double


% --- Executes during object creation, after setting all properties.
function editDescription_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editDescription (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: delete(hObject) closes the figure
uiresume(hObject);




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% assayEditorGUI.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function varargout = assayEditorGUI(varargin)
% ASSAYEDITORGUI M-file for assayEditorGUI.fig
%      ASSAYEDITORGUI, by itself, creates a new ASSAYEDITORGUI or raises the existing
%      singleton*.
%
%      H = ASSAYEDITORGUI returns the handle to a new ASSAYEDITORGUI or the handle to
%      the existing singleton*.
%
%      ASSAYEDITORGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ASSAYEDITORGUI.M with the given input arguments.
%
%      ASSAYEDITORGUI('Property','Value',...) creates a new ASSAYEDITORGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before assayEditorGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to assayEditorGUI_OpeningFcn via varargin.
```

```
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help assayEditorGUI

% Last Modified by GUIDE v2.5 11-Oct-2011 15:39:28

% Begin initialization code - DO NOT EDIT
gui_Singleton = 0;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @assayEditorGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @assayEditorGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before assayEditorGUI is made visible.
function assayEditorGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to assayEditorGUI (see VARARGIN)

% Choose default command line output for assayEditorGUI
handles.output = hObject;
aeMenuNewAssay(hObject, eventdata, handles);
handles=guidata(hObject);

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes assayEditorGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = assayEditorGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --------------------------------------------------------------------
function menuFile_Callback(hObject, eventdata, handles)
% hObject    handle to menuFile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function menuNewAssay_Callback(hObject, eventdata, handles)
% hObject    handle to menuNewAssay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
aeMenuNewAssay(hObject, eventdata, handles);


% --------------------------------------------------------------------
function menuOpenAssay_Callback(hObject, eventdata, handles)
% hObject    handle to menuOpenAssay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
aeMenuOpenAssay(hObject, eventdata, handles);


% --------------------------------------------------------------------
function menuSaveAssay_Callback(hObject, eventdata, handles)
% hObject    handle to menuSaveAssay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if isempty(handles.CurrentAssay)
    [file path]=uiputfile('*.m','Save assay as:');
    saveAssay(handles,file,path);
else
    saveAssay(handles,handles.CurrentAssay,handles.AssayPath);
end



% --- Executes on selection change in listboxAvailableModules.
function listboxAvailableModules_Callback(hObject, eventdata, handles)
% hObject    handle to listboxAvailableModules (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listboxAvailableModules contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listboxAvailableModules
modules_list=get(hObject,'String');
selection_idx=get(hObject,'Value');
set(handles.editModuleDescription,'String',getModuleDescription(modules_list{selection_idx}));




% --- Executes during object creation, after setting all properties.
function listboxAvailableModules_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listboxAvailableModules (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
set(hObject,'String',getModuleList());



function editModuleDescription_Callback(hObject, eventdata, handles)
% hObject    handle to editModuleDescription (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editModuleDescription as text
%        str2double(get(hObject,'String')) returns contents of editModuleDescription as a double


% --- Executes during object creation, after setting all properties.
function editModuleDescription_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editModuleDescription (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
assay_list=getModuleList();
set(hObject,'String',getModuleDescription(assay_list{1}));


% --- Executes on selection change in listboxCurrentAssay.
function listboxCurrentAssay_Callback(hObject, eventdata, handles)
% hObject    handle to listboxCurrentAssay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
modules_list=get(hObject,'String');
if isempty(modules_list)
    return;
end
```

```
selection_idx=get(hObject,'Value');
selection_text=modules_list{selection_idx};
chain_struct=false;
if (strcmp(selection_text(1:9),'<html><i>'))
    chain_struct=true;
else
    %remove the html code
    module_id=stripHTMLFromString(selection_text);
end
if (chain_struct)
    set(handles.editModuleDescription,'String','List of SubModules.');
else
    modules_list=handles.ModulesList;
    modules_map=handles.ModulesMap;
    module_idx=modules_map.get(module_id);
    module_struct=modules_list{module_idx};
    set(handles.editModuleDescription,'String',getModuleDescription([module_struct.ModuleName
'.m']));
end

selection_type=get(handles.figure1,'SelectionType');

if strcmp(selection_type,'open')
    if (chain_struct)
        %expand/collapse sub-modules list

modules_list=manageChainText(modules_list,selection_idx,handles.ModulesList,handles.ModulesMap);
        set(hObject,'String',modules_list);
    else
        %show the arguments for the selected module
        modules_list=handles.ModulesList;
        modules_map=handles.ModulesMap;
        module_idx=modules_map.get(module_id);
        module_struct=modules_list{module_idx};
        [dlg_ok
module_struct]=inputArgumentsGUI('ModulesList',handles.ModulesList,'ModulesMap',handles.ModulesMa
p,'ModuleStruct',module_struct);
        if (dlg_ok)
            modules_list{module_idx}=module_struct;
            handles.ModulesList=modules_list;
            guidata(handles.figure1,handles);
        end
    end
end



% Hints: contents = get(hObject,'String') returns listboxCurrentAssay contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listboxCurrentAssay


% --- Executes during object creation, after setting all properties.
function listboxCurrentAssay_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listboxCurrentAssay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --------------------------------------------------------------------
function menuModuleParameters_Callback(hObject, eventdata, handles)
% hObject    handle to menuModuleParameters (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function assay_context_menu_Callback(hObject, eventdata, handles)
% hObject    handle to assay_context_menu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbuttonUp.
function pushbuttonUp_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonUp (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
moveModule(handles,-1);


% --- Executes on button press in pushbuttonDown.
function pushbuttonDown_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonDown (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
moveModule(handles,1);


% --- Executes on button press in pushbuttonAdd.
function pushbuttonAdd_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonAdd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
addModuleToAssay(handles);


% --- Executes during object creation, after setting all properties.
function pushbuttonUp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbuttonUp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
button_img=imread('up arrow.tif');
button_img=repmat(button_img,[1 1 3]);
set(hObject,'CData',button_img);


% --- Executes during object creation, after setting all properties.
function pushbuttonDown_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbuttonDown (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
button_img=imread('down arrow.tif');
button_img=repmat(button_img,[1 1 3]);
set(hObject,'CData',button_img);


% --- Executes during object creation, after setting all properties.
function pushbuttonAdd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbuttonAdd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
button_img=imread('right arrow.tif');
button_img=repmat(button_img,[1 1 3]);
set(hObject,'CData',button_img);


% --- Executes on button press in pushbuttonRemove.
function pushbuttonRemove_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonRemove (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
removeModuleFromAssay(handles);


% --- Executes during object creation, after setting all properties.
function pushbuttonRemove_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbuttonRemove (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
button_img=imread('remove.tif');
set(hObject,'CData',button_img);


% --------------------------------------------------------------------
function menuSaveAsAssay_Callback(hObject, eventdata, handles)
% hObject    handle to menuSaveAsAssay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[file path]=uiputfile('*.m','Save assay as:');
saveAssay(handles,file,path);



% --------------------------------------------------------------------
function menuTools_Callback(hObject, eventdata, handles)
% hObject    handle to menuTools (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function menuHelp_Callback(hObject, eventdata, handles)
% hObject    handle to menuHelp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function tagCellAnimationHelp_Callback(hObject, eventdata, handles)
% hObject    handle to tagCellAnimationHelp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
open('CellAnimation Help File.pdf');


% --------------------------------------------------------------------
function menuScriptVariables_Callback(hObject, eventdata, handles)
% hObject    handle to menuScriptVariables (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[dlg_ok script_variables]=scriptVariablesGUI('ScriptVariables',handles.ScriptVariables);
if (dlg_ok)
    handles.ScriptVariables=script_variables;
    guidata(handles.figure1,handles);
end


% --------------------------------------------------------------------
function menuWrapMatlabFunction_Callback(hObject, eventdata, handles)
% hObject    handle to menuWrapMatlabFunction (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
wrapFunction(handles);

% --------------------------------------------------------------------
function tagEdit_Callback(hObject, eventdata, handles)
% hObject    handle to tagEdit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function menuAssayDescription_Callback(hObject, eventdata, handles)
% hObject    handle to menuAssayDescription (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[dlg_ok assay_description]=assayDescriptionGUI('AssayDescription', handles.AssayDescription);
if (dlg_ok)
    handles.AssayDescription=assay_description;
    guidata(handles.figure1,handles);
end


% --------------------------------------------------------------------
function menuExportScriptVariables_Callback(hObject, eventdata, handles)
% hObject    handle to menuExportScriptVariables (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
exportScriptVariables(handles);


% --------------------------------------------------------------------
function menuImportScriptVariables_Callback(hObject, eventdata, handles)
% hObject    handle to menuImportScriptVariables (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
importScriptVariables(handles);


% --------------------------------------------------------------------
function menuRunAssay_Callback(hObject, eventdata, handles)
% hObject    handle to menuRunAssay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 cur_pointer=get(handles.figure1,'Pointer');
 set(handles.figure1,'Pointer','watch');
 drawnow();
```

```
assay_cmd=handles.CurrentAssay(1:(end-2));
clear functions;
try
    run(assay_cmd);
catch run_error
    set(handles.figure1,'Pointer',cur_pointer);
    rethrow(run_error);
end

set(handles.figure1,'Pointer',cur_pointer);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayFlCytoLNCapManSegWG.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayFlCytoLNCapManSegWG()
%assayFlCytoLNCapManSegWG - This assay is used to manually review segmentation of objects labeled
using a
% fluorescent dye  after they have been segmented using another assay.  ImageFolder
%-  String variable that specifies the absolute location of the directory which contains
%the   time-lapse images. An example of such a string variable would be
%'c:/sample images/high-density'. ImageFilesRoot  - String variable specifying the root image
file name. The
%root image file name   for a set of images is the image
%file name of any of  the  images without the number or the
%file extension. For example, if the  file name  is 'Experiment-0002_Position(8)_t021.tif' the
root
%image file name will be 'Experiment-0002_Position(8)_t'. ImageExtension  - String variable
specifying the image
%file extension including the preceding dot. For example   if the file name
%is 'image003.jpg' the image extension is '.jpg'. StartFrame -  Number specifying the first
%image in the sequence to be analyzed. The minimum   value for this
%variable depends on the numbering of the image sequence so if   the
%first image in the sequence is 'image003.tif' then the minimum value is  3.
%FrameCount - Number specifying how many images from the image sequence should be
%processed. FrameStep - Number specifying the step size when reading images. Set this variable
% to 1  to read every image in the sequence, 2 to read
%every  other image and  so on. NumberFormat - String value specifying the
%number of  digits in the image file names in  the sequence. For
%example if the  image file name is 'image020.jpg' the value for  the
%NumberFormat is '%03d', while  if the file name is 'image000020.jpg' the value should
% be '%06d'. OutputFolder -  The folder where the overlayed images and track
%data will be saved. By   default this value is set to a
%folder named 'output' within the folder where   the images to be analyzed
%are located. TracksFolder - The folder where the  label matrixes containing the cell
%outlines are saved. By  default this value is  set to a folder
%named 'track' within the output folder. SegmentationFilesRoot - The root  file name of
%the label matrixes containing the cell outlines. ImageFileBase - The path  name to
%the images. This value is generated from the ImageFolder  and the  ImageFilesRoot
%and should not be changed. ApproximationDistance - Number specifying how close the convex
%hull in the getConvexObjects module approximates the  object outline. By default this value
% is set to 2.5. Setting it to a  lower value will result
%in  convex hulls that more closely resemble the object outlines  however this
%increases the  chance of detecting insignificant concavities. BrightnessThresholdPct - Number
specifying the percentage
%threshold value for  the image generated by the generateBinImgUsingLocAvg  filter. Any pixel
%in the original image  smaller than the threshold value times the  corresponding
%value in the local average  image below this value will be set to
% zero while the rest will  be set to one. ClearBorder - Boolean
%value specifying whether objects next to or  touching the image border in
%the binary images generated by the generateBinImgUsingGradient module  will be erased (true) or
%not  (false). ClearBorderDist - Number specifying how close  to the border objects
%may be and still be  erased if the ClearBorder  parameter is set
%to true in the generateBinImgUsingGradient module. MedianFilterSize - Number specifying the
size of
%the median filter used by the distanceWatershed module.  Setting this to  a
%higher integer value will reduce the number of objects detected  by the
%module and can be used to prevent oversegmentation. ObjectArea - Number specifying the threshold
% area for the clearSmallObjects, polygonalAssistedWatershed filter. Objects below this  value
will be
%removed  from the filtered image. Strel - String variable specifying the type of
%filter used  to generate the local average  image in generateBinImgUsingLocAvg. Currently 'disk'
%is the only  value supported. StrelSize - Number specifying the size of the
%local neighborhood used to  calculate the average  for each pixel in the
%local average image generated by  the generateBinImgUsingLocAvg module. ResizeImageScale -
Number specifying by
%what ratio the images will be  resized before they are  processed. By
%default this value is set to 0.5.  Used by the resizeImage module. Important
```

```
%Modules - clearSmallObjects, distanceWatershed, generateBinImgUsingLocAvg, getConvexObjects,
manualSegmentationReview, polygonalAssistedWatershed,  segmentObjectsUsingMarkers.

global functions_list;
functions_list=[];
%script variables
ImageFolder='C:/sample movies/low density';
ImageFilesRoot='low density sample';
ImageExtension='.tif';
StartFrame=1;
FrameCount=10;
FrameStep=1;
NumberFormat='%06d';
OutputFolder=[ImageFolder '/output'];
TracksFolder=[OutputFolder '/track'];
SegmentationFilesRoot=[TracksFolder '/grayscale'];
ImageFileBase=[ImageFolder '/' ImageFilesRoot];
ApproximationDistance=2.4;
BrightnessThresholdPct=1.1;
ClearBorder=true;
ClearBorderDist=2;
MedianFilterSize=3;
ObjectArea=30;
Strel='disk';
StrelSize=10;
ResizeImageScale=0.5;
%end script variables

image_read_loop_functions=[];

displaycurframe.InstanceName='DisplayCurFrame';
displaycurframe.FunctionHandle=@displayVariable;
displaycurframe.FunctionArgs.VariableName.Value='Current Tracking Frame';
displaycurframe.FunctionArgs.Variable.FunctionInstance='SegmentationLoop';
displaycurframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,displaycurframe);

makeimagenamesinsegmentationloop.InstanceName='MakeImageNamesInSegmentationLoop';
makeimagenamesinsegmentationloop.FunctionHandle=@makeImgFileName;
makeimagenamesinsegmentationloop.FunctionArgs.FileBase.Value=ImageFileBase;
makeimagenamesinsegmentationloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinsegmentationloop.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makeimagenamesinsegmentati
onloop);

readimagesinsegmentationloop.InstanceName='ReadImagesInSegmentationLoop';
readimagesinsegmentationloop.FunctionHandle=@readImage;
readimagesinsegmentationloop.FunctionArgs.ImageChannel.Value='';
readimagesinsegmentationloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInSegmentatio
nLoop';
readimagesinsegmentationloop.FunctionArgs.ImageName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,readimagesinsegmentationlo
op);

normalizeimageto16bit.InstanceName='NormalizeImageTo16Bit';
normalizeimageto16bit.FunctionHandle=@imNorm;
normalizeimageto16bit.FunctionArgs.IntegerClass.Value='uint16';
normalizeimageto16bit.FunctionArgs.RawImage.FunctionInstance='ReadImagesInSegmentationLoop';
normalizeimageto16bit.FunctionArgs.RawImage.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,normalizeimageto16bit);

resizeimage.InstanceName='ResizeImage';
resizeimage.FunctionHandle=@resizeImage;
resizeimage.FunctionArgs.Scale.Value=ResizeImageScale;
resizeimage.FunctionArgs.Method.Value='bicubic';
resizeimage.FunctionArgs.Image.FunctionInstance='NormalizeImageTo16Bit';
resizeimage.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,resizeimage);

cytobrightnesslocalaveragingfilter.InstanceName='CytoBrightnessLocalAveragingFilter';
cytobrightnesslocalaveragingfilter.FunctionHandle=@generateBinImgUsingLocAvg;
cytobrightnesslocalaveragingfilter.FunctionArgs.Strel.Value=Strel;
cytobrightnesslocalaveragingfilter.FunctionArgs.StrelSize.Value=StrelSize;
cytobrightnesslocalaveragingfilter.FunctionArgs.BrightnessThresholdPct.Value=BrightnessThresholdP
ct;
cytobrightnesslocalaveragingfilter.FunctionArgs.ClearBorder.Value=ClearBorder;
cytobrightnesslocalaveragingfilter.FunctionArgs.ClearBorderDist.Value=ClearBorderDist;
cytobrightnesslocalaveragingfilter.FunctionArgs.Image.FunctionInstance='ResizeImage';
cytobrightnesslocalaveragingfilter.FunctionArgs.Image.OutputArg='Image';
```

```
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,cytobrightnesslocalaveragi
ngfilter);

fillholescytoplasmimages.InstanceName='FillHolesCytoplasmImages';
fillholescytoplasmimages.FunctionHandle=@fillHoles;
fillholescytoplasmimages.FunctionArgs.Image.FunctionInstance='CytoBrightnessLocalAveragingFilter'
;
fillholescytoplasmimages.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,fillholescytoplasmimages);

clearsmallcells.InstanceName='ClearSmallCells';
clearsmallcells.FunctionHandle=@clearSmallObjects;
clearsmallcells.FunctionArgs.MinObjectArea.Value=ObjectArea;
clearsmallcells.FunctionArgs.Image.FunctionInstance='FillHolesCytoplasmImages';
clearsmallcells.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,clearsmallcells);

nuclbrightnesslocalaveragingfilter.InstanceName='NuclBrightnessLocalAveragingFilter';
nuclbrightnesslocalaveragingfilter.FunctionHandle=@generateBinImgUsingLocAvg;
nuclbrightnesslocalaveragingfilter.FunctionArgs.Strel.Value=Strel;
nuclbrightnesslocalaveragingfilter.FunctionArgs.StrelSize.Value=StrelSize;
nuclbrightnesslocalaveragingfilter.FunctionArgs.BrightnessThresholdPct.Value=BrightnessThresholdP
ct;
nuclbrightnesslocalaveragingfilter.FunctionArgs.ClearBorder.Value=ClearBorder;
nuclbrightnesslocalaveragingfilter.FunctionArgs.ClearBorderDist.Value=ClearBorderDist;
nuclbrightnesslocalaveragingfilter.FunctionArgs.Image.FunctionInstance='ResizeImage';
nuclbrightnesslocalaveragingfilter.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,nuclbrightnesslocalaveragi
ngfilter);

fillholesnuclearimages.InstanceName='FillHolesNuclearImages';
fillholesnuclearimages.FunctionHandle=@fillHoles;
fillholesnuclearimages.FunctionArgs.Image.FunctionInstance='NuclBrightnessLocalAveragingFilter';
fillholesnuclearimages.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,fillholesnuclearimages);

clearsmallnuclei.InstanceName='ClearSmallNuclei';
clearsmallnuclei.FunctionHandle=@clearSmallObjects;
clearsmallnuclei.FunctionArgs.MinObjectArea.Value=ObjectArea;
clearsmallnuclei.FunctionArgs.Image.FunctionInstance='FillHolesNuclearImages';
clearsmallnuclei.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,clearsmallnuclei);

combinenuclearandcytoplasmimages.InstanceName='CombineNuclearAndCytoplasmImages';
combinenuclearandcytoplasmimages.FunctionHandle=@combineImages;
combinenuclearandcytoplasmimages.FunctionArgs.CombineOperation.Value='AND';
combinenuclearandcytoplasmimages.FunctionArgs.Image1.FunctionInstance='ClearSmallNuclei';
combinenuclearandcytoplasmimages.FunctionArgs.Image1.OutputArg='Image';
combinenuclearandcytoplasmimages.FunctionArgs.Image2.FunctionInstance='ClearSmallCells';
combinenuclearandcytoplasmimages.FunctionArgs.Image2.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,combinenuclearandcytoplasm
images);

reconstructcytoplasmimage.InstanceName='ReconstructCytoplasmImage';
reconstructcytoplasmimage.FunctionHandle=@reconstructObjects;
reconstructcytoplasmimage.FunctionArgs.GuideImage.FunctionInstance='CombineNuclearAndCytoplasmIma
ges';
reconstructcytoplasmimage.FunctionArgs.GuideImage.OutputArg='Image';
reconstructcytoplasmimage.FunctionArgs.ImageToReconstruct.FunctionInstance='ClearSmallNuclei';
reconstructcytoplasmimage.FunctionArgs.ImageToReconstruct.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,reconstructcytoplasmimage)
;

labelnuclei.InstanceName='LabelNuclei';
labelnuclei.FunctionHandle=@labelObjects;
labelnuclei.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
labelnuclei.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,labelnuclei);

labelcytoplasm.InstanceName='LabelCytoplasm';
labelcytoplasm.FunctionHandle=@labelObjects;
labelcytoplasm.FunctionArgs.Image.FunctionInstance='ReconstructCytoplasmImage';
labelcytoplasm.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,labelcytoplasm);

getconvexobjects.InstanceName='GetConvexObjects';
getconvexobjects.FunctionHandle=@getConvexObjects;
getconvexobjects.FunctionArgs.ApproximationDistance.Value=ApproximationDistance;
getconvexobjects.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
getconvexobjects.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getconvexobjects);
```

```
distancewatershed.InstanceName='DistanceWatershed';
distancewatershed.FunctionHandle=@distanceWatershed;
distancewatershed.FunctionArgs.MedianFilterNhood.Value=MedianFilterSize;
distancewatershed.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
distancewatershed.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,distancewatershed);

polygonalassistedwatershed.InstanceName='PolygonalAssistedWatershed';
polygonalassistedwatershed.FunctionHandle=@polygonalAssistedWatershed;
polygonalassistedwatershed.FunctionArgs.MinBlobArea.Value=ObjectArea;
polygonalassistedwatershed.FunctionArgs.ImageLabel.FunctionInstance='LabelNuclei';
polygonalassistedwatershed.FunctionArgs.ImageLabel.OutputArg='LabelMatrix';
polygonalassistedwatershed.FunctionArgs.WatershedLabel.FunctionInstance='DistanceWatershed';
polygonalassistedwatershed.FunctionArgs.WatershedLabel.OutputArg='LabelMatrix';
polygonalassistedwatershed.FunctionArgs.ConvexObjectsIndex.FunctionInstance='GetConvexObjects';
polygonalassistedwatershed.FunctionArgs.ConvexObjectsIndex.OutputArg='ConvexObjectsIndex';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,polygonalassistedwatershed
);

segmentobjectsusingmarkers.InstanceName='SegmentObjectsUsingMarkers';
segmentobjectsusingmarkers.FunctionHandle=@segmentObjectsUsingMarkers;
segmentobjectsusingmarkers.FunctionArgs.MarkersLabel.FunctionInstance='PolygonalAssistedWatershed
';
segmentobjectsusingmarkers.FunctionArgs.MarkersLabel.OutputArg='LabelMatrix';
segmentobjectsusingmarkers.FunctionArgs.ObjectsLabel.FunctionInstance='LabelCytoplasm';
segmentobjectsusingmarkers.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,segmentobjectsusingmarkers
);

getpreviousframenr.InstanceName='GetPreviousFrameNr';
getpreviousframenr.FunctionHandle=@addFunction;
getpreviousframenr.FunctionArgs.Number2.Value=-1;
getpreviousframenr.FunctionArgs.Number1.FunctionInstance='SegmentationLoop';
getpreviousframenr.FunctionArgs.Number1.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getpreviousframenr);

makelabelnames.InstanceName='MakeLabelNames';
makelabelnames.FunctionHandle=@makeImgFileName;
makelabelnames.FunctionArgs.FileBase.Value=SegmentationFilesRoot;
makelabelnames.FunctionArgs.NumberFmt.Value=NumberFormat;
makelabelnames.FunctionArgs.FileExt.Value='.mat';
makelabelnames.FunctionArgs.CurFrame.FunctionInstance='GetPreviousFrameNr';
makelabelnames.FunctionArgs.CurFrame.OutputArg='Sum';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makelabelnames);

loadpreviouslabel.InstanceName='LoadPreviousLabel';
loadpreviouslabel.FunctionHandle=@loadCellsLabel;
loadpreviouslabel.FunctionArgs.FileName.FunctionInstance='MakeLabelNames';
loadpreviouslabel.FunctionArgs.FileName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,loadpreviouslabel);

resizepreviouslabel.InstanceName='ResizePreviousLabel';
resizepreviouslabel.FunctionHandle=@resizeImage;
resizepreviouslabel.FunctionArgs.Scale.Value=ResizeImageScale;
resizepreviouslabel.FunctionArgs.Method.Value='nearest';
resizepreviouslabel.FunctionArgs.Image.FunctionInstance='LoadPreviousLabel';
resizepreviouslabel.FunctionArgs.Image.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,resizepreviouslabel);

refinesegmentation.InstanceName='RefineSegmentation';
refinesegmentation.FunctionHandle=@refineSegmentation;
refinesegmentation.FunctionArgs.CurrentLabel.FunctionInstance='SegmentObjectsUsingMarkers';
refinesegmentation.FunctionArgs.CurrentLabel.OutputArg='LabelMatrix';
refinesegmentation.FunctionArgs.PreviousLabel.FunctionInstance='ResizePreviousLabel';
refinesegmentation.FunctionArgs.PreviousLabel.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,refinesegmentation);

reviewsegmentation.InstanceName='ReviewSegmentation';
reviewsegmentation.FunctionHandle=@manualSegmentationReview;
reviewsegmentation.FunctionArgs.ObjectsLabel.FunctionInstance='RefineSegmentation';
reviewsegmentation.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
reviewsegmentation.FunctionArgs.RawLabel.FunctionInstance='SegmentObjectsUsingMarkers';
reviewsegmentation.FunctionArgs.RawLabel.OutputArg='LabelMatrix';
reviewsegmentation.FunctionArgs.PreviousLabel.FunctionInstance='ResizePreviousLabel';
reviewsegmentation.FunctionArgs.PreviousLabel.OutputArg='Image';
reviewsegmentation.FunctionArgs.Image.FunctionInstance='ResizeImage';
reviewsegmentation.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,reviewsegmentation);

resizecytolabel.InstanceName='ResizeCytoLabel';
```

```
resizecytolabel.FunctionHandle=@resizeImage;
resizecytolabel.FunctionArgs.Scale.Value=1/ResizeImageScale;
resizecytolabel.FunctionArgs.Method.Value='nearest';
resizecytolabel.FunctionArgs.Image.FunctionInstance='ReviewSegmentation';
resizecytolabel.FunctionArgs.Image.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,resizecytolabel);

savecellslabel.InstanceName='SaveCellsLabel';
savecellslabel.FunctionHandle=@saveCellsLabel;
savecellslabel.FunctionArgs.FileRoot.Value=SegmentationFilesRoot;
savecellslabel.FunctionArgs.NumberFormat.Value=NumberFormat;
savecellslabel.FunctionArgs.CellsLabel.FunctionInstance='ResizeCytoLabel';
savecellslabel.FunctionArgs.CellsLabel.OutputArg='Image';
savecellslabel.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
savecellslabel.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,savecellslabel);

segmentationloop.InstanceName='SegmentationLoop';
segmentationloop.FunctionHandle=@forLoop;
segmentationloop.FunctionArgs.StartLoop.Value=StartFrame;
segmentationloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
segmentationloop.FunctionArgs.IncrementLoop.Value=FrameStep;
segmentationloop.LoopFunctions=image_read_loop_functions;
functions_list=addToFunctionChain(functions_list,segmentationloop);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayFlCytoLNCapTracksReviewWG.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayFlCytoLNCapTracksReviewWG()
%assayFlCytoLNCapTracksReviewWG - This assay is used to manually review the automatic tracks
generated using a
%  tracking assay.  ImageFolder - String variable that specifies the absolute location
%of  the  directory which contains the  time-lapse images. An example of
%such a  string variable  would be 'c:/sample images/high-density'. ImageFilesRoot - String
variable
%specifying the root  image file name.  The root image file name
%for a set of  images is the image  file name of any
%of the  images without  the number or the file  extension. For
%example, if the file name   is 'Experiment-0002_Position(8)_t021.tif' the root image file
%name will be 'Experiment-0002_Position(8)_t'. ImageExtension - String  variable specifying the
image file extension
%including  the preceding dot. For example   if the file name is
%'image003.jpg' the image  extension is '.jpg'. StartFrame -  Number specifying the first
%image in the sequence to  be analyzed. The minimum   value for
%this variable depends on the numbering of  the image sequence  so if
% the first image in the sequence is 'image003.tif'  then the  minimum
%value is 3. FrameCount - Number specifying how many images from  the
%image sequence should be processed. TimeFrame - Number specifying the time between consecutive
% images in minutes. FrameStep - Number specifying the step size when reading images.
%Set   this variable to 1  to read every image in the
%sequence, 2  to  read every other image and  so on. NumberFormat
%- String value  specifying the  number of digits in the image file
%names in  the  sequence. For example  if the image file name
%is 'image020.jpg' the value for   the NumberFormat is  '%03d', while if
%the file name is 'image000020.jpg' the  value should  be '%06d'.  MaxFramesMissing
%- Number specifying for how many frames  a cell may be disappear before
% its  track is ended. OutputFolder -  The folder where the overlayed
%images and  track data will be saved. By   default this value
%is set to a  folder named 'output' within the  folder where
%the images to be analyzed are  located. AncestryFolder - The  folder where
%the overlayed images and ancestry data will be  saved. By   default
%this value is set to a folder named 'ancestry' within  the output
%folder. AncestrySpreadsheet - The path name to the spreadsheet containing the ancestry  data.
% By default this  value is set to a file named 'ancestry.csv' within
%  the ancestry folder. ShapesSpreadsheet - The path name to the spreadsheet containing
%the position   and shape properties for  each cell in the timelapse
%sequence at every  time  point. By default this is  set to
%to a file named  'shapes.csv' within  the ancestry folder. TracksFolder - The
%folder where the label matrixes  containing the cell  outlines are saved. By
% default this value is set  to a folder named  'track' within
%the output folder. SegmentationFilesRoot - The root  file name of the label
```

```
%matrixes containing the cell outlines. ImageFileBase - The  path name to the images.
%This  value is generated from the ImageFolder   and the ImageFilesRoot and
%should not be  changed. Important Modules - manualTrackingReview.

global functions_list;
functions_list=[];
%script variables
ImageFolder='C:/sample movies/low density';
ImageFilesRoot='low density sample';
ImageExtension='.tif';
StartFrame=1;
FrameCount=71;
TimeFrame=15;
FrameStep=1;
NumberFormat='%06d';
MaxFramesMissing=6;
OutputFolder=[ImageFolder '/output'];
AncestryFolder=[OutputFolder '/ancestry'];
AncestrySpreadsheet=[AncestryFolder 'ancestry.csv'];
ShapesSpreadsheet=[AncestryFolder 'shapes.csv'];
TracksFolder=[OutputFolder '/track'];
SegmentationFilesRoot=[TracksFolder '/grayscale'];
ImageFileBase=[ImageFolder '/' ImageFilesRoot];
%end script variables


loadtrackslayout.InstanceName='LoadTracksLayout';
loadtrackslayout.FunctionHandle=@loadTracksLayout;
loadtrackslayout.FunctionArgs.FileName.Value='tracks_layout.mat';
functions_list=addToFunctionChain(functions_list,loadtrackslayout);

loadtracks.InstanceName='LoadTracks';
loadtracks.FunctionHandle=@loadTracks;
loadtracks.FunctionArgs.FileName.Value=[AncestryFolder '/tracks.mat'];
functions_list=addToFunctionChain(functions_list,loadtracks);

loadancestry.InstanceName='LoadAncestry';
loadancestry.FunctionHandle=@loadAncestry;
loadancestry.FunctionArgs.FileName.Value=[AncestryFolder '/ancestry.mat'];
functions_list=addToFunctionChain(functions_list,loadancestry);

loadcolormap.InstanceName='LoadColormap';
loadcolormap.FunctionHandle=@loadColormap;
loadcolormap.FunctionArgs.FileName.Value='colormap_lines';
functions_list=addToFunctionChain(functions_list,loadcolormap);

loadancestrylayout.InstanceName='LoadAncestryLayout';
loadancestrylayout.FunctionHandle=@loadAncestryLayout;
loadancestrylayout.FunctionArgs.FileName.Value='ancestry_layout.mat';
functions_list=addToFunctionChain(functions_list,loadancestrylayout);

manualtracksreview.InstanceName='ManualTracksReview';
manualtracksreview.FunctionHandle=@manualTrackingReview;
manualtracksreview.FunctionArgs.ImageFileBase.Value=ImageFileBase;
manualtracksreview.FunctionArgs.NumberFormat.Value=NumberFormat;
manualtracksreview.FunctionArgs.ImgExt.Value=ImageExtension;
manualtracksreview.FunctionArgs.TimeFrame.Value=TimeFrame;
manualtracksreview.FunctionArgs.TimeCol.Value=2;
manualtracksreview.FunctionArgs.TrackIDCol.Value=1;
manualtracksreview.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
manualtracksreview.FunctionArgs.FrameStep.Value=FrameStep;
manualtracksreview.FunctionArgs.SegFileRoot.Value=SegmentationFilesRoot;
manualtracksreview.FunctionArgs.FrameCount.Value=FrameCount;
manualtracksreview.FunctionArgs.StartFrame.Value=StartFrame;
manualtracksreview.FunctionArgs.Tracks.FunctionInstance='LoadTracks';
manualtracksreview.FunctionArgs.Tracks.OutputArg='Tracks';
manualtracksreview.FunctionArgs.CellsAncestry.FunctionInstance='LoadAncestry';
manualtracksreview.FunctionArgs.CellsAncestry.OutputArg='Ancestry';
manualtracksreview.FunctionArgs.ColorMap.FunctionInstance='LoadColormap';
manualtracksreview.FunctionArgs.ColorMap.OutputArg='Colormap';
manualtracksreview.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
manualtracksreview.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
manualtracksreview.FunctionArgs.AncestryLayout.FunctionInstance='LoadAncestryLayout';
manualtracksreview.FunctionArgs.AncestryLayout.OutputArg='AncestryLayout';
functions_list=addToFunctionChain(functions_list,manualtracksreview);

saveupdatedtracks.InstanceName='SaveUpdatedTracks';
saveupdatedtracks.FunctionHandle=@saveTracks;
saveupdatedtracks.FunctionArgs.TracksFileName.Value=[AncestryFolder '/tracks.mat'];
saveupdatedtracks.FunctionArgs.Tracks.FunctionInstance='ManualTracksReview';
saveupdatedtracks.FunctionArgs.Tracks.OutputArg='Tracks';
```

```
functions_list=addToFunctionChain(functions_list,saveupdatedtracks);

saveancestry.InstanceName='SaveAncestry';
saveancestry.FunctionHandle=@saveAncestry;
saveancestry.FunctionArgs.AncestryFileName.Value=[AncestryFolder '/ancestry.mat'];
saveancestry.FunctionArgs.CellsAncestry.FunctionInstance='ManualTracksReview';
saveancestry.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
functions_list=addToFunctionChain(functions_list,saveancestry);

saveancestryspreadsheets.InstanceName='SaveAncestrySpreadsheets';
saveancestryspreadsheets.FunctionHandle=@saveAncestrySpreadsheets;
saveancestryspreadsheets.FunctionArgs.ShapesXlsFile.Value=ShapesSpreadsheet;
saveancestryspreadsheets.FunctionArgs.ProlXlsFile.Value=AncestrySpreadsheet;
saveancestryspreadsheets.FunctionArgs.Tracks.FunctionInstance='ManualTracksReview';
saveancestryspreadsheets.FunctionArgs.Tracks.OutputArg='Tracks';
saveancestryspreadsheets.FunctionArgs.CellsAncestry.FunctionInstance='ManualTracksReview';
saveancestryspreadsheets.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
saveancestryspreadsheets.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
saveancestryspreadsheets.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
functions_list=addToFunctionChain(functions_list,saveancestryspreadsheets);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayFlCytoLNCapTrackWG.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayFlCytoLNCapTrackWG()
%assayFlCytoLNCapTrackWG - This assay is used to automatically track cells that have been
segmented using
% another assay.  ImageFolder - String variable that specifies the absolute location of
%the  directory which contains the  time-lapse images. An example of such a
%string variable  would be 'c:/sample images/high-density'. ImageFilesRoot - String variable
specifying the root
%image file name.  The root image file name  for a set of
%images is the image  file name of any of the  images without
%the number or the file  extension. For example, if the file name
%is 'Experiment-0002_Position(8)_t021.tif' the root image file  name will be 'Experiment-
0002_Position(8)_t'. ImageExtension - String
%variable specifying the image file extension including  the preceding dot. For example
%if the file name is 'image003.jpg' the image  extension is '.jpg'. StartFrame -
%Number specifying the first image in the sequence to  be analyzed. The minimum
% value for this variable depends on the numbering of  the image sequence
%so if  the first image in the sequence is 'image003.tif'  then the
%minimum value is 3. FrameCount - Number specifying how many images from  the
%image sequence should be processed. TimeFrame - Number specifying the time between consecutive
%images in minutes. FrameStep - Number specifying the step size when reading images. Set
% this variable to 1  to read every image in the sequence, 2
%to  read every other image and  so on. NumberFormat - String value
%specifying the  number of digits in the image file names in  the
%sequence. For example  if the image file name is 'image020.jpg' the value for
% the NumberFormat is  '%03d', while if the file name is 'image000020.jpg' the
%value should  be '%06d'.  MaxFramesMissing - Number specifying for how many frames
%a cell may be disappear before  its  track is ended. OutputFolder -
%The folder where the overlayed images and  track data will be saved. By
% default this value is set to a  folder named 'output' within the
%folder where  the images to be analyzed are  located. AncestryFolder - The
%folder where the overlayed images and ancestry data will be  saved. By
%default this value is set to a folder named 'ancestry' within  the output
%folder. AncestrySpreadsheet - The path name to the spreadsheet containing the ancestry  data.
%By default this  value is set to a file named 'ancestry.csv' within
%the ancestry folder. ShapesSpreadsheet - The path name to the spreadsheet containing the
position
% and shape properties for  each cell in the timelapse sequence at every
%time  point. By default this is  set to to a file named
%'shapes.csv' within  the ancestry folder. TracksFolder - The folder where the label matrixes
%containing the cell  outlines are saved. By  default this value is set
%to a folder named  'track' within the output folder. SegmentationFilesRoot - The root
%file name of the label  matrixes containing the cell outlines. ImageFileBase - The
%path name to the images. This  value is generated from the ImageFolder
%and the ImageFilesRoot and should not be  changed. MaxSearchRadius - Number specifying the
%absolute lower bound for the search radius to  prevent selecting  too few
%candidate objects for a track. Used by assignCellToTrackUsingAll module.  MinSearchRadius -
Number specifying
```

```
%the absolute higher bound for the search radius to prevent  selecting  too
%many candidate objects for a track. Used by assignCellToTrackUsingAll module. MinSecondDistance
- Number
%specifying the minimum significant distance between the closest candidate object to a
%track and the second closest. Used to determine when distance should be used
%as  a ranking parameter. Used by assignCellToTrackUsingAll module. MaxDistRatio - Number
specifying the
% maximum allowed distance ratio between the two nearest candidate objects.  If the
%ratio  is higher than this value distance ranking will not be used.
%Used by  assignCellToTrackUsingAll module. MaxAngleDiff - Number specifying the maximum allowed
angle difference
%between a track  and a candidate  object. If the angle is larger
%than this value direction  ranking will not be  used for this object.
%Used by assignCellToTrackUsingAll module. NrParamsForSureMatch  - Number specifying the minimum
number of closest
%matches between a candidate object parameters   and a track's object parameters that
%make the candidate object a sure match  to  the track. Used by
%assignCellToTrackUsingAll module. SearchRadiusPct - Number specifying the size  of the
neighborhood from which
%candidate objects for matching  the track are selected.  It is a multiple
%of the distance to the nearest  candidate in the  current frame. Setting
%this variable equal to 1 turns this module  into a  nearest-neighbor algorithm
%(only the nearest cell can be a candidate). It does  not  make
%sense to have a value lower than 1. Used by assignCellToTrackUsingAll module. MaxSplitArea
%- Number specifying the maximum area a nucleus may be and still be considered
%  as a part of a possible mitotic event. Used by detectMitoticEvents module.
%MaxSplitDistance  - Number specifying the maximum distance a new nucleus may be from
%another nucleus   and still be considered as part of a possible mitotic
%event. Used by  detectMitoticEvents  module. MinSplitEccentricity - Number specifying the
minimum eccentricity
%a new nucleus may  have and still be  considered as part of
%a possible mitotic event. Used  by detectMitoticEvents module. MaxSplitEccentricity - Number
specifying the
%maximum eccentricity a new nucleus may  have and still be  considered as
%part of a possible mitotic event. Used  by detectMitoticEvents module. MinTimeForSplit - Number
%specifying the minimum time in minutes a track  needs to exist before
%it is considered for a possible mitotic event. Used  by detectMitoticEvents module. MinLifespan
%- Number specifying the minimum length in frames a frame  has to be
%to  not be removed by the removeShortTracks module. FrontParams -  Numeric array
%specifying a set of column indices from the shape and motility   parameters
%matrix. The parameters in those columns will be heavily weighted, and have more
% influence in determining the best match for a track from a list of
% objects.  Used by assignCellToTrackUsingAll module. DefaultParamWeights - Numeric array
specifying a set
%of  weights that is assigned to each shape  and motility parameter based
%on its  prediction power. Parameters with high prediction power are  assigned high
%weights and parameters  with low prediction power are assigned lower weights. Used
%by assignCellToTrackUsingAll module. DistanceRankingOrder  - Numeric array specifying the
default order of shape
%and motility parameters for slow   moving objects when it cannot be determined
%based on prediction power. Used by  assignCellToTrackUsingAll  module. DirectionRankingOrder -
Numeric array
%specifying the default order of shape and  motility parameters for fast  moving
%directional objects when it cannot be determined based  on prediction power. Used by
% assignCellToTrackUsingAll module. RelevantParametersIndex - Boolean array specifying column
indexes in the shape
%and motility matrix that have  been determined to be  irrelevant for tracking.
%This indicates to the module not to  use the parameters  those columns
%in computing track assignment probabilities. The order of column  indexes is  provided
%in TracksLayout variable. Used by assignCellToTrackUsingAll module. UnknownParamWeights -
Numeric array specifying a
%set of weights to be assigned to shape and  motility parameters when the
% prediction power of the parameters cannot be determined. Used by  assignCellToTrackUsingAll
module.
%UnknownRankingOrder  - Numeric array specifying the order of the shape and motility parameters
%when their   predictive power cannot be determined. If the objects cannot be
%categorized as either  slow-moving  or fast directional the parameter order provided in
%this variable is used.  Used by  assignCellToTrackUsingAll module. Important Modules -
assignCellToTrackUsingAll,
%detectMitoticEvents, splitTracks.

global functions_list;
functions_list=[];
%script variables
ImageFolder='C:/sample movies/low density';
ImageFilesRoot='low density sample';
ImageExtension='.tif';
StartFrame=1;
FrameCount=10;
TimeFrame=15;
FrameStep=1;
```

```
NumberFormat='%06d';
MaxFramesMissing=6;
OutputFolder=[ImageFolder '/output'];
AncestryFolder=[OutputFolder '/ancestry'];
AncestrySpreadsheet=[AncestryFolder 'ancestry.csv'];
ShapesSpreadsheet=[AncestryFolder 'shapes.csv'];
TracksFolder=[OutputFolder '/track'];
SegmentationFilesRoot=[TracksFolder '/grayscale'];
ImageFileBase=[ImageFolder '/' ImageFilesRoot];
MaxSearchRadius=Inf;
MinSearchRadius=0;
MinSecondDistance=5;
MaxDistRatio=0.6;
MaxAngleDiff=0.35;
NrParamsForSureMatch=5;
SearchRadiusPct=1.5;
MaxMergeDistance=23;
MaxSplitArea=400;
MaxSplitDistance=45;
MinSplitEccentricity=0.5;
MaxSplitEccentricity=0.95;
MinTimeForSplit=900;
MinLifespan=30;
FrontParams=[];
DefaultParamWeights=[34 21 13 8 5 3 2 2 2];
DistanceRankingOrder=[1 3 4 5 6 7 8 9 2];
DirectionRankingOrder=[2 3 4 5 6 7 8 9 1];
RelevantParametersIndex=[true true true false true false true true false];
UnknownParamWeights=[5 3 1 1 1 1 1 1 1];
UnknownRankingOrder=[1 2 3 4 5 6 7 8 9];
%end script variables

assign_cells_to_tracks_functions=[];
else_is_empty_cells_label_functions=[];
if_is_empty_cells_label_functions=[];
image_read_loop_functions=[];
image_overlay_loop_functions=[];

loadtrackslayout.InstanceName='LoadTracksLayout';
loadtrackslayout.FunctionHandle=@loadTracksLayout;
loadtrackslayout.FunctionArgs.FileName.Value='tracks_layout.mat';
functions_list=addToFunctionChain(functions_list,loadtrackslayout);

loadancestrylayout.InstanceName='LoadAncestryLayout';
loadancestrylayout.FunctionHandle=@loadAncestryLayout;
loadancestrylayout.FunctionArgs.FileName.Value='ancestry_layout.mat';
functions_list=addToFunctionChain(functions_list,loadancestrylayout);

displaycurframe.InstanceName='DisplayCurFrame';
displaycurframe.FunctionHandle=@displayVariable;
displaycurframe.FunctionArgs.VariableName.Value='Current Tracking Frame';
displaycurframe.FunctionArgs.Variable.FunctionInstance='SegmentationLoop';
displaycurframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,displaycurframe);

makelabelname.InstanceName='MakeLabelName';
makelabelname.FunctionHandle=@makeImgFileName;
makelabelname.FunctionArgs.FileBase.Value=SegmentationFilesRoot;
makelabelname.FunctionArgs.FileExt.Value='.mat';
makelabelname.FunctionArgs.NumberFmt.Value=NumberFormat;
makelabelname.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
makelabelname.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makelabelname);

loadlabelmatrix.InstanceName='LoadLabelMatrix';
loadlabelmatrix.FunctionHandle=@loadCellsLabel;
loadlabelmatrix.FunctionArgs.FileName.FunctionInstance='MakeLabelName';
loadlabelmatrix.FunctionArgs.FileName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,loadlabelmatrix);

getshapeparameters.InstanceName='GetShapeParameters';
getshapeparameters.FunctionHandle=@getShapeParams;
getshapeparameters.FunctionArgs.LabelMatrix.FunctionInstance='LoadLabelMatrix';
getshapeparameters.FunctionArgs.LabelMatrix.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getshapeparameters);

isemptypreviouscellslabel.InstanceName='IsEmptyPreviousCellsLabel';
isemptypreviouscellslabel.FunctionHandle=@isEmptyFunction;
isemptypreviouscellslabel.FunctionArgs.TestVariable.Value=[];
isemptypreviouscellslabel.FunctionArgs.TestVariable.FunctionInstance='SaveCellsLabel';
isemptypreviouscellslabel.FunctionArgs.TestVariable.OutputArg='CellsLabel';
```

```
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,isemptypreviouscellslabel)
;

starttracks.InstanceName='StartTracks';
starttracks.FunctionHandle=@startTracks;
starttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
starttracks.FunctionArgs.CellsLabel.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.CellsLabel.InputArg='LoadLabelMatrix_LabelMatrix';
starttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
starttracks.FunctionArgs.ShapeParameters.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.ShapeParameters.InputArg='GetShapeParameters_ShapeParameters';
if_is_empty_cells_label_functions=addToFunctionChain(if_is_empty_cells_label_functions,starttrack
s);

getcurrenttracks.InstanceName='GetCurrentTracks';
getcurrenttracks.FunctionHandle=@getCurrentTracks;
getcurrenttracks.FunctionArgs.OffsetFrame.Value=-1;
getcurrenttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getcurrenttracks.FunctionArgs.TimeCol.Value=2;
getcurrenttracks.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getcurrenttracks.FunctionArgs.FrameStep.Value=FrameStep;
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg='Tracks';
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getcurrenttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getcurrenttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getcur
renttracks);

getprevioustracks.InstanceName='GetPreviousTracks';
getprevioustracks.FunctionHandle=@getCurrentTracks;
getprevioustracks.FunctionArgs.OffsetFrame.Value=-2;
getprevioustracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getprevioustracks.FunctionArgs.TimeCol.Value=2;
getprevioustracks.FunctionArgs.TrackIDCol.Value=1;
getprevioustracks.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getprevioustracks.FunctionArgs.FrameStep.Value=FrameStep;
getprevioustracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg='Tracks';
getprevioustracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getprevioustracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getprevioustracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getpre
vioustracks);

makeunassignedcellslist.InstanceName='MakeUnassignedCellsList';
makeunassignedcellslist.FunctionHandle=@makeUnassignedCellsList;
makeunassignedcellslist.FunctionArgs.CellsCentroids.FunctionInstance='IfIsEmptyPreviousCellsLabel
';
makeunassignedcellslist.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,makeun
assignedcellslist);

makeexcludedtrackslist.InstanceName='MakeExcludedTracksList';
makeexcludedtrackslist.FunctionHandle=@makeExcludedTracksList;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.FunctionInstance='MakeUnassignedCellsList'
;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.OutputArg='UnassignedCellsIDs';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,makeex
cludedtrackslist);

getcellsmeandisplacement.InstanceName='GetCellsMeanDisplacement';
getcellsmeandisplacement.FunctionHandle=@getObjectsMeanDisplacement;
getcellsmeandisplacement.FunctionArgs.Centroid1Col.Value=3;
getcellsmeandisplacement.FunctionArgs.Centroid2Col.Value=4;
getcellsmeandisplacement.FunctionArgs.CurrentTracks.FunctionInstance='GetCurrentTracks';
getcellsmeandisplacement.FunctionArgs.CurrentTracks.OutputArg='Tracks';
getcellsmeandisplacement.FunctionArgs.ObjectCentroids.FunctionInstance='IfIsEmptyPreviousCellsLab
el';
getcellsmeandisplacement.FunctionArgs.ObjectCentroids.InputArg='GetShapeParameters_Centroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getcel
lsmeandisplacement);

getparamscoefficientofvariation.InstanceName='GetParamsCoefficientOfVariation';
getparamscoefficientofvariation.FunctionHandle=@getParamsCoefficientOfVariation;
getparamscoefficientofvariation.FunctionArgs.AreaCol.Value=5;
getparamscoefficientofvariation.FunctionArgs.SolidityCol.Value=11;
```

```
getparamscoefficientofvariation.FunctionArgs.Params.FunctionInstance='IfIsEmptyPreviousCellsLabel
';
getparamscoefficientofvariation.FunctionArgs.Params.InputArg='GetShapeParameters_ShapeParameters'
;
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getpar
amscoefficientofvariation);

getmaxtrackid.InstanceName='GetMaxTrackID';
getmaxtrackid.FunctionHandle=@getMaxTrackID;
getmaxtrackid.FunctionArgs.TrackIDCol.Value=1;
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg='Tracks';
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg2='Tracks';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getmax
trackid);

isnotemptyunassignedcells.InstanceName='IsNotEmptyUnassignedCells';
isnotemptyunassignedcells.FunctionHandle=@isNotEmptyFunction;
isnotemptyunassignedcells.FunctionArgs.TestVariable.FunctionInstance='AssignCellToTrackUsingAll';
isnotemptyunassignedcells.FunctionArgs.TestVariable.OutputArg='UnassignedIDs';
isnotemptyunassignedcells.FunctionArgs.TestVariable.FunctionInstance2='AssignCellsToTracksLoop';
isnotemptyunassignedcells.FunctionArgs.TestVariable.InputArg2='MakeUnassignedCellsList_Unassigned
CellsIDs';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,isnotemptyun
assignedcells);

getcurrentunassignedcell.InstanceName='GetCurrentUnassignedCell';
getcurrentunassignedcell.FunctionHandle=@getCurrentUnassignedCell;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance='AssignCellToTrackUsingAll
';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTracksLoop'
;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassign
edCellsIDs';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,getcurrentun
assignedcell);

assigncelltotrackusingall.InstanceName='AssignCellToTrackUsingAll';
assigncelltotrackusingall.FunctionHandle=@assignCellToTrackUsingAll;
assigncelltotrackusingall.FunctionArgs.CheckCellPath.Value=true;
assigncelltotrackusingall.FunctionArgs.FrontParams.Value=FrontParams;
assigncelltotrackusingall.FunctionArgs.MaxSearchRadius.Value=MaxSearchRadius;
assigncelltotrackusingall.FunctionArgs.MinSearchRadius.Value=MinSearchRadius;
assigncelltotrackusingall.FunctionArgs.SearchRadiusPct.Value=SearchRadiusPct;
assigncelltotrackusingall.FunctionArgs.RelevantParametersIndex.Value=RelevantParametersIndex;
assigncelltotrackusingall.FunctionArgs.NrParamsForSureMatch.Value=NrParamsForSureMatch;
assigncelltotrackusingall.FunctionArgs.DefaultParamWeights.Value=DefaultParamWeights;
assigncelltotrackusingall.FunctionArgs.UnknownParamWeights.Value=UnknownParamWeights;
assigncelltotrackusingall.FunctionArgs.DistanceRankingOrder.Value=DistanceRankingOrder;
assigncelltotrackusingall.FunctionArgs.DirectionRankingOrder.Value=DirectionRankingOrder;
assigncelltotrackusingall.FunctionArgs.UnknownRankingOrder.Value=UnknownRankingOrder;
assigncelltotrackusingall.FunctionArgs.MinSecondDistance.Value=MinSecondDistance;
assigncelltotrackusingall.FunctionArgs.MaxDistRatio.Value=MaxDistRatio;
assigncelltotrackusingall.FunctionArgs.MaxAngleDiff.Value=MaxAngleDiff;
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.Value=[];
assigncelltotrackusingall.FunctionArgs.TrackAssignments.Value=[];
assigncelltotrackusingall.FunctionArgs.MatchingGroups.Value=[];
assigncelltotrackusingall.FunctionArgs.MatchingGroupsStats.Value=[];
assigncelltotrackusingall.FunctionArgs.UnassignedCells.FunctionInstance='AssignCellToTrackUsingAl
l';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.FunctionInstance='AssignCellToTrackUsingAll
';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.OutputArg='ExcludedTracks';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.FunctionInstance='SetMatchingGroupIndex';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.OutputArg='ShapeParameters';
assigncelltotrackusingall.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellToTrackUsingA
ll';
assigncelltotrackusingall.FunctionArgs.TrackAssignments.OutputArg='TrackAssignments';
assigncelltotrackusingall.FunctionArgs.MatchingGroups.FunctionInstance='AssignCellToTrackUsingAll
';
assigncelltotrackusingall.FunctionArgs.MatchingGroups.OutputArg='MatchingGroups';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTracksLoop
';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassig
nedCellsIDs';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.FunctionInstance2='AssignCellsToTracksLoop'
;
```

```
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.InputArg2='MakeExcludedTracksList_ExcludedT
racks';
assigncelltotrackusingall.FunctionArgs.CellsLabel.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CellsLabel.InputArg='LoadLabelMatrix_LabelMatrix';
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.FunctionInstance='AssignCellsToTracksLo
op';
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.InputArg='SaveCellsLabel_CellsLabel';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.FunctionInstance2='AssignCellsToTracksLoop
';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.InputArg2='GetShapeParameters_ShapeParamet
ers';
assigncelltotrackusingall.FunctionArgs.CellsCentroids.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
assigncelltotrackusingall.FunctionArgs.CurrentTracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CurrentTracks.InputArg='GetCurrentTracks_Tracks';
assigncelltotrackusingall.FunctionArgs.MaxTrackID.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.MaxTrackID.InputArg='GetMaxTrackID_MaxTrackID';
assigncelltotrackusingall.FunctionArgs.Tracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.Tracks.InputArg='IfIsEmptyPreviousCellsLabel_Tracks';
assigncelltotrackusingall.FunctionArgs.MatchingGroupsStats.FunctionInstance='AssignCellsToTracksL
oop';
assigncelltotrackusingall.FunctionArgs.MatchingGroupsStats.InputArg='GetMatchingGroupMeans_Matchi
ngGroupStats';
assigncelltotrackusingall.FunctionArgs.ParamsCoeffOfVariation.FunctionInstance='AssignCellsToTrac
ksLoop';
assigncelltotrackusingall.FunctionArgs.ParamsCoeffOfVariation.InputArg='GetParamsCoefficientOfVar
iation_CoefficientOfVariation';
assigncelltotrackusingall.FunctionArgs.PreviousTracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.PreviousTracks.InputArg='GetPreviousTracks_Tracks';
assigncelltotrackusingall.FunctionArgs.TracksLayout.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
assigncelltotrackusingall.FunctionArgs.MatchingGroups.FunctionInstance2='AssignCellsToTracksLoop'
;
assigncelltotrackusingall.FunctionArgs.MatchingGroups.InputArg2='HoldMatchingGroups_ValueToHold';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,assigncellto
trackusingall);

setmatchinggroupindex.InstanceName='SetMatchingGroupIndex';
setmatchinggroupindex.FunctionHandle=@setGroupIndex;
setmatchinggroupindex.FunctionArgs.AreaCol.Value=5;
setmatchinggroupindex.FunctionArgs.GroupIDCol.Value=13;
setmatchinggroupindex.FunctionArgs.CellID.FunctionInstance='GetCurrentUnassignedCell';
setmatchinggroupindex.FunctionArgs.CellID.OutputArg='CellID';
setmatchinggroupindex.FunctionArgs.GroupIndex.FunctionInstance='AssignCellToTrackUsingAll';
setmatchinggroupindex.FunctionArgs.GroupIndex.OutputArg='GroupIndex';
setmatchinggroupindex.FunctionArgs.ShapeParameters.FunctionInstance='SetMatchingGroupIndex';
setmatchinggroupindex.FunctionArgs.ShapeParameters.OutputArg='ShapeParameters';
setmatchinggroupindex.FunctionArgs.ShapeParameters.FunctionInstance2='AssignCellsToTracksLoop';
setmatchinggroupindex.FunctionArgs.ShapeParameters.InputArg2='GetShapeParameters_ShapeParameters'
;
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,setmatchingg
roupindex);

assigncellstotracksloop.InstanceName='AssignCellsToTracksLoop';
assigncellstotracksloop.FunctionHandle=@whileLoop;
assigncellstotracksloop.FunctionArgs.TestFunction.FunctionInstance='IsNotEmptyUnassignedCells';
assigncellstotracksloop.FunctionArgs.TestFunction.OutputArg='Boolean';
assigncellstotracksloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.FunctionInstance=
'MakeUnassignedCellsList';
assigncellstotracksloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.OutputArg='Unassi
gnedCellsIDs';
assigncellstotracksloop.FunctionArgs.MakeExcludedTracksList_ExcludedTracks.FunctionInstance='Make
ExcludedTracksList';
assigncellstotracksloop.FunctionArgs.MakeExcludedTracksList_ExcludedTracks.OutputArg='ExcludedTra
cks';
assigncellstotracksloop.FunctionArgs.GetCurrentTracks_Tracks.FunctionInstance='GetCurrentTracks';
assigncellstotracksloop.FunctionArgs.GetCurrentTracks_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.GetMaxTrackID_MaxTrackID.FunctionInstance='GetMaxTrackID';
assigncellstotracksloop.FunctionArgs.GetMaxTrackID_MaxTrackID.OutputArg='MaxTrackID';
assigncellstotracksloop.FunctionArgs.IfIsEmptyPreviousCellsLabel_Tracks.FunctionInstance='IfIsEmp
tyPreviousCellsLabel';
assigncellstotracksloop.FunctionArgs.IfIsEmptyPreviousCellsLabel_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.GetMatchingGroupMeans_MatchingGroupStats.FunctionInstance='G
etMatchingGroupMeans';
assigncellstotracksloop.FunctionArgs.GetMatchingGroupMeans_MatchingGroupStats.OutputArg='Matching
GroupStats';
assigncellstotracksloop.FunctionArgs.GetParamsCoefficientOfVariation_CoefficientOfVariation.Funct
ionInstance='GetParamsCoefficientOfVariation';
assigncellstotracksloop.FunctionArgs.GetParamsCoefficientOfVariation_CoefficientOfVariation.Outpu
tArg='CoefficientOfVariation';
```

```
assigncellstotracksloop.FunctionArgs.GetPreviousTracks_Tracks.FunctionInstance='GetPreviousTracks
';
assigncellstotracksloop.FunctionArgs.GetPreviousTracks_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.LoadLabelMatrix_LabelMatrix.FunctionInstance='IfIsEmptyPrevi
ousCellsLabel';
assigncellstotracksloop.FunctionArgs.LoadLabelMatrix_LabelMatrix.InputArg='LoadLabelMatrix_LabelM
atrix';
assigncellstotracksloop.FunctionArgs.SaveCellsLabel_CellsLabel.FunctionInstance='IfIsEmptyPreviou
sCellsLabel';
assigncellstotracksloop.FunctionArgs.SaveCellsLabel_CellsLabel.InputArg='SaveCellsLabel_CellsLabe
l';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_ShapeParameters.FunctionInstance='IfIsEmp
tyPreviousCellsLabel';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_ShapeParameters.InputArg='GetShapeParamet
ers_ShapeParameters';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_Centroids.FunctionInstance='IfIsEmptyPrev
iousCellsLabel';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_Centroids.InputArg='GetShapeParameters_Ce
ntroids';
assigncellstotracksloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='IfIsEmptyPre
viousCellsLabel';
assigncellstotracksloop.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout_Tra
cksLayout';
assigncellstotracksloop.FunctionArgs.HoldMatchingGroups_ValueToHold.FunctionInstance='IfIsEmptyPr
eviousCellsLabel';
assigncellstotracksloop.FunctionArgs.HoldMatchingGroups_ValueToHold.InputArg='HoldMatchingGroups_
ValueToHold';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_TrackAssignments.FunctionInstance='A
ssignCellToTrackUsingAll';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_TrackAssignments.OutputArg='TrackAss
ignments';
assigncellstotracksloop.KeepValues.SetMatchingGroupIndex_ShapeParameters.FunctionInstance='SetMat
chingGroupIndex';
assigncellstotracksloop.KeepValues.SetMatchingGroupIndex_ShapeParameters.OutputArg='ShapeParamete
rs';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.FunctionInstance='Ass
ignCellToTrackUsingAll';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.OutputArg='MatchingGr
oups';
assigncellstotracksloop.LoopFunctions=assign_cells_to_tracks_functions;
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,assign
cellstotracksloop);

continuetracks.InstanceName='ContinueTracks';
continuetracks.FunctionHandle=@continueTracks;
continuetracks.FunctionArgs.TimeFrame.Value=TimeFrame;
continuetracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
continuetracks.FunctionArgs.Tracks.OutputArg='Tracks';
continuetracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
continuetracks.FunctionArgs.Tracks.OutputArg2='Tracks';
continuetracks.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellsToTracksLoop';
continuetracks.FunctionArgs.TrackAssignments.OutputArg='AssignCellToTrackUsingAll_TrackAssignment
s';
continuetracks.FunctionArgs.ShapeParameters.FunctionInstance='AssignCellsToTracksLoop';
continuetracks.FunctionArgs.ShapeParameters.OutputArg='SetMatchingGroupIndex_ShapeParameters';
continuetracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
continuetracks.FunctionArgs.CellsCentroids.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,contin
uetracks);

getmatchinggroupmeans.InstanceName='GetMatchingGroupMeans';
getmatchinggroupmeans.FunctionHandle=@getMatchingGroupMeans;
getmatchinggroupmeans.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getmatchinggroupmeans.FunctionArgs.Tracks.OutputArg='Tracks';
getmatchinggroupmeans.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getmatchinggroupmeans.FunctionArgs.Tracks.OutputArg2='Tracks';
getmatchinggroupmeans.FunctionArgs.TracksLayout.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getmatchinggroupmeans.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getmat
chinggroupmeans);

ifisemptypreviouscellslabel.InstanceName='IfIsEmptyPreviousCellsLabel';
ifisemptypreviouscellslabel.FunctionHandle=@if_statement;
ifisemptypreviouscellslabel.FunctionArgs.PreviousCellsLabel.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.MatchingGroupsStats.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.TrackAssignments.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.TestVariable.FunctionInstance='IsEmptyPreviousCellsLabel
';
ifisemptypreviouscellslabel.FunctionArgs.TestVariable.OutputArg='Boolean';
```

```
ifisemptypreviouscellslabel.FunctionArgs.LoadLabelMatrix_LabelMatrix.FunctionInstance='LoadLabelM
atrix';
ifisemptypreviouscellslabel.FunctionArgs.LoadLabelMatrix_LabelMatrix.OutputArg='LabelMatrix';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_LoopCounter.FunctionInstance='Segmentat
ionLoop';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_LoopCounter.OutputArg='LoopCounter';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_ShapeParameters.FunctionInstance='Get
ShapeParameters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_ShapeParameters.OutputArg='ShapeParam
eters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_Centroids.FunctionInstance='GetShapeP
arameters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_Centroids.OutputArg='Centroids';
ifisemptypreviouscellslabel.FunctionArgs.SaveCellsLabel_CellsLabel.FunctionInstance='SaveCellsLab
el';
ifisemptypreviouscellslabel.FunctionArgs.SaveCellsLabel_CellsLabel.OutputArg='CellsLabel';
ifisemptypreviouscellslabel.FunctionArgs.HoldMatchingGroups_ValueToHold.FunctionInstance='HoldMat
chingGroups';
ifisemptypreviouscellslabel.FunctionArgs.HoldMatchingGroups_ValueToHold.OutputArg='ValueToHold';
ifisemptypreviouscellslabel.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='Segmenta
tionLoop';
ifisemptypreviouscellslabel.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout
_TracksLayout';
ifisemptypreviouscellslabel.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.FunctionInstance=
'AssignCellsToTracksLoop';
ifisemptypreviouscellslabel.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.OutputArg='Assign
CellToTrackUsingAll_MatchingGroups';
ifisemptypreviouscellslabel.KeepValues.ContinueTracks_Tracks.FunctionInstance='ContinueTracks';
ifisemptypreviouscellslabel.KeepValues.ContinueTracks_Tracks.OutputArg='Tracks';
ifisemptypreviouscellslabel.ElseFunctions=else_is_empty_cells_label_functions;
ifisemptypreviouscellslabel.IfFunctions=if_is_empty_cells_label_functions;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,ifisemptypreviouscellslabe
l);

savecellslabel.InstanceName='SaveCellsLabel';
savecellslabel.FunctionHandle=@saveCellsLabel;
savecellslabel.FunctionArgs.FileRoot.Value=SegmentationFilesRoot;
savecellslabel.FunctionArgs.NumberFormat.Value=NumberFormat;
savecellslabel.FunctionArgs.CellsLabel.FunctionInstance='LoadLabelMatrix';
savecellslabel.FunctionArgs.CellsLabel.OutputArg='LabelMatrix';
savecellslabel.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
savecellslabel.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,savecellslabel);

holdmatchinggroups.InstanceName='HoldMatchingGroups';
holdmatchinggroups.FunctionHandle=@holdValue;
holdmatchinggroups.FunctionArgs.ValueToHold.Value=[];
holdmatchinggroups.FunctionArgs.ValueToHold.FunctionInstance='IfIsEmptyPreviousCellsLabel';
holdmatchinggroups.FunctionArgs.ValueToHold.OutputArg='AssignCellToTrackUsingAll_MatchingGroups';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,holdmatchinggroups);

segmentationloop.InstanceName='SegmentationLoop';
segmentationloop.FunctionHandle=@forLoop;
segmentationloop.FunctionArgs.StartLoop.Value=StartFrame;
segmentationloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
segmentationloop.FunctionArgs.IncrementLoop.Value=FrameStep;
segmentationloop.FunctionArgs.MatchingGroups.Value=[];
segmentationloop.FunctionArgs.Tracks.Value=[];
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='LoadTracksLayout';
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.OutputArg='TracksLayout';
segmentationloop.KeepValues.ContinueTracks_Tracks.FunctionInstance='IfIsEmptyPreviousCellsLabel';
segmentationloop.KeepValues.ContinueTracks_Tracks.OutputArg='ContinueTracks_Tracks';
segmentationloop.LoopFunctions=image_read_loop_functions;
functions_list=addToFunctionChain(functions_list,segmentationloop);

savetracks.InstanceName='SaveTracks';
savetracks.FunctionHandle=@saveTracks;
savetracks.FunctionArgs.TracksFileName.Value=[TracksFolder '/tracks.mat'];
savetracks.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
savetracks.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,savetracks);

gettrackids.InstanceName='GetTrackIDs';
gettrackids.FunctionHandle=@getTrackIDs;
gettrackids.FunctionArgs.TrackIDCol.Value=1;
gettrackids.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
gettrackids.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,gettrackids);

detectmergecandidates.InstanceName='DetectMergeCandidates';
detectmergecandidates.FunctionHandle=@detectMergeCandidatesUsingDistance;
```

```
detectmergecandidates.FunctionArgs.MaxMergeDistance.Value=MaxMergeDistance;
detectmergecandidates.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDs';
detectmergecandidates.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
detectmergecandidates.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
detectmergecandidates.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
detectmergecandidates.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
detectmergecandidates.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,detectmergecandidates);

mergetracks.InstanceName='MergeTracks';
mergetracks.FunctionHandle=@mergeTracks;
mergetracks.FunctionArgs.FrameCount.Value=FrameCount;
mergetracks.FunctionArgs.StartFrame.Value=StartFrame;
mergetracks.FunctionArgs.TimeFrame.Value=TimeFrame;
mergetracks.FunctionArgs.SegFileRoot.Value=SegmentationFilesRoot;
mergetracks.FunctionArgs.FrameStep.Value=FrameStep;
mergetracks.FunctionArgs.NumberFormat.Value=NumberFormat;
mergetracks.FunctionArgs.TracksToBeMerged.FunctionInstance='DetectMergeCandidates';
mergetracks.FunctionArgs.TracksToBeMerged.OutputArg='TracksToBeMerged';
mergetracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
mergetracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
mergetracks.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
mergetracks.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,mergetracks);

gettrackidsaftermerge.InstanceName='GetTrackIDsAfterMerge';
gettrackidsaftermerge.FunctionHandle=@getTrackIDs;
gettrackidsaftermerge.FunctionArgs.TrackIDCol.Value=1;
gettrackidsaftermerge.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
gettrackidsaftermerge.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,gettrackidsaftermerge);

makeancestryforfirstframecells.InstanceName='MakeAncestryForFirstFrameCells';
makeancestryforfirstframecells.FunctionHandle=@makeAncestryForFirstFrameCells;
makeancestryforfirstframecells.FunctionArgs.TimeCol.Value=2;
makeancestryforfirstframecells.FunctionArgs.TrackIDCol.Value=1;
makeancestryforfirstframecells.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
makeancestryforfirstframecells.FunctionArgs.Tracks.OutputArg='Tracks';
makeancestryforfirstframecells.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDsAfterMerge';
makeancestryforfirstframecells.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
functions_list=addToFunctionChain(functions_list,makeancestryforfirstframecells);

detectmitoticevents.InstanceName='DetectMitoticEvents';
detectmitoticevents.FunctionHandle=@detectMitoticEvents;
detectmitoticevents.FunctionArgs.MaxSplitArea.Value=MaxSplitArea;
detectmitoticevents.FunctionArgs.MinSplitEccentricity.Value=MinSplitEccentricity;
detectmitoticevents.FunctionArgs.MaxSplitEccentricity.Value=MaxSplitEccentricity;
detectmitoticevents.FunctionArgs.MaxSplitDistance.Value=MaxSplitDistance;
detectmitoticevents.FunctionArgs.MinTimeForSplit.Value=MinTimeForSplit;
detectmitoticevents.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
detectmitoticevents.FunctionArgs.Tracks.OutputArg='Tracks';
detectmitoticevents.FunctionArgs.UntestedIDs.FunctionInstance='MakeAncestryForFirstFrameCells';
detectmitoticevents.FunctionArgs.UntestedIDs.OutputArg='UntestedIDs';
detectmitoticevents.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
detectmitoticevents.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
functions_list=addToFunctionChain(functions_list,detectmitoticevents);

makeancestryforcellsenteringframes.InstanceName='MakeAncestryForCellsEnteringFrames';
makeancestryforcellsenteringframes.FunctionHandle=@makeAncestryForCellsEnteringFrames;
makeancestryforcellsenteringframes.FunctionArgs.TimeCol.Value=2;
makeancestryforcellsenteringframes.FunctionArgs.TrackIDCol.Value=1;
makeancestryforcellsenteringframes.FunctionArgs.SplitCells.FunctionInstance='DetectMitoticEvents'
;
makeancestryforcellsenteringframes.FunctionArgs.SplitCells.OutputArg='SplitCells';
makeancestryforcellsenteringframes.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDsAfterMerge'
;
makeancestryforcellsenteringframes.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
makeancestryforcellsenteringframes.FunctionArgs.FirstFrameIDs.FunctionInstance='MakeAncestryForFi
rstFrameCells';
makeancestryforcellsenteringframes.FunctionArgs.FirstFrameIDs.OutputArg='FirstFrameIDs';
makeancestryforcellsenteringframes.FunctionArgs.CellsAncestry.FunctionInstance='MakeAncestryForFi
rstFrameCells';
makeancestryforcellsenteringframes.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
makeancestryforcellsenteringframes.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
makeancestryforcellsenteringframes.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,makeancestryforcellsenteringframes);

splittracks.InstanceName='SplitTracks';
splittracks.FunctionHandle=@splitTracks;
splittracks.FunctionArgs.TimeFrame.Value=TimeFrame;
splittracks.FunctionArgs.SplitCells.FunctionInstance='DetectMitoticEvents';
```

```
splittracks.FunctionArgs.SplitCells.OutputArg='SplitCells';
splittracks.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
splittracks.FunctionArgs.Tracks.OutputArg='Tracks';
splittracks.FunctionArgs.CellsAncestry.FunctionInstance='MakeAncestryForCellsEnteringFrames';
splittracks.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
splittracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
splittracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
splittracks.FunctionArgs.AncestryLayout.FunctionInstance='LoadAncestryLayout';
splittracks.FunctionArgs.AncestryLayout.OutputArg='AncestryLayout';
functions_list=addToFunctionChain(functions_list,splittracks);

removeshorttracks.InstanceName='RemoveShortTracks';
removeshorttracks.FunctionHandle=@removeShortTracks;
removeshorttracks.FunctionArgs.MinLifespan.Value=MinLifespan;
removeshorttracks.FunctionArgs.Tracks.FunctionInstance='SplitTracks';
removeshorttracks.FunctionArgs.Tracks.OutputArg='Tracks';
removeshorttracks.FunctionArgs.CellsAncestry.FunctionInstance='SplitTracks';
removeshorttracks.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
removeshorttracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
removeshorttracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
removeshorttracks.FunctionArgs.AncestryLayout.FunctionInstance='LoadAncestryLayout';
removeshorttracks.FunctionArgs.AncestryLayout.OutputArg='AncestryLayout';
functions_list=addToFunctionChain(functions_list,removeshorttracks);

saveupdatedtracks.InstanceName='SaveUpdatedTracks';
saveupdatedtracks.FunctionHandle=@saveTracks;
saveupdatedtracks.FunctionArgs.TracksFileName.Value=[AncestryFolder '/tracks.mat'];
saveupdatedtracks.FunctionArgs.Tracks.FunctionInstance='RemoveShortTracks';
saveupdatedtracks.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,saveupdatedtracks);

saveancestry.InstanceName='SaveAncestry';
saveancestry.FunctionHandle=@saveAncestry;
saveancestry.FunctionArgs.AncestryFileName.Value=[AncestryFolder '/ancestry.mat'];
saveancestry.FunctionArgs.CellsAncestry.FunctionInstance='RemoveShortTracks';
saveancestry.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
functions_list=addToFunctionChain(functions_list,saveancestry);

makeimagenamesinoverlayloop.InstanceName='MakeImageNamesInOverlayLoop';
makeimagenamesinoverlayloop.FunctionHandle=@makeImgFileName;
makeimagenamesinoverlayloop.FunctionArgs.FileBase.Value=ImageFileBase;
makeimagenamesinoverlayloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinoverlayloop.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenamesinoverlayloop.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
makeimagenamesinoverlayloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,makeimagenamesinover
layloop);

readimagesinoverlayloop.InstanceName='ReadImagesInOverlayLoop';
readimagesinoverlayloop.FunctionHandle=@readImage;
readimagesinoverlayloop.FunctionArgs.ImageChannel.Value='';
readimagesinoverlayloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInOverlayLoop';
readimagesinoverlayloop.FunctionArgs.ImageName.OutputArg='FileName';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,readimagesinoverlayl
oop);

getcurrenttracks2.InstanceName='GetCurrentTracks2';
getcurrenttracks2.FunctionHandle=@getCurrentTracks;
getcurrenttracks2.FunctionArgs.OffsetFrame.Value=0;
getcurrenttracks2.FunctionArgs.TimeFrame.Value=TimeFrame;
getcurrenttracks2.FunctionArgs.TimeCol.Value=2;
getcurrenttracks2.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks2.FunctionArgs.MaxMissingFrames.Value=0;
getcurrenttracks2.FunctionArgs.FrameStep.Value=FrameStep;
getcurrenttracks2.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
getcurrenttracks2.FunctionArgs.CurFrame.OutputArg='LoopCounter';
getcurrenttracks2.FunctionArgs.Tracks.FunctionInstance='ImageOverlayLoop';
getcurrenttracks2.FunctionArgs.Tracks.InputArg='RemoveShortTracks_Tracks';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,getcurrenttracks2);

makematnamesinoverlayloop.InstanceName='MakeMatNamesInOverlayLoop';
makematnamesinoverlayloop.FunctionHandle=@makeImgFileName;
makematnamesinoverlayloop.FunctionArgs.FileBase.Value=SegmentationFilesRoot;
makematnamesinoverlayloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makematnamesinoverlayloop.FunctionArgs.FileExt.Value='.mat';
makematnamesinoverlayloop.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
makematnamesinoverlayloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,makematnamesinoverla
yloop);

loadcellslabel.InstanceName='LoadCellsLabel';
```

```
loadcellslabel.FunctionHandle=@loadCellsLabel;
loadcellslabel.FunctionArgs.FileName.FunctionInstance='MakeMatNamesInOverlayLoop';
loadcellslabel.FunctionArgs.FileName.OutputArg='FileName';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,loadcellslabel);

loadcolormap.InstanceName='LoadColormap';
loadcolormap.FunctionHandle=@loadColormap;
loadcolormap.FunctionArgs.FileName.Value='colormap_lines';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,loadcolormap);

displayoverlayingframe.InstanceName='DisplayOverlayingFrame';
displayoverlayingframe.FunctionHandle=@displayVariable;
displayoverlayingframe.FunctionArgs.VariableName.Value='Overlaying Frame';
displayoverlayingframe.FunctionArgs.Variable.FunctionInstance='ImageOverlayLoop';
displayoverlayingframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,displayoverlayingfra
me);

displayancestry.InstanceName='DisplayAncestry';
displayancestry.FunctionHandle=@displayAncestryData;
displayancestry.FunctionArgs.NumberFormat.Value=NumberFormat;
displayancestry.FunctionArgs.ProlDir.Value=AncestryFolder;
displayancestry.FunctionArgs.ImageFileName.Value=ImageFilesRoot;
displayancestry.FunctionArgs.DS.Value='/';
displayancestry.FunctionArgs.Image.FunctionInstance='ReadImagesInOverlayLoop';
displayancestry.FunctionArgs.Image.OutputArg='Image';
displayancestry.FunctionArgs.CurrentTracks.FunctionInstance='GetCurrentTracks2';
displayancestry.FunctionArgs.CurrentTracks.OutputArg='Tracks';
displayancestry.FunctionArgs.CellsLabel.FunctionInstance='LoadCellsLabel';
displayancestry.FunctionArgs.CellsLabel.OutputArg='LabelMatrix';
displayancestry.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.CurFrame.OutputArg='LoopCounter';
displayancestry.FunctionArgs.ColorMap.FunctionInstance='LoadColormap';
displayancestry.FunctionArgs.ColorMap.OutputArg='Colormap';
displayancestry.FunctionArgs.CellsAncestry.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.CellsAncestry.InputArg='RemoveShortTracks_CellsAncestry';
displayancestry.FunctionArgs.TracksLayout.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
displayancestry.FunctionArgs.AncestryLayout.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.AncestryLayout.InputArg='LoadAncestryLayout_AncestryLayout';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,displayancestry);

imageoverlayloop.InstanceName='ImageOverlayLoop';
imageoverlayloop.FunctionHandle=@forLoop;
imageoverlayloop.FunctionArgs.StartLoop.Value=StartFrame;
imageoverlayloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
imageoverlayloop.FunctionArgs.IncrementLoop.Value=FrameStep;
imageoverlayloop.FunctionArgs.RemoveShortTracks_Tracks.FunctionInstance='RemoveShortTracks';
imageoverlayloop.FunctionArgs.RemoveShortTracks_Tracks.OutputArg='Tracks';
imageoverlayloop.FunctionArgs.RemoveShortTracks_CellsAncestry.FunctionInstance='RemoveShortTracks
';
imageoverlayloop.FunctionArgs.RemoveShortTracks_CellsAncestry.OutputArg='CellsAncestry';
imageoverlayloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='LoadTracksLayout';
imageoverlayloop.FunctionArgs.LoadTracksLayout_TracksLayout.OutputArg='TracksLayout';
imageoverlayloop.FunctionArgs.LoadAncestryLayout_AncestryLayout.FunctionInstance='LoadAncestryLay
out';
imageoverlayloop.FunctionArgs.LoadAncestryLayout_AncestryLayout.OutputArg='AncestryLayout';
imageoverlayloop.LoopFunctions=image_overlay_loop_functions;
functions_list=addToFunctionChain(functions_list,imageoverlayloop);

saveancestryspreadsheets.InstanceName='SaveAncestrySpreadsheets';
saveancestryspreadsheets.FunctionHandle=@saveAncestrySpreadsheets;
saveancestryspreadsheets.FunctionArgs.ShapesXlsFile.Value=ShapesSpreadsheet;
saveancestryspreadsheets.FunctionArgs.ProlXlsFile.Value=AncestrySpreadsheet;
saveancestryspreadsheets.FunctionArgs.Tracks.FunctionInstance='RemoveShortTracks';
saveancestryspreadsheets.FunctionArgs.Tracks.OutputArg='Tracks';
saveancestryspreadsheets.FunctionArgs.CellsAncestry.FunctionInstance='RemoveShortTracks';
saveancestryspreadsheets.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
saveancestryspreadsheets.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
saveancestryspreadsheets.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
functions_list=addToFunctionChain(functions_list,saveancestryspreadsheets);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayFluoNucl.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayFluoNucl()
%assayFluoNuclTestCA - This module is used to track cells that have been stained using a
%      nuclear stain  using a custom CellAnimation algorithm.
%This  algorithm  is  more  accurate  and  flexible than
%the   nearest-neighbor algorithm  at the  expense of  execution speed.
% The assay   tracks the  cells, detects   mitotic events
%and  records tracking  data,  cell shape  parameters  and ancestry
% information to spreadsheets.    For each tracked image,   an
%overlayed image is   saved  to disk that  displays the
% detected cell  outlines, cell   IDs and cell  generation. ImageFolder
%  - String variable that specifies the    absolute location of
%the directory which    contains the   time-lapse images. An
%example  of such a  string variable would   be  'c:/sample
%images/high-density'. ImageFilesRoot  - String variable   specifying the root  image file
% name. The root   image file name   for
%a set of images  is the image  file name
%of any of the   images without  the number  or
% the file extension. For  example, if the  file name
%   is 'Experiment-0002_Position(8)_t021.tif' the root image file name will  be 'Experiment-
0002_Position(8)_t'.
% ImageExtension     - String variable specifying the image file extension
%including the preceding   dot.     For example  if
%the file name is  'image003.jpg' the image  extension is
% '.jpg'. StartFrame - Number  specifying the first image  in the sequence
% to be  analyzed.    The minimum  value  for
%this variable depends  on the numbering   of the  image
% sequence so if  the  first  image in the  sequence
%is 'image003.tif'   then the  minimum value   is 3. FrameCount
%- Number  specifying  how many images  from  the image
% sequence should be processed. TimeFrame   - Number specifying  the time
% between  consecutive images in  minutes.  FrameStep -   Number
%specifying the step size when   reading images. Set  this variable
% to  1  to read every  image in   the
% sequence, 2 to read every   other image and   so
%  on. NumberFormat  - String value specifying the  number of
% digits   in the image file names  in  the sequence.
% For example    if the  image file name is 'image020.jpg'
% the value for      the NumberFormat is '%03d', while
% if the file name   is 'image000020.jpg'    the value
%should  be '%06d'. MaxFramesMissing -   Number specifying for   how
%  many frames a cell may  be disappear before its
% track is    ended. OutputFolder  - The folder where the
%overlayed  images and track data   will    be saved.
%By  default  this value is set to   a folder named
%   'output' within  the folder where  the  images
%to be analyzed are  located. AncestryFolder    - The  folder
%where the  overlayed images and ancestry data   will be saved.
% By   default  this value is set to  a
% folder named 'ancestry' within  the output   folder. AncestrySpreadsheet -
%The  path name  to the spreadsheet containing the   ancestry data.
%By    default this  value  is set to a file
% named    'ancestry.csv' within the ancestry  folder. ShapesSpreadsheet  -
%The path name   to  the spreadsheet  containing the position and
%shape   properties for     each cell in the timelapse
% sequence at every time point.   By    default this
%is  set to to  a  file named 'shapes.csv'   within
% the ancestry  folder. TracksFolder - The  folder where  the label
% matrixes   containing the cell outlines are   saved. By
%default this   value is   set to  a folder named
%'track' within  the  output folder. SegmentationFilesRoot   -   The
%root file name of the label  matrixes containing  the cell
%outlines.   ImageFileBase - The path name  to the images. This
%value is   generated from   the ImageFolder   and the
% ImageFilesRoot and should not  be changed. BrightnessThresholdPct   -
%Number  specifying the percentage threshold value for  the image generated by the
%   generateBinImgUsingLocAvg    filter. Any pixel in  the original
%image smaller   than the  threshold value times   the
% corresponding value in   the local average image  below this value
%will  be    set  to  zero while the rest
% will be set to one.     ObjectArea -  Number
%specifying the threshold area  for the  clearSmallObjects, polygonalAssistedWatershed   and
areaFilterLabel
% filters. Objects   below this value   will be removed
% from the filtered image.  Strel - String  variable  specifying
%the  type  of filter used to generate the   local average
% image in     generateBinImgUsingLocAvg. Currently 'disk' is  the only
%value supported.  StrelSize -  Number specifying   the size
```

```
%of the local neighborhood used to calculate   the average
% for each pixel in  the local average  image generated by
% the  generateBinImgUsingLocAvg  module. ClearBorder - Boolean value specifying   whether
%objects  next to or   touching  the image border in
% the  binary images  generated by  the generateBinImgUsingLocAvg   module
%will  be  erased (true) or not  (false). ClearBorderDist   -
%Number specifying     how close to the border objects may be
% and still be      erased  if the ClearBorder
%parameter is set  to true  in  the generateBinImgUsingLocAvg module.
%MedianFilterSize  - Number specifying the   size of  the median filter
%used by  the distanceWatershed   module.    Setting  this
%to a higher integer value will  reduce the  number of
%  objects detected  by the module and can   be used
%to prevent   oversegmentation.  MinSolidity  - Number specifying  a threshold
%solidity  value for  the  solidityFilterLabelObjects filter. Objects  whose  solidity
%  is below this   value will  be removed from
%the filtered  image. MinAreaOverPerimeter -  Number  specifying  a  threshold
% AOP ratio value for the areaOverPerimeterFilterLabel  filter. Objects  with
%   an AOP ratio smaller than this value will be
%removed. ResizeImageScale  - Number   specifying  by what ratio the images
% will be  resized  before they  are   processed. By
% default  this  value is set to 0.5.   Used by
% the resizeImage module.   ApproximationDistance -  Number specifying how close the
% convex  hull  in   the getConvexObjects module approximates the
% object outline.  By default this     value is set
%to 2.5. Setting it to a     lower  value will
%  result in convex hulls that more closely   resemble the
%object  outlines   however  this increases the  chance of
%detecting  insignificant concavities. MaxSearchRadius - Number   specifying the   absolute
%lower bound   for the search radius to prevent selecting
%too  few  candidate   objects for a track. Used by
%assignCellToTrackUsingAll module. MinSearchRadius  - Number specifying     the absolute higher
% bound for the search radius to  prevent  selecting   too
%many    candidate objects for a track. Used by   assignCellToTrackUsingAll
%module.   MinSecondDistance - Number specifying  the  minimum significant distance
%between the   closest  candidate object to a  track  and
%the   second  closest. Used   to determine when distance should
%be used as     a ranking  parameter.   Used
%by assignCellToTrackUsingAll module. MaxDistRatio -   Number specifying the  maximum allowed
distance
% ratio   between the two   nearest candidate objects.  If
%the  ratio  is higher  than  this   value distance
%ranking will not be used.    Used  by   assignCellToTrackUsingAll
%module. MaxAngleDiff -  Number specifying the maximum allowed   angle difference
% between  a track and a candidate   object.  If
%the angle  is larger   than this value direction ranking  will
%not be     used for this object.   Used
%by assignCellToTrackUsingAll module. NrParamsForSureMatch -   Number specifying the  minimum
number of
%  closest  matches between a candidate   object parameters  and
%a  track's  object  parameters that  make  the  candidate
%object a  sure match to  the track.   Used  by
%  assignCellToTrackUsingAll  module. SearchRadiusPct - Number specifying the size of the
% neighborhood from    which  candidate objects for matching  the
%track are  selected.   It is  a multiple   of
%the distance to the  nearest   candidate  in  the current
%frame. Setting this   variable   equal to 1 turns this
% module  into a nearest-neighbor algorithm (only    the  nearest
%cell can be   a candidate). It does    not make
%sense  to have  a  value  lower  than 1.
%Used by assignCellToTrackUsingAll module. MaxMergeDistance -  Number specifying  the
%maximum  distance that one track may be from another track
% for the   duration  and still be considered for possible
%merging with  the   other track.   Used by  detectMergeCandidatesUsingDistance
% module. MaxSplitArea - Number  specifying  the maximum  area  a
%nucleus  may be and  still be   considered  as a
%part    of a possible mitotic event. Used by
%detectMitoticEvents module. MaxSplitDistance  - Number specifying   the maximum distance a
%new  nucleus may  be  from another nucleus   and
%still  be considered   as part of a possible mitotic  event.
%Used  by  detectMitoticEvents     module. MinSplitEccentricity - Number specifying
%the minimum eccentricity a     new nucleus may   have
%and still be  considered as part   of a  possible
% mitotic event.  Used by detectMitoticEvents module. MaxSplitEccentricity   - Number specifying
%the  maximum  eccentricity a  new  nucleus  may  have
%and still be  considered   as part of a  possible
%  mitotic event. Used by detectMitoticEvents module. MinTimeForSplit   -  Number
%specifying the  minimum time in   minutes a track needs
%to  exist before   it is considered for a  possible
% mitotic  event. Used   by detectMitoticEvents module. MinLifespan - Number specifying
%the  minimum  length  in   frames  a frame has
```

```
%to be  to  not be removed    by the removeShortTracks
%  module.  FrontParams - Numeric array specifying a set   of
%column  indices from   the shape  and motility  parameters matrix.
%  The parameters in  those  columns  will be heavily weighted,
% and  have  more   influence in determining the
%best match for a  track from    a list of objects.
% Used by   assignCellToTrackUsingAll  module.  DefaultParamWeights -  Numeric array
%specifying  a set of weights that    is  assigned to
% each shape  and motility parameter  based on   its
%prediction power.   Parameters with high prediction power are   assigned
%high  weights  and parameters  with  low prediction power  are
%assigned  lower weights. Used  by    assignCellToTrackUsingAll module. DistanceRankingOrder
% - Numeric  array specifying the default order of shape   and
% motility  parameters for   slow  moving objects when it cannot
%be    determined based on  prediction  power.  Used by
%assignCellToTrackUsingAll   module. DirectionRankingOrder  -  Numeric  array specifying the
default
%order    of shape and motility  parameters for   fast
% moving directional objects  when it cannot   be  determined
%based on  prediction power.  Used by  assignCellToTrackUsingAll module. RelevantParametersIndex
-
%  Boolean  array specifying   column indexes in the shape and
%motility matrix that   have    been   determined to
%be irrelevant for tracking. This  indicates   to the module  not
%to    use the parameters  those  columns  in computing
%track assignment probabilities.  The  order of column     indexes
% is provided in TracksLayout variable. Used by   assignCellToTrackUsingAll   module.
%UnknownParamWeights - Numeric   array specifying a set of weights to  be
%   assigned to shape  and  motility  parameters when
%the prediction  power of  the  parameters  cannot be determined. Used
% by    assignCellToTrackUsingAll module. UnknownRankingOrder -  Numeric array
%specifying  the order of the  shape and motility  parameters when
%their    predictive  power cannot  be determined. If the objects
%cannot be     categorized as either slow-moving    or
%fast directional the parameter  order provided in   this   variable
%is used.  Used by   assignCellToTrackUsingAll module. Important Modules  -
% areaFilterLabel, areaOverPerimeterFilterLabel, assignCellToTrackUsingAll,  clearSmallObjects,
detectMergeCandidatesUsingDistance,  detectMitoticEvents, distanceWatershed,
generateBinImgUsingLocAvg, getConvexObjects, polygonalAssistedWatershed,
% removeShortTracks,  segmentObjectsUsingMarkers,  solidityFilterLabel, splitTracks.

global functions_list;
functions_list=[];
%script variables
ImageFolder='C:/darren/movie2';
ImageFilesRoot='DsRed - Confocal - n';
ImageExtension='.tif';
StartFrame=1;
FrameCount=72;
TimeFrame=15;
FrameStep=1;
NumberFormat='%06d';
MaxFramesMissing=6;
OutputFolder=[ImageFolder '/output'];
AncestryFolder=[OutputFolder '/ancestry'];
AncestrySpreadsheet=[AncestryFolder 'ancestry.csv'];
ShapesSpreadsheet=[AncestryFolder 'shapes.csv'];
TracksFolder=[OutputFolder '/track'];
SegmentationFilesRoot=[TracksFolder '/grayscale'];
ImageFileBase=[ImageFolder '/' ImageFilesRoot];
BrightnessThresholdPct=1.1;
ObjectArea=30;
Strel='disk';
StrelSize=10;
ClearBorder=true;
ClearBorderDist=2;
MedianFilterSize=3;
MinSolidity=0.69;
MinAreaOverPerimeter=1.5;
ResizeImageScale=0.5;
ApproximationDistance=2.4;
MaxSearchRadius=Inf;
MinSearchRadius=0;
MinSecondDistance=5;
MaxDistRatio=0.6;
MaxAngleDiff=0.35;
NrParamsForSureMatch=5;
SearchRadiusPct=1.5;
MaxMergeDistance=23;
MaxSplitArea=400;
MaxSplitDistance=45;
```

```
MinSplitEccentricity=0.5;
MaxSplitEccentricity=0.95;
MinTimeForSplit=900;
MinLifespan=30;
FrontParams=[];
DefaultParamWeights=[34 21 13 8 5 3 2 2 2];
DistanceRankingOrder=[1 3 4 5 6 7 8 9 2];
DirectionRankingOrder=[2 3 4 5 6 7 8 9 1];
RelevantParametersIndex=[true true true false true false true true false];
UnknownParamWeights=[5 3 1 1 1 1 1 1 1];
UnknownRankingOrder=[1 2 3 4 5 6 7 8 9];
%end script variables

assign_cells_to_tracks_functions=[];
else_is_empty_cells_label_functions=[];
if_is_empty_cells_label_functions=[];
image_read_loop_functions=[];
image_overlay_loop_functions=[];

loadtrackslayout.InstanceName='LoadTracksLayout';
loadtrackslayout.FunctionHandle=@loadTracksLayout;
loadtrackslayout.FunctionArgs.FileName.Value='tracks_layout.mat';
functions_list=addToFunctionChain(functions_list,loadtrackslayout);

loadancestrylayout.InstanceName='LoadAncestryLayout';
loadancestrylayout.FunctionHandle=@loadAncestryLayout;
loadancestrylayout.FunctionArgs.FileName.Value='ancestry_layout.mat';
functions_list=addToFunctionChain(functions_list,loadancestrylayout);

displaycurframe.InstanceName='DisplayCurFrame';
displaycurframe.FunctionHandle=@displayVariable;
displaycurframe.FunctionArgs.VariableName.Value='Current Tracking Frame';
displaycurframe.FunctionArgs.Variable.FunctionInstance='SegmentationLoop';
displaycurframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,displaycurframe);

makeimagenamesinsegmentationloop.InstanceName='MakeImageNamesInSegmentationLoop';
makeimagenamesinsegmentationloop.FunctionHandle=@makeImgFileName;
makeimagenamesinsegmentationloop.FunctionArgs.FileBase.Value=ImageFileBase;
makeimagenamesinsegmentationloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinsegmentationloop.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makeimagenamesinsegmentati
onloop);

readimagesinsegmentationloop.InstanceName='ReadImagesInSegmentationLoop';
readimagesinsegmentationloop.FunctionHandle=@readImage;
readimagesinsegmentationloop.FunctionArgs.ImageChannel.Value='';
readimagesinsegmentationloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInSegmentatio
nLoop';
readimagesinsegmentationloop.FunctionArgs.ImageName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,readimagesinsegmentationlo
op);

normalizeimageto16bit.InstanceName='NormalizeImageTo16Bit';
normalizeimageto16bit.FunctionHandle=@imNorm;
normalizeimageto16bit.FunctionArgs.IntegerClass.Value='uint16';
normalizeimageto16bit.FunctionArgs.RawImage.FunctionInstance='ReadImagesInSegmentationLoop';
normalizeimageto16bit.FunctionArgs.RawImage.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,normalizeimageto16bit);

resizeimage.InstanceName='ResizeImage';
resizeimage.FunctionHandle=@resizeImage;
resizeimage.FunctionArgs.Scale.Value=ResizeImageScale;
resizeimage.FunctionArgs.Method.Value='bicubic';
resizeimage.FunctionArgs.Image.FunctionInstance='NormalizeImageTo16Bit';
resizeimage.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,resizeimage);

nuclbrightnesslocalaveragingfilter.InstanceName='NuclBrightnessLocalAveragingFilter';
nuclbrightnesslocalaveragingfilter.FunctionHandle=@generateBinImgUsingLocAvg;
nuclbrightnesslocalaveragingfilter.FunctionArgs.Strel.Value=Strel;
nuclbrightnesslocalaveragingfilter.FunctionArgs.StrelSize.Value=StrelSize;
nuclbrightnesslocalaveragingfilter.FunctionArgs.BrightnessThresholdPct.Value=BrightnessThresholdP
ct;
nuclbrightnesslocalaveragingfilter.FunctionArgs.ClearBorder.Value=ClearBorder;
nuclbrightnesslocalaveragingfilter.FunctionArgs.ClearBorderDist.Value=ClearBorderDist;
nuclbrightnesslocalaveragingfilter.FunctionArgs.Image.FunctionInstance='ResizeImage';
nuclbrightnesslocalaveragingfilter.FunctionArgs.Image.OutputArg='Image';
```

```
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,nuclbrightnesslocalaveragi
ngfilter);

fillholesnuclearimages.InstanceName='FillHolesNuclearImages';
fillholesnuclearimages.FunctionHandle=@fillHoles;
fillholesnuclearimages.FunctionArgs.Image.FunctionInstance='NuclBrightnessLocalAveragingFilter';
fillholesnuclearimages.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,fillholesnuclearimages);

clearsmallnuclei.InstanceName='ClearSmallNuclei';
clearsmallnuclei.FunctionHandle=@clearSmallObjects;
clearsmallnuclei.FunctionArgs.MinObjectArea.Value=ObjectArea;
clearsmallnuclei.FunctionArgs.Image.FunctionInstance='FillHolesNuclearImages';
clearsmallnuclei.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,clearsmallnuclei);

labelnuclei.InstanceName='LabelNuclei';
labelnuclei.FunctionHandle=@labelObjects;
labelnuclei.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
labelnuclei.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,labelnuclei);

getconvexobjects.InstanceName='GetConvexObjects';
getconvexobjects.FunctionHandle=@getConvexObjects;
getconvexobjects.FunctionArgs.ApproximationDistance.Value=ApproximationDistance;
getconvexobjects.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
getconvexobjects.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getconvexobjects);

distancewatershed.InstanceName='DistanceWatershed';
distancewatershed.FunctionHandle=@distanceWatershed;
distancewatershed.FunctionArgs.MedianFilterNhood.Value=MedianFilterSize;
distancewatershed.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
distancewatershed.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,distancewatershed);

polygonalassistedwatershed.InstanceName='PolygonalAssistedWatershed';
polygonalassistedwatershed.FunctionHandle=@polygonalAssistedWatershed;
polygonalassistedwatershed.FunctionArgs.MinBlobArea.Value=ObjectArea;
polygonalassistedwatershed.FunctionArgs.ImageLabel.FunctionInstance='LabelNuclei';
polygonalassistedwatershed.FunctionArgs.ImageLabel.OutputArg='LabelMatrix';
polygonalassistedwatershed.FunctionArgs.WatershedLabel.FunctionInstance='DistanceWatershed';
polygonalassistedwatershed.FunctionArgs.WatershedLabel.OutputArg='LabelMatrix';
polygonalassistedwatershed.FunctionArgs.ConvexObjectsIndex.FunctionInstance='GetConvexObjects';
polygonalassistedwatershed.FunctionArgs.ConvexObjectsIndex.OutputArg='ConvexObjectsIndex';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,polygonalassistedwatershed
);

areafilter.InstanceName='AreaFilter';
areafilter.FunctionHandle=@areaFilterLabel;
areafilter.FunctionArgs.MinArea.Value=ObjectArea;
areafilter.FunctionArgs.ObjectsLabel.FunctionInstance='PolygonalAssistedWatershed';
areafilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,areafilter);

solidityfilter.InstanceName='SolidityFilter';
solidityfilter.FunctionHandle=@solidityFilterLabel;
solidityfilter.FunctionArgs.MinSolidity.Value=MinSolidity;
solidityfilter.FunctionArgs.ObjectsLabel.FunctionInstance='AreaFilter';
solidityfilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,solidityfilter);

aoverpfilter.InstanceName='AOverPFilter';
aoverpfilter.FunctionHandle=@areaOverPerimeterFilterLabel;
aoverpfilter.FunctionArgs.MinAreaOverPerimeter.Value=MinAreaOverPerimeter;
aoverpfilter.FunctionArgs.ObjectsLabel.FunctionInstance='SolidityFilter';
aoverpfilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,aoverpfilter);

resizecytolabel.InstanceName='ResizeCytoLabel';
resizecytolabel.FunctionHandle=@resizeImage;
resizecytolabel.FunctionArgs.Scale.Value=1/ResizeImageScale;
resizecytolabel.FunctionArgs.Method.Value='nearest';
resizecytolabel.FunctionArgs.Image.FunctionInstance='AOverPFilter';
resizecytolabel.FunctionArgs.Image.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,resizecytolabel);

getshapeparameters.InstanceName='GetShapeParameters';
getshapeparameters.FunctionHandle=@getShapeParams;
getshapeparameters.FunctionArgs.LabelMatrix.FunctionInstance='ResizeCytoLabel';
getshapeparameters.FunctionArgs.LabelMatrix.OutputArg='Image';
```

```
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getshapeparameters);

isemptypreviouscellslabel.InstanceName='IsEmptyPreviousCellsLabel';
isemptypreviouscellslabel.FunctionHandle=@isEmptyFunction;
isemptypreviouscellslabel.FunctionArgs.TestVariable.Value=[];
isemptypreviouscellslabel.FunctionArgs.TestVariable.FunctionInstance='SaveCellsLabel';
isemptypreviouscellslabel.FunctionArgs.TestVariable.OutputArg='CellsLabel';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,isemptypreviouscellslabel)
;

starttracks.InstanceName='StartTracks';
starttracks.FunctionHandle=@startTracks;
starttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
starttracks.FunctionArgs.CellsLabel.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.CellsLabel.InputArg='ResizeCytoLabel_Image';
starttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
starttracks.FunctionArgs.ShapeParameters.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.ShapeParameters.InputArg='GetShapeParameters_ShapeParameters';
if_is_empty_cells_label_functions=addToFunctionChain(if_is_empty_cells_label_functions,starttrack
s);

getcurrenttracks.InstanceName='GetCurrentTracks';
getcurrenttracks.FunctionHandle=@getCurrentTracks;
getcurrenttracks.FunctionArgs.OffsetFrame.Value=-1;
getcurrenttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getcurrenttracks.FunctionArgs.TimeCol.Value=2;
getcurrenttracks.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getcurrenttracks.FunctionArgs.FrameStep.Value=FrameStep;
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg='Tracks';
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getcurrenttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getcurrenttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getcur
renttracks);

getprevioustracks.InstanceName='GetPreviousTracks';
getprevioustracks.FunctionHandle=@getCurrentTracks;
getprevioustracks.FunctionArgs.OffsetFrame.Value=-2;
getprevioustracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getprevioustracks.FunctionArgs.TimeCol.Value=2;
getprevioustracks.FunctionArgs.TrackIDCol.Value=1;
getprevioustracks.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getprevioustracks.FunctionArgs.FrameStep.Value=FrameStep;
getprevioustracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg='Tracks';
getprevioustracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getprevioustracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getprevioustracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getpre
vioustracks);

makeunassignedcellslist.InstanceName='MakeUnassignedCellsList';
makeunassignedcellslist.FunctionHandle=@makeUnassignedCellsList;
makeunassignedcellslist.FunctionArgs.CellsCentroids.FunctionInstance='IfIsEmptyPreviousCellsLabel
';
makeunassignedcellslist.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,makeun
assignedcellslist);

makeexcludedtrackslist.InstanceName='MakeExcludedTracksList';
makeexcludedtrackslist.FunctionHandle=@makeExcludedTracksList;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.FunctionInstance='MakeUnassignedCellsList'
;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.OutputArg='UnassignedCellsIDs';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,makeex
cludedtrackslist);

getcellsmeandisplacement.InstanceName='GetCellsMeanDisplacement';
getcellsmeandisplacement.FunctionHandle=@getObjectsMeanDisplacement;
getcellsmeandisplacement.FunctionArgs.Centroid1Col.Value=3;
getcellsmeandisplacement.FunctionArgs.Centroid2Col.Value=4;
getcellsmeandisplacement.FunctionArgs.CurrentTracks.FunctionInstance='GetCurrentTracks';
getcellsmeandisplacement.FunctionArgs.CurrentTracks.OutputArg='Tracks';
getcellsmeandisplacement.FunctionArgs.ObjectCentroids.FunctionInstance='IfIsEmptyPreviousCellsLab
el';
getcellsmeandisplacement.FunctionArgs.ObjectCentroids.InputArg='GetShapeParameters_Centroids';
```

```
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getcel
lsmeandisplacement);

getparamscoefficientofvariation.InstanceName='GetParamsCoefficientOfVariation';
getparamscoefficientofvariation.FunctionHandle=@getParamsCoefficientOfVariation;
getparamscoefficientofvariation.FunctionArgs.AreaCol.Value=5;
getparamscoefficientofvariation.FunctionArgs.SolidityCol.Value=11;
getparamscoefficientofvariation.FunctionArgs.Params.FunctionInstance='IfIsEmptyPreviousCellsLabel
';
getparamscoefficientofvariation.FunctionArgs.Params.InputArg='GetShapeParameters_ShapeParameters'
;
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getpar
amscoefficientofvariation);

getmaxtrackid.InstanceName='GetMaxTrackID';
getmaxtrackid.FunctionHandle=@getMaxTrackID;
getmaxtrackid.FunctionArgs.TrackIDCol.Value=1;
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg='Tracks';
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg2='Tracks';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getmax
trackid);

isnotemptyunassignedcells.InstanceName='IsNotEmptyUnassignedCells';
isnotemptyunassignedcells.FunctionHandle=@isNotEmptyFunction;
isnotemptyunassignedcells.FunctionArgs.TestVariable.FunctionInstance='AssignCellToTrackUsingAll';
isnotemptyunassignedcells.FunctionArgs.TestVariable.OutputArg='UnassignedIDs';
isnotemptyunassignedcells.FunctionArgs.TestVariable.FunctionInstance2='AssignCellsToTracksLoop';
isnotemptyunassignedcells.FunctionArgs.TestVariable.InputArg2='MakeUnassignedCellsList_Unassigned
CellsIDs';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,isnotemptyun
assignedcells);

getcurrentunassignedcell.InstanceName='GetCurrentUnassignedCell';
getcurrentunassignedcell.FunctionHandle=@getCurrentUnassignedCell;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance='AssignCellToTrackUsingAll
';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTracksLoop'
;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassign
edCellsIDs';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,getcurrentun
assignedcell);

assigncelltotrackusingall.InstanceName='AssignCellToTrackUsingAll';
assigncelltotrackusingall.FunctionHandle=@assignCellToTrackUsingAll;
assigncelltotrackusingall.FunctionArgs.CheckCellPath.Value=true;
assigncelltotrackusingall.FunctionArgs.FrontParams.Value=FrontParams;
assigncelltotrackusingall.FunctionArgs.MaxSearchRadius.Value=MaxSearchRadius;
assigncelltotrackusingall.FunctionArgs.MinSearchRadius.Value=MinSearchRadius;
assigncelltotrackusingall.FunctionArgs.SearchRadiusPct.Value=SearchRadiusPct;
assigncelltotrackusingall.FunctionArgs.RelevantParametersIndex.Value=RelevantParametersIndex;
assigncelltotrackusingall.FunctionArgs.NrParamsForSureMatch.Value=NrParamsForSureMatch;
assigncelltotrackusingall.FunctionArgs.DefaultParamWeights.Value=DefaultParamWeights;
assigncelltotrackusingall.FunctionArgs.UnknownParamWeights.Value=UnknownParamWeights;
assigncelltotrackusingall.FunctionArgs.DistanceRankingOrder.Value=DistanceRankingOrder;
assigncelltotrackusingall.FunctionArgs.DirectionRankingOrder.Value=DirectionRankingOrder;
assigncelltotrackusingall.FunctionArgs.UnknownRankingOrder.Value=UnknownRankingOrder;
assigncelltotrackusingall.FunctionArgs.MinSecondDistance.Value=MinSecondDistance;
assigncelltotrackusingall.FunctionArgs.MaxDistRatio.Value=MaxDistRatio;
assigncelltotrackusingall.FunctionArgs.MaxAngleDiff.Value=MaxAngleDiff;
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.Value=[];
assigncelltotrackusingall.FunctionArgs.TrackAssignments.Value=[];
assigncelltotrackusingall.FunctionArgs.MatchingGroups.Value=[];
assigncelltotrackusingall.FunctionArgs.MatchingGroupsStats.Value=[];
assigncelltotrackusingall.FunctionArgs.UnassignedCells.FunctionInstance='AssignCellToTrackUsingAl
l';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.FunctionInstance='AssignCellToTrackUsingAll
';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.OutputArg='ExcludedTracks';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.FunctionInstance='SetMatchingGroupIndex';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.OutputArg='ShapeParameters';
assigncelltotrackusingall.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellToTrackUsingA
ll';
assigncelltotrackusingall.FunctionArgs.TrackAssignments.OutputArg='TrackAssignments';
assigncelltotrackusingall.FunctionArgs.MatchingGroups.FunctionInstance='AssignCellToTrackUsingAll
';
assigncelltotrackusingall.FunctionArgs.MatchingGroups.OutputArg='MatchingGroups';
```

```
assigncelltotrackusingall.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTracksLoop
';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassig
nedCellsIDs';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.FunctionInstance2='AssignCellsToTracksLoop'
;
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.InputArg2='MakeExcludedTracksList_ExcludedT
racks';
assigncelltotrackusingall.FunctionArgs.CellsLabel.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CellsLabel.InputArg='ResizeCytoLabel_Image';
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.FunctionInstance='AssignCellsToTracksLo
op';
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.InputArg='SaveCellsLabel_CellsLabel';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.FunctionInstance2='AssignCellsToTracksLoop
';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.InputArg2='GetShapeParameters_ShapeParamet
ers';
assigncelltotrackusingall.FunctionArgs.CellsCentroids.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
assigncelltotrackusingall.FunctionArgs.CurrentTracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CurrentTracks.InputArg='GetCurrentTracks_Tracks';
assigncelltotrackusingall.FunctionArgs.MaxTrackID.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.MaxTrackID.InputArg='GetMaxTrackID_MaxTrackID';
assigncelltotrackusingall.FunctionArgs.Tracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.Tracks.InputArg='IfIsEmptyPreviousCellsLabel_Tracks';
assigncelltotrackusingall.FunctionArgs.MatchingGroupsStats.FunctionInstance='AssignCellsToTracksL
oop';
assigncelltotrackusingall.FunctionArgs.MatchingGroupsStats.InputArg='GetMatchingGroupMeans_Matchi
ngGroupStats';
assigncelltotrackusingall.FunctionArgs.ParamsCoeffOfVariation.FunctionInstance='AssignCellsToTrac
ksLoop';
assigncelltotrackusingall.FunctionArgs.ParamsCoeffOfVariation.InputArg='GetParamsCoefficientOfVar
iation_CoefficientOfVariation';
assigncelltotrackusingall.FunctionArgs.PreviousTracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.PreviousTracks.InputArg='GetPreviousTracks_Tracks';
assigncelltotrackusingall.FunctionArgs.TracksLayout.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
assigncelltotrackusingall.FunctionArgs.MatchingGroups.FunctionInstance2='AssignCellsToTracksLoop'
;
assigncelltotrackusingall.FunctionArgs.MatchingGroups.InputArg2='HoldMatchingGroups_ValueToHold';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,assigncellto
trackusingall);

setmatchinggroupindex.InstanceName='SetMatchingGroupIndex';
setmatchinggroupindex.FunctionHandle=@setGroupIndex;
setmatchinggroupindex.FunctionArgs.AreaCol.Value=5;
setmatchinggroupindex.FunctionArgs.GroupIDCol.Value=13;
setmatchinggroupindex.FunctionArgs.CellID.FunctionInstance='GetCurrentUnassignedCell';
setmatchinggroupindex.FunctionArgs.CellID.OutputArg='CellID';
setmatchinggroupindex.FunctionArgs.GroupIndex.FunctionInstance='AssignCellToTrackUsingAll';
setmatchinggroupindex.FunctionArgs.GroupIndex.OutputArg='GroupIndex';
setmatchinggroupindex.FunctionArgs.ShapeParameters.FunctionInstance='SetMatchingGroupIndex';
setmatchinggroupindex.FunctionArgs.ShapeParameters.OutputArg='ShapeParameters';
setmatchinggroupindex.FunctionArgs.ShapeParameters.FunctionInstance2='AssignCellsToTracksLoop';
setmatchinggroupindex.FunctionArgs.ShapeParameters.InputArg2='GetShapeParameters_ShapeParameters'
;
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,setmatchingg
roupindex);

assigncellstotracksloop.InstanceName='AssignCellsToTracksLoop';
assigncellstotracksloop.FunctionHandle=@whileLoop;
assigncellstotracksloop.FunctionArgs.TestFunction.FunctionInstance='IsNotEmptyUnassignedCells';
assigncellstotracksloop.FunctionArgs.TestFunction.OutputArg='Boolean';
assigncellstotracksloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.FunctionInstance=
'MakeUnassignedCellsList';
assigncellstotracksloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.OutputArg='Unassi
gnedCellsIDs';
assigncellstotracksloop.FunctionArgs.MakeExcludedTracksList_ExcludedTracks.FunctionInstance='Make
ExcludedTracksList';
assigncellstotracksloop.FunctionArgs.MakeExcludedTracksList_ExcludedTracks.OutputArg='ExcludedTra
cks';
assigncellstotracksloop.FunctionArgs.GetCurrentTracks_Tracks.FunctionInstance='GetCurrentTracks';
assigncellstotracksloop.FunctionArgs.GetCurrentTracks_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.GetMaxTrackID_MaxTrackID.FunctionInstance='GetMaxTrackID';
assigncellstotracksloop.FunctionArgs.GetMaxTrackID_MaxTrackID.OutputArg='MaxTrackID';
assigncellstotracksloop.FunctionArgs.IfIsEmptyPreviousCellsLabel_Tracks.FunctionInstance='IfIsEmp
tyPreviousCellsLabel';
assigncellstotracksloop.FunctionArgs.IfIsEmptyPreviousCellsLabel_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.GetMatchingGroupMeans_MatchingGroupStats.FunctionInstance='G
etMatchingGroupMeans';
```

```
assigncellstotracksloop.FunctionArgs.GetMatchingGroupMeans_MatchingGroupStats.OutputArg='Matching
GroupStats';
assigncellstotracksloop.FunctionArgs.GetParamsCoefficientOfVariation_CoefficientOfVariation.Funct
ionInstance='GetParamsCoefficientOfVariation';
assigncellstotracksloop.FunctionArgs.GetParamsCoefficientOfVariation_CoefficientOfVariation.Outpu
tArg='CoefficientOfVariation';
assigncellstotracksloop.FunctionArgs.GetPreviousTracks_Tracks.FunctionInstance='GetPreviousTracks
';
assigncellstotracksloop.FunctionArgs.GetPreviousTracks_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.ResizeCytoLabel_Image.FunctionInstance='IfIsEmptyPreviousCel
lsLabel';
assigncellstotracksloop.FunctionArgs.ResizeCytoLabel_Image.InputArg='ResizeCytoLabel_Image';
assigncellstotracksloop.FunctionArgs.SaveCellsLabel_CellsLabel.FunctionInstance='IfIsEmptyPreviou
sCellsLabel';
assigncellstotracksloop.FunctionArgs.SaveCellsLabel_CellsLabel.InputArg='SaveCellsLabel_CellsLabe
l';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_ShapeParameters.FunctionInstance='IfIsEmp
tyPreviousCellsLabel';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_ShapeParameters.InputArg='GetShapeParamet
ers_ShapeParameters';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_Centroids.FunctionInstance='IfIsEmptyPrev
iousCellsLabel';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_Centroids.InputArg='GetShapeParameters_Ce
ntroids';
assigncellstotracksloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='IfIsEmptyPre
viousCellsLabel';
assigncellstotracksloop.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout_Tra
cksLayout';
assigncellstotracksloop.FunctionArgs.HoldMatchingGroups_ValueToHold.FunctionInstance='IfIsEmptyPr
eviousCellsLabel';
assigncellstotracksloop.FunctionArgs.HoldMatchingGroups_ValueToHold.InputArg='HoldMatchingGroups_
ValueToHold';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_TrackAssignments.FunctionInstance='A
ssignCellToTrackUsingAll';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_TrackAssignments.OutputArg='TrackAss
ignments';
assigncellstotracksloop.KeepValues.SetMatchingGroupIndex_ShapeParameters.FunctionInstance='SetMat
chingGroupIndex';
assigncellstotracksloop.KeepValues.SetMatchingGroupIndex_ShapeParameters.OutputArg='ShapeParamete
rs';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.FunctionInstance='Ass
ignCellToTrackUsingAll';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.OutputArg='MatchingGr
oups';
assigncellstotracksloop.LoopFunctions=assign_cells_to_tracks_functions;
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,assign
cellstotracksloop);

continuetracks.InstanceName='ContinueTracks';
continuetracks.FunctionHandle=@continueTracks;
continuetracks.FunctionArgs.TimeFrame.Value=TimeFrame;
continuetracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
continuetracks.FunctionArgs.Tracks.OutputArg='Tracks';
continuetracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
continuetracks.FunctionArgs.Tracks.OutputArg2='Tracks';
continuetracks.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellsToTracksLoop';
continuetracks.FunctionArgs.TrackAssignments.OutputArg='AssignCellToTrackUsingAll_TrackAssignment
s';
continuetracks.FunctionArgs.ShapeParameters.FunctionInstance='AssignCellsToTracksLoop';
continuetracks.FunctionArgs.ShapeParameters.OutputArg='SetMatchingGroupIndex_ShapeParameters';
continuetracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
continuetracks.FunctionArgs.CellsCentroids.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,contin
uetracks);

getmatchinggroupmeans.InstanceName='GetMatchingGroupMeans';
getmatchinggroupmeans.FunctionHandle=@getMatchingGroupMeans;
getmatchinggroupmeans.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getmatchinggroupmeans.FunctionArgs.Tracks.OutputArg='Tracks';
getmatchinggroupmeans.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getmatchinggroupmeans.FunctionArgs.Tracks.OutputArg2='Tracks';
getmatchinggroupmeans.FunctionArgs.TracksLayout.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getmatchinggroupmeans.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getmat
chinggroupmeans);

ifisemptypreviouscellslabel.InstanceName='IfIsEmptyPreviousCellsLabel';
ifisemptypreviouscellslabel.FunctionHandle=@if_statement;
ifisemptypreviouscellslabel.FunctionArgs.PreviousCellsLabel.Value=[];
```

```
ifisemptypreviouscellslabel.FunctionArgs.MatchingGroupsStats.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.TrackAssignments.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.TestVariable.FunctionInstance='IsEmptyPreviousCellsLabel
';
ifisemptypreviouscellslabel.FunctionArgs.TestVariable.OutputArg='Boolean';
ifisemptypreviouscellslabel.FunctionArgs.ResizeCytoLabel_Image.FunctionInstance='ResizeCytoLabel'
;
ifisemptypreviouscellslabel.FunctionArgs.ResizeCytoLabel_Image.OutputArg='Image';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_LoopCounter.FunctionInstance='Segmentat
ionLoop';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_LoopCounter.OutputArg='LoopCounter';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_ShapeParameters.FunctionInstance='Get
ShapeParameters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_ShapeParameters.OutputArg='ShapeParam
eters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_Centroids.FunctionInstance='GetShapeP
arameters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_Centroids.OutputArg='Centroids';
ifisemptypreviouscellslabel.FunctionArgs.SaveCellsLabel_CellsLabel.FunctionInstance='SaveCellsLab
el';
ifisemptypreviouscellslabel.FunctionArgs.SaveCellsLabel_CellsLabel.OutputArg='CellsLabel';
ifisemptypreviouscellslabel.FunctionArgs.HoldMatchingGroups_ValueToHold.FunctionInstance='HoldMat
chingGroups';
ifisemptypreviouscellslabel.FunctionArgs.HoldMatchingGroups_ValueToHold.OutputArg='ValueToHold';
ifisemptypreviouscellslabel.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='Segmenta
tionLoop';
ifisemptypreviouscellslabel.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout
_TracksLayout';
ifisemptypreviouscellslabel.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.FunctionInstance=
'AssignCellsToTracksLoop';
ifisemptypreviouscellslabel.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.OutputArg='Assign
CellToTrackUsingAll_MatchingGroups';
ifisemptypreviouscellslabel.KeepValues.ContinueTracks_Tracks.FunctionInstance='ContinueTracks';
ifisemptypreviouscellslabel.KeepValues.ContinueTracks_Tracks.OutputArg='Tracks';
ifisemptypreviouscellslabel.ElseFunctions=else_is_empty_cells_label_functions;
ifisemptypreviouscellslabel.IfFunctions=if_is_empty_cells_label_functions;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,ifisemptypreviouscellslabe
l);

savecellslabel.InstanceName='SaveCellsLabel';
savecellslabel.FunctionHandle=@saveCellsLabel;
savecellslabel.FunctionArgs.FileRoot.Value=SegmentationFilesRoot;
savecellslabel.FunctionArgs.NumberFormat.Value=NumberFormat;
savecellslabel.FunctionArgs.CellsLabel.FunctionInstance='ResizeCytoLabel';
savecellslabel.FunctionArgs.CellsLabel.OutputArg='Image';
savecellslabel.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
savecellslabel.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,savecellslabel);

holdmatchinggroups.InstanceName='HoldMatchingGroups';
holdmatchinggroups.FunctionHandle=@holdValue;
holdmatchinggroups.FunctionArgs.ValueToHold.Value=[];
holdmatchinggroups.FunctionArgs.ValueToHold.FunctionInstance='IfIsEmptyPreviousCellsLabel';
holdmatchinggroups.FunctionArgs.ValueToHold.OutputArg='AssignCellToTrackUsingAll_MatchingGroups';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,holdmatchinggroups);

segmentationloop.InstanceName='SegmentationLoop';
segmentationloop.FunctionHandle=@forLoop;
segmentationloop.FunctionArgs.StartLoop.Value=StartFrame;
segmentationloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
segmentationloop.FunctionArgs.IncrementLoop.Value=FrameStep;
segmentationloop.FunctionArgs.MatchingGroups.Value=[];
segmentationloop.FunctionArgs.Tracks.Value=[];
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='LoadTracksLayout';
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.OutputArg='TracksLayout';
segmentationloop.KeepValues.ContinueTracks_Tracks.FunctionInstance='IfIsEmptyPreviousCellsLabel';
segmentationloop.KeepValues.ContinueTracks_Tracks.OutputArg='ContinueTracks_Tracks';
segmentationloop.LoopFunctions=image_read_loop_functions;
functions_list=addToFunctionChain(functions_list,segmentationloop);

savetracks.InstanceName='SaveTracks';
savetracks.FunctionHandle=@saveTracks;
savetracks.FunctionArgs.TracksFileName.Value=[TracksFolder '/tracks.mat'];
savetracks.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
savetracks.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,savetracks);

gettrackids.InstanceName='GetTrackIDs';
gettrackids.FunctionHandle=@getTrackIDs;
gettrackids.FunctionArgs.TrackIDCol.Value=1;
gettrackids.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
```

```
gettrackids.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,gettrackids);

detectmergecandidates.InstanceName='DetectMergeCandidates';
detectmergecandidates.FunctionHandle=@detectMergeCandidatesUsingDistance;
detectmergecandidates.FunctionArgs.MaxMergeDistance.Value=MaxMergeDistance;
detectmergecandidates.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDs';
detectmergecandidates.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
detectmergecandidates.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
detectmergecandidates.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
detectmergecandidates.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
detectmergecandidates.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,detectmergecandidates);

mergetracks.InstanceName='MergeTracks';
mergetracks.FunctionHandle=@mergeTracks;
mergetracks.FunctionArgs.FrameCount.Value=FrameCount;
mergetracks.FunctionArgs.StartFrame.Value=StartFrame;
mergetracks.FunctionArgs.TimeFrame.Value=TimeFrame;
mergetracks.FunctionArgs.SegFileRoot.Value=SegmentationFilesRoot;
mergetracks.FunctionArgs.FrameStep.Value=FrameStep;
mergetracks.FunctionArgs.NumberFormat.Value=NumberFormat;
mergetracks.FunctionArgs.TracksToBeMerged.FunctionInstance='DetectMergeCandidates';
mergetracks.FunctionArgs.TracksToBeMerged.OutputArg='TracksToBeMerged';
mergetracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
mergetracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
mergetracks.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
mergetracks.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,mergetracks);

gettrackidsaftermerge.InstanceName='GetTrackIDsAfterMerge';
gettrackidsaftermerge.FunctionHandle=@getTrackIDs;
gettrackidsaftermerge.FunctionArgs.TrackIDCol.Value=1;
gettrackidsaftermerge.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
gettrackidsaftermerge.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,gettrackidsaftermerge);

makeancestryforfirstframecells.InstanceName='MakeAncestryForFirstFrameCells';
makeancestryforfirstframecells.FunctionHandle=@makeAncestryForFirstFrameCells;
makeancestryforfirstframecells.FunctionArgs.TimeCol.Value=2;
makeancestryforfirstframecells.FunctionArgs.TrackIDCol.Value=1;
makeancestryforfirstframecells.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
makeancestryforfirstframecells.FunctionArgs.Tracks.OutputArg='Tracks';
makeancestryforfirstframecells.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDsAfterMerge';
makeancestryforfirstframecells.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
functions_list=addToFunctionChain(functions_list,makeancestryforfirstframecells);

detectmitoticevents.InstanceName='DetectMitoticEvents';
detectmitoticevents.FunctionHandle=@detectMitoticEvents;
detectmitoticevents.FunctionArgs.MaxSplitArea.Value=MaxSplitArea;
detectmitoticevents.FunctionArgs.MinSplitEccentricity.Value=MinSplitEccentricity;
detectmitoticevents.FunctionArgs.MaxSplitEccentricity.Value=MaxSplitEccentricity;
detectmitoticevents.FunctionArgs.MaxSplitDistance.Value=MaxSplitDistance;
detectmitoticevents.FunctionArgs.MinTimeForSplit.Value=MinTimeForSplit;
detectmitoticevents.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
detectmitoticevents.FunctionArgs.Tracks.OutputArg='Tracks';
detectmitoticevents.FunctionArgs.UntestedIDs.FunctionInstance='MakeAncestryForFirstFrameCells';
detectmitoticevents.FunctionArgs.UntestedIDs.OutputArg='UntestedIDs';
detectmitoticevents.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
detectmitoticevents.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
functions_list=addToFunctionChain(functions_list,detectmitoticevents);

makeancestryforcellsenteringframes.InstanceName='MakeAncestryForCellsEnteringFrames';
makeancestryforcellsenteringframes.FunctionHandle=@makeAncestryForCellsEnteringFrames;
makeancestryforcellsenteringframes.FunctionArgs.TimeCol.Value=2;
makeancestryforcellsenteringframes.FunctionArgs.TrackIDCol.Value=1;
makeancestryforcellsenteringframes.FunctionArgs.SplitCells.FunctionInstance='DetectMitoticEvents'
;
makeancestryforcellsenteringframes.FunctionArgs.SplitCells.OutputArg='SplitCells';
makeancestryforcellsenteringframes.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDsAfterMerge'
;
makeancestryforcellsenteringframes.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
makeancestryforcellsenteringframes.FunctionArgs.FirstFrameIDs.FunctionInstance='MakeAncestryForFi
rstFrameCells';
makeancestryforcellsenteringframes.FunctionArgs.FirstFrameIDs.OutputArg='FirstFrameIDs';
makeancestryforcellsenteringframes.FunctionArgs.CellsAncestry.FunctionInstance='MakeAncestryForFi
rstFrameCells';
makeancestryforcellsenteringframes.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
makeancestryforcellsenteringframes.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
makeancestryforcellsenteringframes.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,makeancestryforcellsenteringframes);
```

```
splittracks.InstanceName='SplitTracks';
splittracks.FunctionHandle=@splitTracks;
splittracks.FunctionArgs.TimeFrame.Value=TimeFrame;
splittracks.FunctionArgs.SplitCells.FunctionInstance='DetectMitoticEvents';
splittracks.FunctionArgs.SplitCells.OutputArg='SplitCells';
splittracks.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
splittracks.FunctionArgs.Tracks.OutputArg='Tracks';
splittracks.FunctionArgs.CellsAncestry.FunctionInstance='MakeAncestryForCellsEnteringFrames';
splittracks.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
splittracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
splittracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
splittracks.FunctionArgs.AncestryLayout.FunctionInstance='LoadAncestryLayout';
splittracks.FunctionArgs.AncestryLayout.OutputArg='AncestryLayout';
functions_list=addToFunctionChain(functions_list,splittracks);

removeshorttracks.InstanceName='RemoveShortTracks';
removeshorttracks.FunctionHandle=@removeShortTracks;
removeshorttracks.FunctionArgs.MinLifespan.Value=MinLifespan;
removeshorttracks.FunctionArgs.Tracks.FunctionInstance='SplitTracks';
removeshorttracks.FunctionArgs.Tracks.OutputArg='Tracks';
removeshorttracks.FunctionArgs.CellsAncestry.FunctionInstance='SplitTracks';
removeshorttracks.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
removeshorttracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
removeshorttracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
removeshorttracks.FunctionArgs.AncestryLayout.FunctionInstance='LoadAncestryLayout';
removeshorttracks.FunctionArgs.AncestryLayout.OutputArg='AncestryLayout';
functions_list=addToFunctionChain(functions_list,removeshorttracks);

saveupdatedtracks.InstanceName='SaveUpdatedTracks';
saveupdatedtracks.FunctionHandle=@saveTracks;
saveupdatedtracks.FunctionArgs.TracksFileName.Value=[AncestryFolder '/tracks.mat'];
saveupdatedtracks.FunctionArgs.Tracks.FunctionInstance='RemoveShortTracks';
saveupdatedtracks.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,saveupdatedtracks);

saveancestry.InstanceName='SaveAncestry';
saveancestry.FunctionHandle=@saveAncestry;
saveancestry.FunctionArgs.AncestryFileName.Value=[AncestryFolder '/ancestry.mat'];
saveancestry.FunctionArgs.CellsAncestry.FunctionInstance='RemoveShortTracks';
saveancestry.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
functions_list=addToFunctionChain(functions_list,saveancestry);

makeimagenamesinoverlayloop.InstanceName='MakeImageNamesInOverlayLoop';
makeimagenamesinoverlayloop.FunctionHandle=@makeImgFileName;
makeimagenamesinoverlayloop.FunctionArgs.FileBase.Value=ImageFileBase;
makeimagenamesinoverlayloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinoverlayloop.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenamesinoverlayloop.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
makeimagenamesinoverlayloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,makeimagenamesinover
layloop);

readimagesinoverlayloop.InstanceName='ReadImagesInOverlayLoop';
readimagesinoverlayloop.FunctionHandle=@readImage;
readimagesinoverlayloop.FunctionArgs.ImageChannel.Value='';
readimagesinoverlayloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInOverlayLoop';
readimagesinoverlayloop.FunctionArgs.ImageName.OutputArg='FileName';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,readimagesinoverlayl
oop);

getcurrenttracks2.InstanceName='GetCurrentTracks2';
getcurrenttracks2.FunctionHandle=@getCurrentTracks;
getcurrenttracks2.FunctionArgs.OffsetFrame.Value=0;
getcurrenttracks2.FunctionArgs.TimeFrame.Value=TimeFrame;
getcurrenttracks2.FunctionArgs.TimeCol.Value=2;
getcurrenttracks2.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks2.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getcurrenttracks2.FunctionArgs.FrameStep.Value=FrameStep;
getcurrenttracks2.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
getcurrenttracks2.FunctionArgs.CurFrame.OutputArg='LoopCounter';
getcurrenttracks2.FunctionArgs.Tracks.FunctionInstance='ImageOverlayLoop';
getcurrenttracks2.FunctionArgs.Tracks.InputArg='RemoveShortTracks_Tracks';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,getcurrenttracks2);

makematnamesinoverlayloop.InstanceName='MakeMatNamesInOverlayLoop';
makematnamesinoverlayloop.FunctionHandle=@makeImgFileName;
makematnamesinoverlayloop.FunctionArgs.FileBase.Value=SegmentationFilesRoot;
makematnamesinoverlayloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makematnamesinoverlayloop.FunctionArgs.FileExt.Value='.mat';
makematnamesinoverlayloop.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
```

```
makematnamesinoverlayloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,makematnamesinoverla
yloop);

loadcellslabel.InstanceName='LoadCellsLabel';
loadcellslabel.FunctionHandle=@loadCellsLabel;
loadcellslabel.FunctionArgs.FileName.FunctionInstance='MakeMatNamesInOverlayLoop';
loadcellslabel.FunctionArgs.FileName.OutputArg='FileName';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,loadcellslabel);

loadcolormap.InstanceName='LoadColormap';
loadcolormap.FunctionHandle=@loadColormap;
loadcolormap.FunctionArgs.FileName.Value='colormap_lines';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,loadcolormap);

displayoverlayingframe.InstanceName='DisplayOverlayingFrame';
displayoverlayingframe.FunctionHandle=@displayVariable;
displayoverlayingframe.FunctionArgs.VariableName.Value='Overlaying Frame';
displayoverlayingframe.FunctionArgs.Variable.FunctionInstance='ImageOverlayLoop';
displayoverlayingframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,displayoverlayingfra
me);

displayancestry.InstanceName='DisplayAncestry';
displayancestry.FunctionHandle=@displayAncestryData;
displayancestry.FunctionArgs.NumberFormat.Value=NumberFormat;
displayancestry.FunctionArgs.ProlDir.Value=AncestryFolder;
displayancestry.FunctionArgs.ImageFileName.Value=ImageFilesRoot;
displayancestry.FunctionArgs.DS.Value='/';
displayancestry.FunctionArgs.Image.FunctionInstance='ReadImagesInOverlayLoop';
displayancestry.FunctionArgs.Image.OutputArg='Image';
displayancestry.FunctionArgs.CurrentTracks.FunctionInstance='GetCurrentTracks2';
displayancestry.FunctionArgs.CurrentTracks.OutputArg='Tracks';
displayancestry.FunctionArgs.CellsLabel.FunctionInstance='LoadCellsLabel';
displayancestry.FunctionArgs.CellsLabel.OutputArg='LabelMatrix';
displayancestry.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.CurFrame.OutputArg='LoopCounter';
displayancestry.FunctionArgs.ColorMap.FunctionInstance='LoadColormap';
displayancestry.FunctionArgs.ColorMap.OutputArg='Colormap';
displayancestry.FunctionArgs.CellsAncestry.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.CellsAncestry.InputArg='RemoveShortTracks_CellsAncestry';
displayancestry.FunctionArgs.TracksLayout.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
displayancestry.FunctionArgs.AncestryLayout.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.AncestryLayout.InputArg='LoadAncestryLayout_AncestryLayout';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,displayancestry);

imageoverlayloop.InstanceName='ImageOverlayLoop';
imageoverlayloop.FunctionHandle=@forLoop;
imageoverlayloop.FunctionArgs.StartLoop.Value=StartFrame;
imageoverlayloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
imageoverlayloop.FunctionArgs.IncrementLoop.Value=FrameStep;
imageoverlayloop.FunctionArgs.RemoveShortTracks_Tracks.FunctionInstance='RemoveShortTracks';
imageoverlayloop.FunctionArgs.RemoveShortTracks_Tracks.OutputArg='Tracks';
imageoverlayloop.FunctionArgs.RemoveShortTracks_CellsAncestry.FunctionInstance='RemoveShortTracks
';
imageoverlayloop.FunctionArgs.RemoveShortTracks_CellsAncestry.OutputArg='CellsAncestry';
imageoverlayloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='LoadTracksLayout';
imageoverlayloop.FunctionArgs.LoadTracksLayout_TracksLayout.OutputArg='TracksLayout';
imageoverlayloop.FunctionArgs.LoadAncestryLayout_AncestryLayout.FunctionInstance='LoadAncestryLay
out';
imageoverlayloop.FunctionArgs.LoadAncestryLayout_AncestryLayout.OutputArg='AncestryLayout';
imageoverlayloop.LoopFunctions=image_overlay_loop_functions;
functions_list=addToFunctionChain(functions_list,imageoverlayloop);

saveancestryspreadsheets.InstanceName='SaveAncestrySpreadsheets';
saveancestryspreadsheets.FunctionHandle=@saveAncestrySpreadsheets;
saveancestryspreadsheets.FunctionArgs.ShapesXlsFile.Value=ShapesSpreadsheet;
saveancestryspreadsheets.FunctionArgs.ProlXlsFile.Value=AncestrySpreadsheet;
saveancestryspreadsheets.FunctionArgs.Tracks.FunctionInstance='RemoveShortTracks';
saveancestryspreadsheets.FunctionArgs.Tracks.OutputArg='Tracks';
saveancestryspreadsheets.FunctionArgs.CellsAncestry.FunctionInstance='RemoveShortTracks';
saveancestryspreadsheets.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
saveancestryspreadsheets.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
saveancestryspreadsheets.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
functions_list=addToFunctionChain(functions_list,saveancestryspreadsheets);


global dependencies_list;
global dependencies_index;
dependencies_list={};
```

```
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayFluoNuclCyto.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayFluoNuclCyto()
%Used to assign cytoplasm regions based on a nuclear stain image. Once the cytoplasm is
%assigned to each nucleus the cell shape properties are extracted and saved to disk.

global functions_list;
functions_list=[];
%script variables
ImageFolder='C:/walter/20111229 area experiment lncap34/1/';
NucleiImage=[ImageFolder '20111229143216-998-R04-C06.jpg'];
CytoImage=[ImageFolder '20111229143216-999-R04-C06.jpg'];
OutputFolder=ImageFolder;
CellOutlines=[ImageFolder 'outlines.jpg'];
CellIntensities=[OutputFolder 'intensity_data.mat'];
ImageChannel='r';
%end script variables


readnuclei.InstanceName='ReadNuclei';
readnuclei.FunctionHandle=@readImage;
readnuclei.FunctionArgs.ImageChannel.Value=ImageChannel;
readnuclei.FunctionArgs.ImageName.Value=NucleiImage;
functions_list=addToFunctionChain(functions_list,readnuclei);

readcyto.InstanceName='ReadCyto';
readcyto.FunctionHandle=@readImage;
readcyto.FunctionArgs.ImageChannel.Value=ImageChannel;
readcyto.FunctionArgs.ImageName.Value=CytoImage;
functions_list=addToFunctionChain(functions_list,readcyto);

normalizenuclearimage.InstanceName='NormalizeNuclearImage';
normalizenuclearimage.FunctionHandle=@imNorm;
normalizenuclearimage.FunctionArgs.IntegerClass.Value='uint16';
normalizenuclearimage.FunctionArgs.RawImage.FunctionInstance='ReadNuclei';
normalizenuclearimage.FunctionArgs.RawImage.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,normalizenuclearimage);

normalizecytoimage.InstanceName='NormalizeCytoImage';
normalizecytoimage.FunctionHandle=@imNorm;
normalizecytoimage.FunctionArgs.IntegerClass.Value='uint16';
normalizecytoimage.FunctionArgs.RawImage.FunctionInstance='ReadCyto';
normalizecytoimage.FunctionArgs.RawImage.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,normalizecytoimage);

thresholdnuclei.InstanceName='ThresholdNuclei';
thresholdnuclei.FunctionHandle=@generateBinImgUsingLocAvg;
thresholdnuclei.FunctionArgs.ClearBorder.Value=true;
thresholdnuclei.FunctionArgs.ClearBorderDist.Value=0;
thresholdnuclei.FunctionArgs.Strel.Value='disk';
thresholdnuclei.FunctionArgs.StrelSize.Value=10;
thresholdnuclei.FunctionArgs.BrightnessThresholdPct.Value=1.05;
thresholdnuclei.FunctionArgs.Image.FunctionInstance='ReadNuclei';
thresholdnuclei.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,thresholdnuclei);

showimage.InstanceName='ShowImage';
showimage.FunctionHandle=@displayImage;
showimage.FunctionArgs.FigureNr.Value=1;
showimage.FunctionArgs.Image.FunctionInstance='ThresholdNuclei';
showimage.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,showimage);

intensityfilter.InstanceName='IntensityFilter';
intensityfilter.FunctionHandle=@generateBinImgUsingGlobInt;
intensityfilter.FunctionArgs.ClearBorder.Value=false;
intensityfilter.FunctionArgs.ClearBorderDist.Value=0;
intensityfilter.FunctionArgs.IntensityThresholdPct.Value=0.15;
intensityfilter.FunctionArgs.Image.FunctionInstance='NormalizeCytoImage';
intensityfilter.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,intensityfilter);

clearnoisepixels.InstanceName='ClearNoisePixels';
clearnoisepixels.FunctionHandle=@clearSmallObjects;
```

```
clearnoisepixels.FunctionArgs.MinObjectArea.Value=60;
clearnoisepixels.FunctionArgs.Image.FunctionInstance='ThresholdNuclei';
clearnoisepixels.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,clearnoisepixels);

clearnoisepixelscyto.InstanceName='ClearNoisePixelsCyto';
clearnoisepixelscyto.FunctionHandle=@clearSmallObjects;
clearnoisepixelscyto.FunctionArgs.MinObjectArea.Value=60;
clearnoisepixelscyto.FunctionArgs.Image.FunctionInstance='IntensityFilter';
clearnoisepixelscyto.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,clearnoisepixelscyto);

fillnucleiholes.InstanceName='FillNucleiHoles';
fillnucleiholes.FunctionHandle=@fillHoles;
fillnucleiholes.FunctionArgs.Image.FunctionInstance='ClearNoisePixels';
fillnucleiholes.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,fillnucleiholes);

labelnuclei.InstanceName='LabelNuclei';
labelnuclei.FunctionHandle=@labelObjects;
labelnuclei.FunctionArgs.Image.FunctionInstance='FillNucleiHoles';
labelnuclei.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,labelnuclei);

segmentcyto.InstanceName='SegmentCyto';
segmentcyto.FunctionHandle=@segmentCytoUsingNuclei;
segmentcyto.FunctionArgs.CytoImage.FunctionInstance='ClearNoisePixelsCyto';
segmentcyto.FunctionArgs.CytoImage.OutputArg='Image';
segmentcyto.FunctionArgs.NuclearLabel.FunctionInstance='LabelNuclei';
segmentcyto.FunctionArgs.NuclearLabel.OutputArg='LabelMatrix';
functions_list=addToFunctionChain(functions_list,segmentcyto);

getcytointensities.InstanceName='GetCytoIntensities';
getcytointensities.FunctionHandle=@getObjectIntensities;
getcytointensities.FunctionArgs.IntensityImage.FunctionInstance='ReadCyto';
getcytointensities.FunctionArgs.IntensityImage.OutputArg='Image';
getcytointensities.FunctionArgs.LabelMatrix.FunctionInstance='SegmentCyto';
getcytointensities.FunctionArgs.LabelMatrix.OutputArg='LabelMatrix';
functions_list=addToFunctionChain(functions_list,getcytointensities);

savecelloutlines.InstanceName='SaveCellOutlines';
savecelloutlines.FunctionHandle=@displayObjectOutlines;
savecelloutlines.FunctionArgs.FileName.Value=CellOutlines;
savecelloutlines.FunctionArgs.ShowIDs.Value=true;
savecelloutlines.FunctionArgs.Image.FunctionInstance='NormalizeCytoImage';
savecelloutlines.FunctionArgs.Image.OutputArg='Image';
savecelloutlines.FunctionArgs.ObjectsLabel.FunctionInstance='SegmentCyto';
savecelloutlines.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
functions_list=addToFunctionChain(functions_list,savecelloutlines);

saveintensities.InstanceName='SaveIntensities';
saveintensities.FunctionHandle=@saveWrapper;
saveintensities.FunctionArgs.FileName.Value=CellIntensities;
saveintensities.FunctionArgs.SaveData.FunctionInstance='GetCytoIntensities';
saveintensities.FunctionArgs.SaveData.OutputArg='MeanIntensities';
functions_list=addToFunctionChain(functions_list,saveintensities);

shownuclei.InstanceName='ShowNuclei';
shownuclei.FunctionHandle=@displayImage;
shownuclei.FunctionArgs.FigureNr.Value=1;
shownuclei.FunctionArgs.Image.FunctionInstance='FillNucleiHoles';
shownuclei.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,shownuclei);

showoriginal.InstanceName='ShowOriginal';
showoriginal.FunctionHandle=@displayImage;
showoriginal.FunctionArgs.FigureNr.Value=2;
showoriginal.FunctionArgs.Image.FunctionInstance='NormalizeCytoImage';
showoriginal.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,showoriginal);

showcyto.InstanceName='ShowCyto';
showcyto.FunctionHandle=@displayImage;
showcyto.FunctionArgs.FigureNr.Value=3;
showcyto.FunctionArgs.Image.FunctionInstance='IntensityFilter';
showcyto.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,showcyto);

showlabel.InstanceName='ShowLabel';
showlabel.FunctionHandle=@showLabelMatrix;
showlabel.FunctionArgs.FigureNr.Value=4;
```

```
showlabel.FunctionArgs.LabelMatrix.FunctionInstance='SegmentCyto';
showlabel.FunctionArgs.LabelMatrix.OutputArg='LabelMatrix';
functions_list=addToFunctionChain(functions_list,showlabel);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayFluoNuclCytoCellavista.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayFluoNuclCytoCellavista()
%Used to assign cytoplasm regions based on a nuclear stain image. Once the cytoplasm is
%       assigned to each nucleus the cell shape
%properties  are  extracted  and  saved  to  disk.

global functions_list;
functions_list=[];
%script variables
ImageFolder='C:/walter/20120103 area experiment lncap34/plastic+dcca-/';
NucleiImage=[ImageFolder '20120105170615-6766-R03-C08.jpg'];
CytoImage=[ImageFolder '20120105170615-6767-R03-C08.jpg'];
OutputFolder=[ImageFolder 'output/'];
CellOutlines=[OutputFolder 'outlines.jpg'];
CellIntensities=[OutputFolder 'intensity_data.mat'];
ImageChannel='r';
NucleiThreshold=0.15;
CytoThreshold=0.03;
%end script variables


readnuclei.InstanceName='ReadNuclei';
readnuclei.FunctionHandle=@readImage;
readnuclei.FunctionArgs.ImageChannel.Value=ImageChannel;
readnuclei.FunctionArgs.ImageName.Value=NucleiImage;
functions_list=addToFunctionChain(functions_list,readnuclei);

readcyto.InstanceName='ReadCyto';
readcyto.FunctionHandle=@readImage;
readcyto.FunctionArgs.ImageChannel.Value=ImageChannel;
readcyto.FunctionArgs.ImageName.Value=CytoImage;
functions_list=addToFunctionChain(functions_list,readcyto);

normalizenuclearimage.InstanceName='NormalizeNuclearImage';
normalizenuclearimage.FunctionHandle=@imNorm;
normalizenuclearimage.FunctionArgs.IntegerClass.Value='uint16';
normalizenuclearimage.FunctionArgs.RawImage.FunctionInstance='ReadNuclei';
normalizenuclearimage.FunctionArgs.RawImage.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,normalizenuclearimage);

normalizecytoimage.InstanceName='NormalizeCytoImage';
normalizecytoimage.FunctionHandle=@imNorm;
normalizecytoimage.FunctionArgs.IntegerClass.Value='uint16';
normalizecytoimage.FunctionArgs.RawImage.FunctionInstance='ReadCyto';
normalizecytoimage.FunctionArgs.RawImage.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,normalizecytoimage);

thresholdnuclei.InstanceName='ThresholdNuclei';
thresholdnuclei.FunctionHandle=@generateBinImgUsingGlobInt;
thresholdnuclei.FunctionArgs.ClearBorder.Value=true;
thresholdnuclei.FunctionArgs.ClearBorderDist.Value=0;
thresholdnuclei.FunctionArgs.IntensityThresholdPct.Value=NucleiThreshold;
thresholdnuclei.FunctionArgs.Image.FunctionInstance='NormalizeNuclearImage';
thresholdnuclei.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,thresholdnuclei);

thresholdcyto.InstanceName='ThresholdCyto';
thresholdcyto.FunctionHandle=@generateBinImgUsingGlobInt;
thresholdcyto.FunctionArgs.ClearBorder.Value=false;
thresholdcyto.FunctionArgs.ClearBorderDist.Value=0;
thresholdcyto.FunctionArgs.IntensityThresholdPct.Value=CytoThreshold;
thresholdcyto.FunctionArgs.Image.FunctionInstance='NormalizeCytoImage';
thresholdcyto.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,thresholdcyto);
```

```
test.InstanceName='Test';
test.FunctionHandle=@generateBinImgUsingGradient;
test.FunctionArgs.ClearBorder.Value='false';
test.FunctionArgs.ClearBorderDist.Value=0;
test.FunctionArgs.GradientThreshold.Value=2000;
test.FunctionArgs.Image.FunctionInstance='ReadCyto';
test.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,test);

showimage.InstanceName='ShowImage';
showimage.FunctionHandle=@displayImage;
showimage.FunctionArgs.FigureNr.Value=1;
showimage.FunctionArgs.Image.FunctionInstance='ThresholdNuclei';
showimage.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,showimage);

clearnoisepixels.InstanceName='ClearNoisePixels';
clearnoisepixels.FunctionHandle=@clearSmallObjects;
clearnoisepixels.FunctionArgs.MinObjectArea.Value=60;
clearnoisepixels.FunctionArgs.Image.FunctionInstance='ThresholdNuclei';
clearnoisepixels.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,clearnoisepixels);

clearnoisepixelscyto.InstanceName='ClearNoisePixelsCyto';
clearnoisepixelscyto.FunctionHandle=@clearSmallObjects;
clearnoisepixelscyto.FunctionArgs.MinObjectArea.Value=60;
clearnoisepixelscyto.FunctionArgs.Image.FunctionInstance='ThresholdCyto';
clearnoisepixelscyto.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,clearnoisepixelscyto);

fillnucleiholes.InstanceName='FillNucleiHoles';
fillnucleiholes.FunctionHandle=@fillHoles;
fillnucleiholes.FunctionArgs.Image.FunctionInstance='ClearNoisePixels';
fillnucleiholes.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,fillnucleiholes);

labelnuclei.InstanceName='LabelNuclei';
labelnuclei.FunctionHandle=@labelObjects;
labelnuclei.FunctionArgs.Image.FunctionInstance='FillNucleiHoles';
labelnuclei.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,labelnuclei);

getconvexnuclei.InstanceName='GetConvexNuclei';
getconvexnuclei.FunctionHandle=@getConvexObjects;
getconvexnuclei.FunctionArgs.ApproximationDistance.Value=2.5;
getconvexnuclei.FunctionArgs.Image.FunctionInstance='FillNucleiHoles';
getconvexnuclei.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,getconvexnuclei);

gaussianblur.InstanceName='GaussianBlur';
gaussianblur.FunctionHandle=@gaussianFilter;
gaussianblur.FunctionArgs.KernelSize.Value=[17,17];
gaussianblur.FunctionArgs.StandardDev.Value=13;
gaussianblur.FunctionArgs.Image.FunctionInstance='NormalizeNuclearImage';
gaussianblur.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,gaussianblur);

regionalmax.InstanceName='RegionalMax';
regionalmax.FunctionHandle=@imregionalmax_Wrapper;
regionalmax.FunctionArgs.Image.FunctionInstance='GaussianBlur';
regionalmax.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,regionalmax);

getmarkers.InstanceName='GetMarkers';
getmarkers.FunctionHandle=@combineImages;
getmarkers.FunctionArgs.CombineOperation.Value='AND';
getmarkers.FunctionArgs.Image1.FunctionInstance='FillNucleiHoles';
getmarkers.FunctionArgs.Image1.OutputArg='Image';
getmarkers.FunctionArgs.Image2.FunctionInstance='RegionalMax';
getmarkers.FunctionArgs.Image2.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,getmarkers);

intensitywatershed.InstanceName='IntensityWatershed';
intensitywatershed.FunctionHandle=@intensityWatershed;
intensitywatershed.FunctionArgs.Image.FunctionInstance='NormalizeNuclearImage';
intensitywatershed.FunctionArgs.Image.OutputArg='Image';
intensitywatershed.FunctionArgs.MarkerImage.FunctionInstance='GetMarkers';
intensitywatershed.FunctionArgs.MarkerImage.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,intensitywatershed);

segmentnuclei.InstanceName='SegmentNuclei';
```

```
segmentnuclei.FunctionHandle=@polygonalAssistedWatershed;
segmentnuclei.FunctionArgs.MinBlobArea.Value=60;
segmentnuclei.FunctionArgs.MaxObjectSize.Value=2000;
segmentnuclei.FunctionArgs.ConvexObjectsIndex.FunctionInstance='GetConvexNuclei';
segmentnuclei.FunctionArgs.ConvexObjectsIndex.OutputArg='ConvexObjectsIndex';
segmentnuclei.FunctionArgs.ImageLabel.FunctionInstance='LabelNuclei';
segmentnuclei.FunctionArgs.ImageLabel.OutputArg='LabelMatrix';
segmentnuclei.FunctionArgs.WatershedLabel.FunctionInstance='IntensityWatershed';
segmentnuclei.FunctionArgs.WatershedLabel.OutputArg='LabelMatrix';
functions_list=addToFunctionChain(functions_list,segmentnuclei);

segmentcyto.InstanceName='SegmentCyto';
segmentcyto.FunctionHandle=@segmentCytoUsingNuclei;
segmentcyto.FunctionArgs.CytoImage.FunctionInstance='ClearNoisePixelsCyto';
segmentcyto.FunctionArgs.CytoImage.OutputArg='Image';
segmentcyto.FunctionArgs.NuclearLabel.FunctionInstance='SegmentNuclei';
segmentcyto.FunctionArgs.NuclearLabel.OutputArg='LabelMatrix';
functions_list=addToFunctionChain(functions_list,segmentcyto);

clearcellsatedges.InstanceName='ClearCellsAtEdges';
clearcellsatedges.FunctionHandle=@clearBorderObjectsInLabel;
clearcellsatedges.FunctionArgs.LabelMatrix.FunctionInstance='SegmentCyto';
clearcellsatedges.FunctionArgs.LabelMatrix.OutputArg='LabelMatrix';
functions_list=addToFunctionChain(functions_list,clearcellsatedges);

getshapeparameters.InstanceName='GetShapeParameters';
getshapeparameters.FunctionHandle=@getRegionProps;
getshapeparameters.FunctionArgs.LabelMatrix.FunctionInstance='ClearCellsAtEdges';
getshapeparameters.FunctionArgs.LabelMatrix.OutputArg='LabelMatrix';
functions_list=addToFunctionChain(functions_list,getshapeparameters);

makeoutputdirectory.InstanceName='MakeOutputDirectory';
makeoutputdirectory.FunctionHandle=@mkdir_Wrapper;
makeoutputdirectory.FunctionArgs.DirectoryName.Value=OutputFolder;
functions_list=addToFunctionChain(functions_list,makeoutputdirectory);

getcytofilename.InstanceName='GetCytoFileName';
getcytofilename.FunctionHandle=@getFileInfo;
getcytofilename.FunctionArgs.DirSep.Value='/';
getcytofilename.FunctionArgs.PathName.Value=CytoImage;
functions_list=addToFunctionChain(functions_list,getcytofilename);

makeshapeparampath.InstanceName='MakeShapeParamPath';
makeshapeparampath.FunctionHandle=@concatenateText;
makeshapeparampath.FunctionArgs.Text1.Value=OutputFolder;
makeshapeparampath.FunctionArgs.Text3.Value='.csv';
makeshapeparampath.FunctionArgs.Text2.FunctionInstance='GetCytoFileName';
makeshapeparampath.FunctionArgs.Text2.OutputArg='FileName';
functions_list=addToFunctionChain(functions_list,makeshapeparampath);

saveshapeparameters.InstanceName='SaveShapeParameters';
saveshapeparameters.FunctionHandle=@saveRegionPropsSpreadsheets;
saveshapeparameters.FunctionArgs.RegionProps.FunctionInstance='GetShapeParameters';
saveshapeparameters.FunctionArgs.RegionProps.OutputArg='RegionProps';
saveshapeparameters.FunctionArgs.SpreadsheetFileName.FunctionInstance='MakeShapeParamPath';
saveshapeparameters.FunctionArgs.SpreadsheetFileName.OutputArg='String';
functions_list=addToFunctionChain(functions_list,saveshapeparameters);

makecelloutlinespath.InstanceName='MakeCellOutlinesPath';
makecelloutlinespath.FunctionHandle=@concatenateText;
makecelloutlinespath.FunctionArgs.Text1.Value=OutputFolder;
makecelloutlinespath.FunctionArgs.Text3.Value='.jpg';
makecelloutlinespath.FunctionArgs.Text2.FunctionInstance='GetCytoFileName';
makecelloutlinespath.FunctionArgs.Text2.OutputArg='FileName';
functions_list=addToFunctionChain(functions_list,makecelloutlinespath);

savecelloutlines.InstanceName='SaveCellOutlines';
savecelloutlines.FunctionHandle=@displayObjectOutlines;
savecelloutlines.FunctionArgs.ShowIDs.Value=true;
savecelloutlines.FunctionArgs.Image.FunctionInstance='NormalizeCytoImage';
savecelloutlines.FunctionArgs.Image.OutputArg='Image';
savecelloutlines.FunctionArgs.ObjectsLabel.FunctionInstance='ClearCellsAtEdges';
savecelloutlines.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
savecelloutlines.FunctionArgs.FileName.FunctionInstance='MakeCellOutlinesPath';
savecelloutlines.FunctionArgs.FileName.OutputArg='String';
functions_list=addToFunctionChain(functions_list,savecelloutlines);

showcyto.InstanceName='ShowCyto';
showcyto.FunctionHandle=@displayImage;
showcyto.FunctionArgs.FigureNr.Value=1;
showcyto.FunctionArgs.Image.FunctionInstance='NormalizeCytoImage';
```

```
showcyto.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,showcyto);

shownuclei.InstanceName='ShowNuclei';
shownuclei.FunctionHandle=@displayImage;
shownuclei.FunctionArgs.FigureNr.Value=2;
shownuclei.FunctionArgs.Image.FunctionInstance='NormalizeNuclearImage';
shownuclei.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,shownuclei);

shownuclearlabels.InstanceName='ShowNuclearLabels';
shownuclearlabels.FunctionHandle=@showLabelMatrix;
shownuclearlabels.FunctionArgs.FigureNr.Value=3;
shownuclearlabels.FunctionArgs.LabelMatrix.FunctionInstance='SegmentNuclei';
shownuclearlabels.FunctionArgs.LabelMatrix.OutputArg='LabelMatrix';
functions_list=addToFunctionChain(functions_list,shownuclearlabels);

showcytolabels.InstanceName='ShowCytoLabels';
showcytolabels.FunctionHandle=@showLabelMatrix;
showcytolabels.FunctionArgs.FigureNr.Value=4;
showcytolabels.FunctionArgs.LabelMatrix.FunctionInstance='ClearCellsAtEdges';
showcytolabels.FunctionArgs.LabelMatrix.OutputArg='LabelMatrix';
functions_list=addToFunctionChain(functions_list,showcytolabels);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayFluoNuclTestCA.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayFluoNuclTestCA()
%assayFluoNuclTestCA - This module is used to track cells that have been stained using a
%        nuclear stain  using a custom
%CellAnimation  algorithm.  This  algorithm  is  more  accurate
%and   flexible than  the   nearest-neighbor algorithm  at the
% expense of   execution speed.   The assay   tracks
%the  cells, detects    mitotic events  and  records tracking
% data,  cell shape   parameters  and ancestry   information
%to spreadsheets.    For  each tracked image,   an
%overlayed image is   saved   to disk that  displays the
%   detected cell  outlines, cell    IDs and cell
% generation. ImageFolder    - String variable  that specifies the
%  absolute location of  the directory which     contains
%the   time-lapse images. An   example  of  such a
% string variable would   be  'c:/sample  images/high-density'. ImageFilesRoot
%- String variable   specifying the root  image file
%name. The root   image file name   for
% a set of images  is the image  file name
%   of any of the   images without  the number
% or     the file extension. For  example, if the
% file name        is 'Experiment-0002_Position(8)_t021.tif' the root
%image file name will   be 'Experiment-0002_Position(8)_t'.   ImageExtension
% - String variable specifying  the image file extension  including the preceding
%  dot.      For example  if  the
%file name is  'image003.jpg' the image   extension is
%  '.jpg'. StartFrame - Number  specifying  the first image  in
%the sequence   to be  analyzed.     The minimum
% value  for  this variable depends  on  the numbering
% of the  image    sequence so if   the
% first  image in the  sequence  is 'image003.tif'
%then the  minimum value   is 3. FrameCount  - Number
% specifying  how many images  from  the image
% sequence should be processed. TimeFrame   - Number specifying  the time
%   between  consecutive images in  minutes.  FrameStep -
% Number   specifying the step size when   reading images. Set
% this variable     to  1  to read every
% image in    the   sequence, 2 to read every
%  other image and    so    on. NumberFormat
% - String value specifying the   number of    digits
%  in the image file names   in  the sequence.
% For example    if the   image file name is
%'image020.jpg'   the value for       the NumberFormat
%is '%03d', while   if the file name    is 'image000020.jpg'
%   the value  should  be '%06d'. MaxFramesMissing  -
```

```
% Number specifying for   how    many frames  a
%cell may  be disappear before its     track is
%   ended. OutputFolder  - The folder where the  overlayed
% images and track data   will    be saved.
%By   default  this value is set to   a folder
%named      'output' within  the folder where  the
% images    to be analyzed are  located. AncestryFolder
% - The   folder  where the  overlayed images and ancestry
%data   will be  saved.    By   default
% this value is set to   a    folder named
%'ancestry' within  the output    folder. AncestrySpreadsheet -   The
% path name  to the spreadsheet containing  the   ancestry data.
% By    default this  value   is set to
%a file   named    'ancestry.csv' within  the ancestry
%folder. ShapesSpreadsheet  -  The path name   to   the
%spreadsheet  containing the position and  shape   properties for
%   each cell in the timelapse   sequence at every
%time point.   By    default this  is  set
% to to  a  file named 'shapes.csv'   within
%the  ancestry  folder. TracksFolder - The  folder where  the label
%   matrixes   containing the cell outlines are   saved.
%By    default this   value is   set to
% a folder named   'track' within  the  output folder. SegmentationFilesRoot
%  -    The  root file name of the label
% matrixes containing  the cell     outlines.   ImageFileBase
%- The path name  to the  images. This   value is
%  generated from   the ImageFolder    and the
% ImageFilesRoot and should not  be changed. BrightnessThresholdPct    -
%  Number  specifying the percentage threshold value for   the image
%generated by the     generateBinImgUsingLocAvg     filter. Any
%pixel in  the original  image smaller   than the
%threshold value times   the    corresponding value in
% the local average image  below this value  will  be
%   set  to  zero while the rest   will
%be  set to one.     ObjectArea -  Number
%specifying the  threshold area  for the  clearSmallObjects, polygonalAssistedWatershed   and
%areaFilterLabel    filters. Objects   below this value   will
%be removed     from the filtered image.  Strel - String
% variable  specifying    the  type  of filter used
%to generate the   local  average   image in
%  generateBinImgUsingLocAvg. Currently 'disk' is   the only  value supported.
%StrelSize -  Number specifying   the  size    of
%the local neighborhood used to calculate   the  average
%  for each pixel in  the local average   image generated
%by    the  generateBinImgUsingLocAvg  module. ClearBorder -  Boolean value
%specifying   whether  objects  next to or    touching
% the image border in    the  binary images
%generated by  the generateBinImgUsingLocAvg   module  will  be  erased
% (true) or not  (false). ClearBorderDist   -  Number specifying
%    how close to the border objects may be
%and still  be      erased  if the ClearBorder
% parameter is  set  to true  in  the generateBinImgUsingLocAvg module.
%   MedianFilterSize   - Number specifying the   size of
% the median filter   used by  the distanceWatershed   module.
%   Setting  this   to a higher integer value will
% reduce the  number of       objects detected
% by the module and can    be used  to prevent
%  oversegmentation.  MinSolidity  - Number specifying   a threshold
%solidity  value for  the  solidityFilterLabelObjects filter. Objects   whose
%solidity    is below this   value will   be
%removed from   the filtered  image. MinAreaOverPerimeter -  Number
%specifying  a  threshold   AOP ratio value for the areaOverPerimeterFilterLabel
% filter. Objects  with       an AOP ratio
% smaller than this value will be    removed. ResizeImageScale  -
%Number    specifying  by what ratio the images   will
%be   resized  before they  are   processed. By
% default   this  value is set to 0.5.   Used
%by   the  resizeImage module.   ApproximationDistance -  Number specifying
%how close the    convex  hull  in   the
%getConvexObjects module approximates the     object outline.  By default this
%    value is set   to 2.5. Setting it to
%a     lower  value  will    result
%in convex hulls that more closely   resemble  the   object
% outlines   however  this increases the   chance of
% detecting  insignificant concavities. MaxSearchRadius - Number   specifying  the
% absolute  lower bound   for the search radius to  prevent
%selecting     too  few  candidate   objects
%for a track. Used by   assignCellToTrackUsingAll module. MinSearchRadius  - Number specifying
%     the absolute higher   bound for the search
%radius  to  prevent  selecting   too  many
% candidate  objects for a track. Used by   assignCellToTrackUsingAll  module.
```

```
%  MinSecondDistance  - Number specifying  the  minimum significant distance
% between the    closest  candidate object to a  track
% and  the    second  closest. Used   to
%determine when distance should  be  used as     a
%ranking  parameter.   Used   by assignCellToTrackUsingAll module. MaxDistRatio -
% Number specifying the  maximum allowed distance    ratio
%between the two   nearest candidate objects.   If  the
%ratio  is higher  than  this    value distance
%ranking will not be used.    Used  by
%assignCellToTrackUsingAll  module. MaxAngleDiff -  Number specifying the maximum allowed
%angle difference    between  a track and a candidate
% object.  If   the angle  is larger   than
% this value direction ranking  will  not be
%used  for this object.   Used   by assignCellToTrackUsingAll module. NrParamsForSureMatch
%-    Number specifying the  minimum number of
%closest  matches  between a candidate   object parameters  and
%a  track's   object  parameters that  make  the
%candidate  object a   sure match to  the track.
%Used  by     assignCellToTrackUsingAll  module. SearchRadiusPct - Number specifying
%the size of the     neighborhood from    which
% candidate objects for matching  the   track are  selected.
% It is  a multiple   of   the distance to
%the  nearest   candidate  in  the  current  frame.
%Setting this   variable   equal to 1 turns  this
%  module  into a nearest-neighbor algorithm (only     the
% nearest  cell can be   a candidate). It does
%  not make  sense  to have  a  value
% lower  than 1.   Used by assignCellToTrackUsingAll module. MaxMergeDistance -
%Number  specifying  the     maximum  distance that one
%track may  be from another track      for the
%  duration   and still be considered for possible   merging
%with  the    other track.   Used by  detectMergeCandidatesUsingDistance
%  module. MaxSplitArea -  Number  specifying  the maximum  area
% a  nucleus  may  be and  still be
%considered  as a  part     of a possible mitotic
%event. Used by     detectMitoticEvents  module. MaxSplitDistance  - Number
%specifying   the maximum distance a    new  nucleus may
% be  from another nucleus   and    still
%be considered   as part of a possible mitotic   event.
%Used  by  detectMitoticEvents     module. MinSplitEccentricity -  Number
%specifying  the minimum eccentricity a     new nucleus may
%  have  and still be  considered as part   of
% a  possible    mitotic event.  Used by detectMitoticEvents module.
%MaxSplitEccentricity    - Number specifying  the  maximum  eccentricity a
% new   nucleus  may  have  and still be
%considered    as part of a  possible
%mitotic event. Used by  detectMitoticEvents module. MinTimeForSplit   -  Number
%specifying the  minimum time  in   minutes a track needs
%  to  exist before    it is considered for a
% possible    mitotic   event. Used   by detectMitoticEvents
%module. MinLifespan - Number specifying  the   minimum  length  in
%  frames  a frame has  to  be  to
%not be removed    by the removeShortTracks     module.
% FrontParams - Numeric array specifying a set   of   column
% indices from   the shape  and motility  parameters  matrix.
%   The parameters in  those  columns  will be
%heavily weighted,   and  have  more   influence in determining
% the    best match for a  track from
%  a list of objects.   Used by   assignCellToTrackUsingAll
%module.   DefaultParamWeights -  Numeric array  specifying  a set of
%weights that     is  assigned to   each shape
% and motility parameter   based on   its   prediction
%power.   Parameters with  high prediction power are   assigned
% high  weights  and  parameters  with  low prediction power
% are  assigned  lower weights.  Used  by
%assignCellToTrackUsingAll module. DistanceRankingOrder    - Numeric   array specifying the
default
%order of shape   and   motility   parameters for
% slow  moving objects when it cannot  be
%determined based on  prediction  power.  Used by   assignCellToTrackUsingAll
% module. DirectionRankingOrder  -  Numeric  array specifying the default
%order    of shape and motility  parameters for
%fast   moving directional objects  when it cannot   be
% determined   based on  prediction power.  Used by  assignCellToTrackUsingAll
%module.  RelevantParametersIndex -     Boolean  array specifying
%column indexes  in the shape and  motility matrix that   have
%    been   determined to  be irrelevant for tracking.
%This  indicates    to the module  not  to
%  use the parameters   those  columns  in computing
%track assignment probabilities.  The   order of column
%indexes   is provided in TracksLayout  variable. Used by   assignCellToTrackUsingAll
```

```
%  module.  UnknownParamWeights - Numeric    array specifying a set
%of weights to  be      assigned to shape
%and  motility  parameters when   the prediction   power of
% the  parameters  cannot be determined. Used    by
%  assignCellToTrackUsingAll module. UnknownRankingOrder -  Numeric array     specifying
% the order of the  shape and motility  parameters when
% their    predictive  power cannot  be determined. If the
% objects  cannot be     categorized as either slow-moving
%   or  fast directional the parameter  order provided in
% this    variable  is used.  Used by
%assignCellToTrackUsingAll module. Important  Modules  -    areaFilterLabel,
areaOverPerimeterFilterLabel, assignCellToTrackUsingAll,
%clearSmallObjects,  detectMergeCandidatesUsingDistance,   detectMitoticEvents,
distanceWatershed, generateBinImgUsingLocAvg, getConvexObjects, polygonalAssistedWatershed,
removeShortTracks,
%segmentObjectsUsingMarkers,  solidityFilterLabel, splitTracks.

global functions_list;
functions_list=[];
%script variables
ImageFolder='C:/darren/movie2';
ImageFilesRoot='DsRed - Confocal - n';
ImageExtension='.tif';
StartFrame=1;
FrameCount=72;
TimeFrame=15;
FrameStep=1;
NumberFormat='%06d';
MaxFramesMissing=6;
OutputFolder=[ImageFolder '/output'];
AncestryFolder=[OutputFolder '/ancestry'];
AncestrySpreadsheet=[AncestryFolder 'ancestry.csv'];
ShapesSpreadsheet=[AncestryFolder 'shapes.csv'];
TracksFolder=[OutputFolder '/track'];
SegmentationFilesRoot=[TracksFolder '/grayscale'];
ImageFileBase=[ImageFolder '/' ImageFilesRoot];
BrightnessThresholdPct=1.1;
ObjectArea=30;
Strel='disk';
StrelSize=10;
ClearBorder=true;
ClearBorderDist=2;
MedianFilterSize=3;
MinSolidity=0.69;
MinAreaOverPerimeter=1.5;
ResizeImageScale=0.5;
ApproximationDistance=2.4;
MaxSearchRadius=Inf;
MinSearchRadius=0;
MinSecondDistance=5;
MaxDistRatio=0.6;
MaxAngleDiff=0.35;
NrParamsForSureMatch=5;
SearchRadiusPct=1.5;
MaxMergeDistance=23;
MaxSplitArea=400;
MaxSplitDistance=45;
MinSplitEccentricity=0.5;
MaxSplitEccentricity=0.95;
MinTimeForSplit=900;
MinLifespan=30;
FrontParams=[];
DefaultParamWeights=[34 21 13 8 5 3 2 2 2];
DistanceRankingOrder=[1 3 4 5 6 7 8 9 2];
DirectionRankingOrder=[2 3 4 5 6 7 8 9 1];
RelevantParametersIndex=[true true true false true false true true false];
UnknownParamWeights=[5 3 1 1 1 1 1 1 1];
UnknownRankingOrder=[1 2 3 4 5 6 7 8 9];
%end script variables

assign_cells_to_tracks_functions=[];
else_is_empty_cells_label_functions=[];
if_is_empty_cells_label_functions=[];
image_read_loop_functions=[];
image_overlay_loop_functions=[];

loadtrackslayout.InstanceName='LoadTracksLayout';
loadtrackslayout.FunctionHandle=@loadTracksLayout;
loadtrackslayout.FunctionArgs.FileName.Value='tracks_layout.mat';
functions_list=addToFunctionChain(functions_list,loadtrackslayout);
```

```
loadancestrylayout.InstanceName='LoadAncestryLayout';
loadancestrylayout.FunctionHandle=@loadAncestryLayout;
loadancestrylayout.FunctionArgs.FileName.Value='ancestry_layout.mat';
functions_list=addToFunctionChain(functions_list,loadancestrylayout);

displaycurframe.InstanceName='DisplayCurFrame';
displaycurframe.FunctionHandle=@displayVariable;
displaycurframe.FunctionArgs.VariableName.Value='Current Tracking Frame';
displaycurframe.FunctionArgs.Variable.FunctionInstance='SegmentationLoop';
displaycurframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,displaycurframe);

makeimagenamesinsegmentationloop.InstanceName='MakeImageNamesInSegmentationLoop';
makeimagenamesinsegmentationloop.FunctionHandle=@makeImgFileName;
makeimagenamesinsegmentationloop.FunctionArgs.FileBase.Value=ImageFileBase;
makeimagenamesinsegmentationloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinsegmentationloop.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makeimagenamesinsegmentati
onloop);

readimagesinsegmentationloop.InstanceName='ReadImagesInSegmentationLoop';
readimagesinsegmentationloop.FunctionHandle=@readImage;
readimagesinsegmentationloop.FunctionArgs.ImageChannel.Value='';
readimagesinsegmentationloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInSegmentatio
nLoop';
readimagesinsegmentationloop.FunctionArgs.ImageName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,readimagesinsegmentationlo
op);

normalizeimageto16bit.InstanceName='NormalizeImageTo16Bit';
normalizeimageto16bit.FunctionHandle=@imNorm;
normalizeimageto16bit.FunctionArgs.IntegerClass.Value='uint16';
normalizeimageto16bit.FunctionArgs.RawImage.FunctionInstance='ReadImagesInSegmentationLoop';
normalizeimageto16bit.FunctionArgs.RawImage.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,normalizeimageto16bit);

resizeimage.InstanceName='ResizeImage';
resizeimage.FunctionHandle=@resizeImage;
resizeimage.FunctionArgs.Scale.Value=ResizeImageScale;
resizeimage.FunctionArgs.Method.Value='bicubic';
resizeimage.FunctionArgs.Image.FunctionInstance='NormalizeImageTo16Bit';
resizeimage.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,resizeimage);

cytobrightnesslocalaveragingfilter.InstanceName='CytoBrightnessLocalAveragingFilter';
cytobrightnesslocalaveragingfilter.FunctionHandle=@generateBinImgUsingLocAvg;
cytobrightnesslocalaveragingfilter.FunctionArgs.Strel.Value=Strel;
cytobrightnesslocalaveragingfilter.FunctionArgs.StrelSize.Value=StrelSize;
cytobrightnesslocalaveragingfilter.FunctionArgs.BrightnessThresholdPct.Value=BrightnessThresholdP
ct;
cytobrightnesslocalaveragingfilter.FunctionArgs.ClearBorder.Value=ClearBorder;
cytobrightnesslocalaveragingfilter.FunctionArgs.ClearBorderDist.Value=ClearBorderDist;
cytobrightnesslocalaveragingfilter.FunctionArgs.Image.FunctionInstance='ResizeImage';
cytobrightnesslocalaveragingfilter.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,cytobrightnesslocalaveragi
ngfilter);

fillholescytoplasmimages.InstanceName='FillHolesCytoplasmImages';
fillholescytoplasmimages.FunctionHandle=@fillHoles;
fillholescytoplasmimages.FunctionArgs.Image.FunctionInstance='CytoBrightnessLocalAveragingFilter'
;
fillholescytoplasmimages.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,fillholescytoplasmimages);

clearsmallcells.InstanceName='ClearSmallCells';
clearsmallcells.FunctionHandle=@clearSmallObjects;
clearsmallcells.FunctionArgs.MinObjectArea.Value=ObjectArea;
clearsmallcells.FunctionArgs.Image.FunctionInstance='FillHolesCytoplasmImages';
clearsmallcells.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,clearsmallcells);

nuclbrightnesslocalaveragingfilter.InstanceName='NuclBrightnessLocalAveragingFilter';
nuclbrightnesslocalaveragingfilter.FunctionHandle=@generateBinImgUsingLocAvg;
nuclbrightnesslocalaveragingfilter.FunctionArgs.Strel.Value=Strel;
nuclbrightnesslocalaveragingfilter.FunctionArgs.StrelSize.Value=StrelSize;
nuclbrightnesslocalaveragingfilter.FunctionArgs.BrightnessThresholdPct.Value=BrightnessThresholdP
ct;
nuclbrightnesslocalaveragingfilter.FunctionArgs.ClearBorder.Value=ClearBorder;
nuclbrightnesslocalaveragingfilter.FunctionArgs.ClearBorderDist.Value=ClearBorderDist;
```

```
nuclbrightnesslocalaveragingfilter.FunctionArgs.Image.FunctionInstance='ResizeImage';
nuclbrightnesslocalaveragingfilter.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,nuclbrightnesslocalaveragi
ngfilter);

fillholesnuclearimages.InstanceName='FillHolesNuclearImages';
fillholesnuclearimages.FunctionHandle=@fillHoles;
fillholesnuclearimages.FunctionArgs.Image.FunctionInstance='NuclBrightnessLocalAveragingFilter';
fillholesnuclearimages.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,fillholesnuclearimages);

clearsmallnuclei.InstanceName='ClearSmallNuclei';
clearsmallnuclei.FunctionHandle=@clearSmallObjects;
clearsmallnuclei.FunctionArgs.MinObjectArea.Value=ObjectArea;
clearsmallnuclei.FunctionArgs.Image.FunctionInstance='FillHolesNuclearImages';
clearsmallnuclei.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,clearsmallnuclei);

combinenuclearandcytoplasmimages.InstanceName='CombineNuclearAndCytoplasmImages';
combinenuclearandcytoplasmimages.FunctionHandle=@combineImages;
combinenuclearandcytoplasmimages.FunctionArgs.CombineOperation.Value='AND';
combinenuclearandcytoplasmimages.FunctionArgs.Image1.FunctionInstance='ClearSmallNuclei';
combinenuclearandcytoplasmimages.FunctionArgs.Image1.OutputArg='Image';
combinenuclearandcytoplasmimages.FunctionArgs.Image2.FunctionInstance='ClearSmallCells';
combinenuclearandcytoplasmimages.FunctionArgs.Image2.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,combinenuclearandcytoplasm
images);

reconstructcytoplasmimage.InstanceName='ReconstructCytoplasmImage';
reconstructcytoplasmimage.FunctionHandle=@reconstructObjects;
reconstructcytoplasmimage.FunctionArgs.GuideImage.FunctionInstance='CombineNuclearAndCytoplasmIma
ges';
reconstructcytoplasmimage.FunctionArgs.GuideImage.OutputArg='Image';
reconstructcytoplasmimage.FunctionArgs.ImageToReconstruct.FunctionInstance='ClearSmallNuclei';
reconstructcytoplasmimage.FunctionArgs.ImageToReconstruct.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,reconstructcytoplasmimage)
;

labelnuclei.InstanceName='LabelNuclei';
labelnuclei.FunctionHandle=@labelObjects;
labelnuclei.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
labelnuclei.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,labelnuclei);

labelcytoplasm.InstanceName='LabelCytoplasm';
labelcytoplasm.FunctionHandle=@labelObjects;
labelcytoplasm.FunctionArgs.Image.FunctionInstance='ReconstructCytoplasmImage';
labelcytoplasm.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,labelcytoplasm);

getconvexobjects.InstanceName='GetConvexObjects';
getconvexobjects.FunctionHandle=@getConvexObjects;
getconvexobjects.FunctionArgs.ApproximationDistance.Value=ApproximationDistance;
getconvexobjects.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
getconvexobjects.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getconvexobjects);

distancewatershed.InstanceName='DistanceWatershed';
distancewatershed.FunctionHandle=@distanceWatershed;
distancewatershed.FunctionArgs.MedianFilterNhood.Value=MedianFilterSize;
distancewatershed.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
distancewatershed.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,distancewatershed);

polygonalassistedwatershed.InstanceName='PolygonalAssistedWatershed';
polygonalassistedwatershed.FunctionHandle=@polygonalAssistedWatershed;
polygonalassistedwatershed.FunctionArgs.MinBlobArea.Value=ObjectArea;
polygonalassistedwatershed.FunctionArgs.ImageLabel.FunctionInstance='LabelNuclei';
polygonalassistedwatershed.FunctionArgs.ImageLabel.OutputArg='LabelMatrix';
polygonalassistedwatershed.FunctionArgs.WatershedLabel.FunctionInstance='DistanceWatershed';
polygonalassistedwatershed.FunctionArgs.WatershedLabel.OutputArg='LabelMatrix';
polygonalassistedwatershed.FunctionArgs.ConvexObjectsIndex.FunctionInstance='GetConvexObjects';
polygonalassistedwatershed.FunctionArgs.ConvexObjectsIndex.OutputArg='ConvexObjectsIndex';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,polygonalassistedwatershed
);

segmentobjectsusingmarkers.InstanceName='SegmentObjectsUsingMarkers';
segmentobjectsusingmarkers.FunctionHandle=@segmentObjectsUsingMarkers;
segmentobjectsusingmarkers.FunctionArgs.MarkersLabel.FunctionInstance='PolygonalAssistedWatershed
';
segmentobjectsusingmarkers.FunctionArgs.MarkersLabel.OutputArg='LabelMatrix';
```

```
segmentobjectsusingmarkers.FunctionArgs.ObjectsLabel.FunctionInstance='LabelCytoplasm';
segmentobjectsusingmarkers.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,segmentobjectsusingmarkers
);

areafilter.InstanceName='AreaFilter';
areafilter.FunctionHandle=@areaFilterLabel;
areafilter.FunctionArgs.MinArea.Value=ObjectArea;
areafilter.FunctionArgs.ObjectsLabel.FunctionInstance='SegmentObjectsUsingMarkers';
areafilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,areafilter);

solidityfilter.InstanceName='SolidityFilter';
solidityfilter.FunctionHandle=@solidityFilterLabel;
solidityfilter.FunctionArgs.MinSolidity.Value=MinSolidity;
solidityfilter.FunctionArgs.ObjectsLabel.FunctionInstance='AreaFilter';
solidityfilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,solidityfilter);

aoverpfilter.InstanceName='AOverPFilter';
aoverpfilter.FunctionHandle=@areaOverPerimeterFilterLabel;
aoverpfilter.FunctionArgs.MinAreaOverPerimeter.Value=MinAreaOverPerimeter;
aoverpfilter.FunctionArgs.ObjectsLabel.FunctionInstance='SolidityFilter';
aoverpfilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,aoverpfilter);

resizecytolabel.InstanceName='ResizeCytoLabel';
resizecytolabel.FunctionHandle=@resizeImage;
resizecytolabel.FunctionArgs.Scale.Value=1/ResizeImageScale;
resizecytolabel.FunctionArgs.Method.Value='nearest';
resizecytolabel.FunctionArgs.Image.FunctionInstance='AOverPFilter';
resizecytolabel.FunctionArgs.Image.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,resizecytolabel);

makeexportfilenames.InstanceName='MakeExportFileNames';
makeexportfilenames.FunctionHandle=@makeImgFileName;
makeexportfilenames.FunctionArgs.FileBase.Value='C:\darren\movie2\cellanimation output\exported
labels\CALabel';
makeexportfilenames.FunctionArgs.FileExt.Value='.tif';
makeexportfilenames.FunctionArgs.NumberFmt.Value=NumberFormat;
makeexportfilenames.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
makeexportfilenames.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makeexportfilenames);

exportlabeltoimage.InstanceName='ExportLabelToImage';
exportlabeltoimage.FunctionHandle=@exportLabelToImage;
exportlabeltoimage.FunctionArgs.Format.Value='tif';
exportlabeltoimage.FunctionArgs.Image.FunctionInstance='ResizeCytoLabel';
exportlabeltoimage.FunctionArgs.Image.OutputArg='Image';
exportlabeltoimage.FunctionArgs.FileName.FunctionInstance='MakeExportFileNames';
exportlabeltoimage.FunctionArgs.FileName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,exportlabeltoimage);

getshapeparameters.InstanceName='GetShapeParameters';
getshapeparameters.FunctionHandle=@getShapeParams;
getshapeparameters.FunctionArgs.LabelMatrix.FunctionInstance='ResizeCytoLabel';
getshapeparameters.FunctionArgs.LabelMatrix.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getshapeparameters);

isemptypreviouscellslabel.InstanceName='IsEmptyPreviousCellsLabel';
isemptypreviouscellslabel.FunctionHandle=@isEmptyFunction;
isemptypreviouscellslabel.FunctionArgs.TestVariable.Value=[];
isemptypreviouscellslabel.FunctionArgs.TestVariable.FunctionInstance='SaveCellsLabel';
isemptypreviouscellslabel.FunctionArgs.TestVariable.OutputArg='CellsLabel';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,isemptypreviouscellslabel)
;

starttracks.InstanceName='StartTracks';
starttracks.FunctionHandle=@startTracks;
starttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
starttracks.FunctionArgs.CellsLabel.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.CellsLabel.InputArg='ResizeCytoLabel_Image';
starttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
starttracks.FunctionArgs.ShapeParameters.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.ShapeParameters.InputArg='GetShapeParameters_ShapeParameters';
if_is_empty_cells_label_functions=addToFunctionChain(if_is_empty_cells_label_functions,starttrack
s);

getcurrenttracks.InstanceName='GetCurrentTracks';
getcurrenttracks.FunctionHandle=@getCurrentTracks;
```

```
getcurrenttracks.FunctionArgs.OffsetFrame.Value=-1;
getcurrenttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getcurrenttracks.FunctionArgs.TimeCol.Value=2;
getcurrenttracks.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getcurrenttracks.FunctionArgs.FrameStep.Value=FrameStep;
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg='Tracks';
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getcurrenttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getcurrenttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getcur
renttracks);

getprevioustracks.InstanceName='GetPreviousTracks';
getprevioustracks.FunctionHandle=@getCurrentTracks;
getprevioustracks.FunctionArgs.OffsetFrame.Value=-2;
getprevioustracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getprevioustracks.FunctionArgs.TimeCol.Value=2;
getprevioustracks.FunctionArgs.TrackIDCol.Value=1;
getprevioustracks.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getprevioustracks.FunctionArgs.FrameStep.Value=FrameStep;
getprevioustracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg='Tracks';
getprevioustracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getprevioustracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getprevioustracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getpre
vioustracks);

makeunassignedcellslist.InstanceName='MakeUnassignedCellsList';
makeunassignedcellslist.FunctionHandle=@makeUnassignedCellsList;
makeunassignedcellslist.FunctionArgs.CellsCentroids.FunctionInstance='IfIsEmptyPreviousCellsLabel
';
makeunassignedcellslist.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,makeun
assignedcellslist);

makeexcludedtrackslist.InstanceName='MakeExcludedTracksList';
makeexcludedtrackslist.FunctionHandle=@makeExcludedTracksList;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.FunctionInstance='MakeUnassignedCellsList'
;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.OutputArg='UnassignedCellsIDs';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,makeex
cludedtrackslist);

getcellsmeandisplacement.InstanceName='GetCellsMeanDisplacement';
getcellsmeandisplacement.FunctionHandle=@getObjectsMeanDisplacement;
getcellsmeandisplacement.FunctionArgs.Centroid1Col.Value=3;
getcellsmeandisplacement.FunctionArgs.Centroid2Col.Value=4;
getcellsmeandisplacement.FunctionArgs.CurrentTracks.FunctionInstance='GetCurrentTracks';
getcellsmeandisplacement.FunctionArgs.CurrentTracks.OutputArg='Tracks';
getcellsmeandisplacement.FunctionArgs.ObjectCentroids.FunctionInstance='IfIsEmptyPreviousCellsLab
el';
getcellsmeandisplacement.FunctionArgs.ObjectCentroids.InputArg='GetShapeParameters_Centroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getcel
lsmeandisplacement);

getparamscoefficientofvariation.InstanceName='GetParamsCoefficientOfVariation';
getparamscoefficientofvariation.FunctionHandle=@getParamsCoefficientOfVariation;
getparamscoefficientofvariation.FunctionArgs.AreaCol.Value=5;
getparamscoefficientofvariation.FunctionArgs.SolidityCol.Value=11;
getparamscoefficientofvariation.FunctionArgs.Params.FunctionInstance='IfIsEmptyPreviousCellsLabel
';
getparamscoefficientofvariation.FunctionArgs.Params.InputArg='GetShapeParameters_ShapeParameters'
;
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getpar
amscoefficientofvariation);

getmaxtrackid.InstanceName='GetMaxTrackID';
getmaxtrackid.FunctionHandle=@getMaxTrackID;
getmaxtrackid.FunctionArgs.TrackIDCol.Value=1;
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg='Tracks';
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg2='Tracks';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getmax
trackid);
```

```
isnotemptyunassignedcells.InstanceName='IsNotEmptyUnassignedCells';
isnotemptyunassignedcells.FunctionHandle=@isNotEmptyFunction;
isnotemptyunassignedcells.FunctionArgs.TestVariable.FunctionInstance='AssignCellToTrackUsingAll';
isnotemptyunassignedcells.FunctionArgs.TestVariable.OutputArg='UnassignedIDs';
isnotemptyunassignedcells.FunctionArgs.TestVariable.FunctionInstance2='AssignCellsToTracksLoop';
isnotemptyunassignedcells.FunctionArgs.TestVariable.InputArg2='MakeUnassignedCellsList_Unassigned
CellsIDs';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,isnotemptyun
assignedcells);

getcurrentunassignedcell.InstanceName='GetCurrentUnassignedCell';
getcurrentunassignedcell.FunctionHandle=@getCurrentUnassignedCell;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance='AssignCellToTrackUsingAll
';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTracksLoop'
;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassign
edCellsIDs';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,getcurrentun
assignedcell);

assigncelltotrackusingall.InstanceName='AssignCellToTrackUsingAll';
assigncelltotrackusingall.FunctionHandle=@assignCellToTrackUsingAll;
assigncelltotrackusingall.FunctionArgs.CheckCellPath.Value=true;
assigncelltotrackusingall.FunctionArgs.FrontParams.Value=FrontParams;
assigncelltotrackusingall.FunctionArgs.MaxSearchRadius.Value=MaxSearchRadius;
assigncelltotrackusingall.FunctionArgs.MinSearchRadius.Value=MinSearchRadius;
assigncelltotrackusingall.FunctionArgs.SearchRadiusPct.Value=SearchRadiusPct;
assigncelltotrackusingall.FunctionArgs.RelevantParametersIndex.Value=RelevantParametersIndex;
assigncelltotrackusingall.FunctionArgs.NrParamsForSureMatch.Value=NrParamsForSureMatch;
assigncelltotrackusingall.FunctionArgs.DefaultParamWeights.Value=DefaultParamWeights;
assigncelltotrackusingall.FunctionArgs.UnknownParamWeights.Value=UnknownParamWeights;
assigncelltotrackusingall.FunctionArgs.DistanceRankingOrder.Value=DistanceRankingOrder;
assigncelltotrackusingall.FunctionArgs.DirectionRankingOrder.Value=DirectionRankingOrder;
assigncelltotrackusingall.FunctionArgs.UnknownRankingOrder.Value=UnknownRankingOrder;
assigncelltotrackusingall.FunctionArgs.MinSecondDistance.Value=MinSecondDistance;
assigncelltotrackusingall.FunctionArgs.MaxDistRatio.Value=MaxDistRatio;
assigncelltotrackusingall.FunctionArgs.MaxAngleDiff.Value=MaxAngleDiff;
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.Value=[];
assigncelltotrackusingall.FunctionArgs.TrackAssignments.Value=[];
assigncelltotrackusingall.FunctionArgs.MatchingGroups.Value=[];
assigncelltotrackusingall.FunctionArgs.MatchingGroupsStats.Value=[];
assigncelltotrackusingall.FunctionArgs.UnassignedCells.FunctionInstance='AssignCellToTrackUsingAl
l';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.FunctionInstance='AssignCellToTrackUsingAll
';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.OutputArg='ExcludedTracks';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.FunctionInstance='SetMatchingGroupIndex';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.OutputArg='ShapeParameters';
assigncelltotrackusingall.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellToTrackUsingA
ll';
assigncelltotrackusingall.FunctionArgs.TrackAssignments.OutputArg='TrackAssignments';
assigncelltotrackusingall.FunctionArgs.MatchingGroups.FunctionInstance='AssignCellToTrackUsingAll
';
assigncelltotrackusingall.FunctionArgs.MatchingGroups.OutputArg='MatchingGroups';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTracksLoop
';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassig
nedCellsIDs';
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.FunctionInstance2='AssignCellsToTracksLoop'
;
assigncelltotrackusingall.FunctionArgs.ExcludedTracks.InputArg2='MakeExcludedTracksList_ExcludedT
racks';
assigncelltotrackusingall.FunctionArgs.CellsLabel.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CellsLabel.InputArg='ResizeCytoLabel_Image';
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.FunctionInstance='AssignCellsToTracksLo
op';
assigncelltotrackusingall.FunctionArgs.PreviousCellsLabel.InputArg='SaveCellsLabel_CellsLabel';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.FunctionInstance2='AssignCellsToTracksLoop
';
assigncelltotrackusingall.FunctionArgs.ShapeParameters.InputArg2='GetShapeParameters_ShapeParamet
ers';
assigncelltotrackusingall.FunctionArgs.CellsCentroids.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
assigncelltotrackusingall.FunctionArgs.CurrentTracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CurrentTracks.InputArg='GetCurrentTracks_Tracks';
assigncelltotrackusingall.FunctionArgs.MaxTrackID.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.MaxTrackID.InputArg='GetMaxTrackID_MaxTrackID';
assigncelltotrackusingall.FunctionArgs.Tracks.FunctionInstance='AssignCellsToTracksLoop';
```

```
assigncelltotrackusingall.FunctionArgs.Tracks.InputArg='IfIsEmptyPreviousCellsLabel_Tracks';
assigncelltotrackusingall.FunctionArgs.MatchingGroupsStats.FunctionInstance='AssignCellsToTracksL
oop';
assigncelltotrackusingall.FunctionArgs.MatchingGroupsStats.InputArg='GetMatchingGroupMeans_Matchi
ngGroupStats';
assigncelltotrackusingall.FunctionArgs.ParamsCoeffOfVariation.FunctionInstance='AssignCellsToTrac
ksLoop';
assigncelltotrackusingall.FunctionArgs.ParamsCoeffOfVariation.InputArg='GetParamsCoefficientOfVar
iation_CoefficientOfVariation';
assigncelltotrackusingall.FunctionArgs.PreviousTracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.PreviousTracks.InputArg='GetPreviousTracks_Tracks';
assigncelltotrackusingall.FunctionArgs.TracksLayout.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
assigncelltotrackusingall.FunctionArgs.MatchingGroups.FunctionInstance2='AssignCellsToTracksLoop'
;
assigncelltotrackusingall.FunctionArgs.MatchingGroups.InputArg2='HoldMatchingGroups_ValueToHold';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,assigncellto
trackusingall);

setmatchinggroupindex.InstanceName='SetMatchingGroupIndex';
setmatchinggroupindex.FunctionHandle=@setGroupIndex;
setmatchinggroupindex.FunctionArgs.AreaCol.Value=5;
setmatchinggroupindex.FunctionArgs.GroupIDCol.Value=13;
setmatchinggroupindex.FunctionArgs.CellID.FunctionInstance='GetCurrentUnassignedCell';
setmatchinggroupindex.FunctionArgs.CellID.OutputArg='CellID';
setmatchinggroupindex.FunctionArgs.GroupIndex.FunctionInstance='AssignCellToTrackUsingAll';
setmatchinggroupindex.FunctionArgs.GroupIndex.OutputArg='GroupIndex';
setmatchinggroupindex.FunctionArgs.ShapeParameters.FunctionInstance='SetMatchingGroupIndex';
setmatchinggroupindex.FunctionArgs.ShapeParameters.OutputArg='ShapeParameters';
setmatchinggroupindex.FunctionArgs.ShapeParameters.FunctionInstance2='AssignCellsToTracksLoop';
setmatchinggroupindex.FunctionArgs.ShapeParameters.InputArg2='GetShapeParameters_ShapeParameters'
;
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,setmatchingg
roupindex);

assigncellstotracksloop.InstanceName='AssignCellsToTracksLoop';
assigncellstotracksloop.FunctionHandle=@whileLoop;
assigncellstotracksloop.FunctionArgs.TestFunction.FunctionInstance='IsNotEmptyUnassignedCells';
assigncellstotracksloop.FunctionArgs.TestFunction.OutputArg='Boolean';
assigncellstotracksloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.FunctionInstance=
'MakeUnassignedCellsList';
assigncellstotracksloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.OutputArg='Unassi
gnedCellsIDs';
assigncellstotracksloop.FunctionArgs.MakeExcludedTracksList_ExcludedTracks.FunctionInstance='Make
ExcludedTracksList';
assigncellstotracksloop.FunctionArgs.MakeExcludedTracksList_ExcludedTracks.OutputArg='ExcludedTra
cks';
assigncellstotracksloop.FunctionArgs.GetCurrentTracks_Tracks.FunctionInstance='GetCurrentTracks';
assigncellstotracksloop.FunctionArgs.GetCurrentTracks_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.GetMaxTrackID_MaxTrackID.FunctionInstance='GetMaxTrackID';
assigncellstotracksloop.FunctionArgs.GetMaxTrackID_MaxTrackID.OutputArg='MaxTrackID';
assigncellstotracksloop.FunctionArgs.IfIsEmptyPreviousCellsLabel_Tracks.FunctionInstance='IfIsEmp
tyPreviousCellsLabel';
assigncellstotracksloop.FunctionArgs.IfIsEmptyPreviousCellsLabel_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.GetMatchingGroupMeans_MatchingGroupStats.FunctionInstance='G
etMatchingGroupMeans';
assigncellstotracksloop.FunctionArgs.GetMatchingGroupMeans_MatchingGroupStats.OutputArg='Matching
GroupStats';
assigncellstotracksloop.FunctionArgs.GetParamsCoefficientOfVariation_CoefficientOfVariation.Funct
ionInstance='GetParamsCoefficientOfVariation';
assigncellstotracksloop.FunctionArgs.GetParamsCoefficientOfVariation_CoefficientOfVariation.Outpu
tArg='CoefficientOfVariation';
assigncellstotracksloop.FunctionArgs.GetPreviousTracks_Tracks.FunctionInstance='GetPreviousTracks
';
assigncellstotracksloop.FunctionArgs.GetPreviousTracks_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.ResizeCytoLabel_Image.FunctionInstance='IfIsEmptyPreviousCel
lsLabel';
assigncellstotracksloop.FunctionArgs.ResizeCytoLabel_Image.InputArg='ResizeCytoLabel_Image';
assigncellstotracksloop.FunctionArgs.SaveCellsLabel_CellsLabel.FunctionInstance='IfIsEmptyPreviou
sCellsLabel';
assigncellstotracksloop.FunctionArgs.SaveCellsLabel_CellsLabel.InputArg='SaveCellsLabel_CellsLabe
l';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_ShapeParameters.FunctionInstance='IfIsEmp
tyPreviousCellsLabel';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_ShapeParameters.InputArg='GetShapeParamet
ers_ShapeParameters';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_Centroids.FunctionInstance='IfIsEmptyPrev
iousCellsLabel';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_Centroids.InputArg='GetShapeParameters_Ce
ntroids';
```

```
assigncellstotracksloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='IfIsEmptyPre
viousCellsLabel';
assigncellstotracksloop.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout_Tra
cksLayout';
assigncellstotracksloop.FunctionArgs.HoldMatchingGroups_ValueToHold.FunctionInstance='IfIsEmptyPr
eviousCellsLabel';
assigncellstotracksloop.FunctionArgs.HoldMatchingGroups_ValueToHold.InputArg='HoldMatchingGroups_
ValueToHold';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_TrackAssignments.FunctionInstance='A
ssignCellToTrackUsingAll';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_TrackAssignments.OutputArg='TrackAss
ignments';
assigncellstotracksloop.KeepValues.SetMatchingGroupIndex_ShapeParameters.FunctionInstance='SetMat
chingGroupIndex';
assigncellstotracksloop.KeepValues.SetMatchingGroupIndex_ShapeParameters.OutputArg='ShapeParamete
rs';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.FunctionInstance='Ass
ignCellToTrackUsingAll';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.OutputArg='MatchingGr
oups';
assigncellstotracksloop.LoopFunctions=assign_cells_to_tracks_functions;
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,assign
cellstotracksloop);

continuetracks.InstanceName='ContinueTracks';
continuetracks.FunctionHandle=@continueTracks;
continuetracks.FunctionArgs.TimeFrame.Value=TimeFrame;
continuetracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
continuetracks.FunctionArgs.Tracks.OutputArg='Tracks';
continuetracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
continuetracks.FunctionArgs.Tracks.OutputArg2='Tracks';
continuetracks.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellsToTracksLoop';
continuetracks.FunctionArgs.TrackAssignments.OutputArg='AssignCellToTrackUsingAll_TrackAssignment
s';
continuetracks.FunctionArgs.ShapeParameters.FunctionInstance='AssignCellsToTracksLoop';
continuetracks.FunctionArgs.ShapeParameters.OutputArg='SetMatchingGroupIndex_ShapeParameters';
continuetracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
continuetracks.FunctionArgs.CellsCentroids.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,contin
uetracks);

getmatchinggroupmeans.InstanceName='GetMatchingGroupMeans';
getmatchinggroupmeans.FunctionHandle=@getMatchingGroupMeans;
getmatchinggroupmeans.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getmatchinggroupmeans.FunctionArgs.Tracks.OutputArg='Tracks';
getmatchinggroupmeans.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getmatchinggroupmeans.FunctionArgs.Tracks.OutputArg2='Tracks';
getmatchinggroupmeans.FunctionArgs.TracksLayout.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getmatchinggroupmeans.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getmat
chinggroupmeans);

ifisemptypreviouscellslabel.InstanceName='IfIsEmptyPreviousCellsLabel';
ifisemptypreviouscellslabel.FunctionHandle=@if_statement;
ifisemptypreviouscellslabel.FunctionArgs.PreviousCellsLabel.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.MatchingGroupsStats.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.TrackAssignments.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.TestVariable.FunctionInstance='IsEmptyPreviousCellsLabel
';
ifisemptypreviouscellslabel.FunctionArgs.TestVariable.OutputArg='Boolean';
ifisemptypreviouscellslabel.FunctionArgs.ResizeCytoLabel_Image.FunctionInstance='ResizeCytoLabel'
;
ifisemptypreviouscellslabel.FunctionArgs.ResizeCytoLabel_Image.OutputArg='Image';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_LoopCounter.FunctionInstance='Segmentat
ionLoop';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_LoopCounter.OutputArg='LoopCounter';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_ShapeParameters.FunctionInstance='Get
ShapeParameters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_ShapeParameters.OutputArg='ShapeParam
eters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_Centroids.FunctionInstance='GetShapeP
arameters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_Centroids.OutputArg='Centroids';
ifisemptypreviouscellslabel.FunctionArgs.SaveCellsLabel_CellsLabel.FunctionInstance='SaveCellsLab
el';
ifisemptypreviouscellslabel.FunctionArgs.SaveCellsLabel_CellsLabel.OutputArg='CellsLabel';
ifisemptypreviouscellslabel.FunctionArgs.HoldMatchingGroups_ValueToHold.FunctionInstance='HoldMat
chingGroups';
ifisemptypreviouscellslabel.FunctionArgs.HoldMatchingGroups_ValueToHold.OutputArg='ValueToHold';
```

```
ifisemptypreviouscellslabel.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='Segmenta
tionLoop';
ifisemptypreviouscellslabel.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout
_TracksLayout';
ifisemptypreviouscellslabel.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.FunctionInstance=
'AssignCellsToTracksLoop';
ifisemptypreviouscellslabel.KeepValues.AssignCellToTrackUsingAll_MatchingGroups.OutputArg='Assign
CellToTrackUsingAll_MatchingGroups';
ifisemptypreviouscellslabel.KeepValues.ContinueTracks_Tracks.FunctionInstance='ContinueTracks';
ifisemptypreviouscellslabel.KeepValues.ContinueTracks_Tracks.OutputArg='Tracks';
ifisemptypreviouscellslabel.ElseFunctions=else_is_empty_cells_label_functions;
ifisemptypreviouscellslabel.IfFunctions=if_is_empty_cells_label_functions;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,ifisemptypreviouscellslabe
l);

savecellslabel.InstanceName='SaveCellsLabel';
savecellslabel.FunctionHandle=@saveCellsLabel;
savecellslabel.FunctionArgs.FileRoot.Value=SegmentationFilesRoot;
savecellslabel.FunctionArgs.NumberFormat.Value=NumberFormat;
savecellslabel.FunctionArgs.CellsLabel.FunctionInstance='ResizeCytoLabel';
savecellslabel.FunctionArgs.CellsLabel.OutputArg='Image';
savecellslabel.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
savecellslabel.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,savecellslabel);

holdmatchinggroups.InstanceName='HoldMatchingGroups';
holdmatchinggroups.FunctionHandle=@holdValue;
holdmatchinggroups.FunctionArgs.ValueToHold.Value=[];
holdmatchinggroups.FunctionArgs.ValueToHold.FunctionInstance='IfIsEmptyPreviousCellsLabel';
holdmatchinggroups.FunctionArgs.ValueToHold.OutputArg='AssignCellToTrackUsingAll_MatchingGroups';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,holdmatchinggroups);

segmentationloop.InstanceName='SegmentationLoop';
segmentationloop.FunctionHandle=@forLoop;
segmentationloop.FunctionArgs.StartLoop.Value=StartFrame;
segmentationloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
segmentationloop.FunctionArgs.IncrementLoop.Value=FrameStep;
segmentationloop.FunctionArgs.MatchingGroups.Value=[];
segmentationloop.FunctionArgs.Tracks.Value=[];
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='LoadTracksLayout';
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.OutputArg='TracksLayout';
segmentationloop.KeepValues.ContinueTracks_Tracks.FunctionInstance='IfIsEmptyPreviousCellsLabel';
segmentationloop.KeepValues.ContinueTracks_Tracks.OutputArg='ContinueTracks_Tracks';
segmentationloop.LoopFunctions=image_read_loop_functions;
functions_list=addToFunctionChain(functions_list,segmentationloop);

savetracks.InstanceName='SaveTracks';
savetracks.FunctionHandle=@saveTracks;
savetracks.FunctionArgs.TracksFileName.Value=[TracksFolder '/tracks.mat'];
savetracks.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
savetracks.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,savetracks);

gettrackids.InstanceName='GetTrackIDs';
gettrackids.FunctionHandle=@getTrackIDs;
gettrackids.FunctionArgs.TrackIDCol.Value=1;
gettrackids.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
gettrackids.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,gettrackids);

detectmergecandidates.InstanceName='DetectMergeCandidates';
detectmergecandidates.FunctionHandle=@detectMergeCandidatesUsingDistance;
detectmergecandidates.FunctionArgs.MaxMergeDistance.Value=MaxMergeDistance;
detectmergecandidates.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDs';
detectmergecandidates.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
detectmergecandidates.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
detectmergecandidates.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
detectmergecandidates.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
detectmergecandidates.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,detectmergecandidates);

mergetracks.InstanceName='MergeTracks';
mergetracks.FunctionHandle=@mergeTracks;
mergetracks.FunctionArgs.FrameCount.Value=FrameCount;
mergetracks.FunctionArgs.StartFrame.Value=StartFrame;
mergetracks.FunctionArgs.TimeFrame.Value=TimeFrame;
mergetracks.FunctionArgs.SegFileRoot.Value=SegmentationFilesRoot;
mergetracks.FunctionArgs.FrameStep.Value=FrameStep;
mergetracks.FunctionArgs.NumberFormat.Value=NumberFormat;
mergetracks.FunctionArgs.TracksToBeMerged.FunctionInstance='DetectMergeCandidates';
mergetracks.FunctionArgs.TracksToBeMerged.OutputArg='TracksToBeMerged';
```

```
mergetracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
mergetracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
mergetracks.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
mergetracks.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,mergetracks);

gettrackidsaftermerge.InstanceName='GetTrackIDsAfterMerge';
gettrackidsaftermerge.FunctionHandle=@getTrackIDs;
gettrackidsaftermerge.FunctionArgs.TrackIDCol.Value=1;
gettrackidsaftermerge.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
gettrackidsaftermerge.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,gettrackidsaftermerge);

makeancestryforfirstframecells.InstanceName='MakeAncestryForFirstFrameCells';
makeancestryforfirstframecells.FunctionHandle=@makeAncestryForFirstFrameCells;
makeancestryforfirstframecells.FunctionArgs.TimeCol.Value=2;
makeancestryforfirstframecells.FunctionArgs.TrackIDCol.Value=1;
makeancestryforfirstframecells.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
makeancestryforfirstframecells.FunctionArgs.Tracks.OutputArg='Tracks';
makeancestryforfirstframecells.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDsAfterMerge';
makeancestryforfirstframecells.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
functions_list=addToFunctionChain(functions_list,makeancestryforfirstframecells);

detectmitoticevents.InstanceName='DetectMitoticEvents';
detectmitoticevents.FunctionHandle=@detectMitoticEvents;
detectmitoticevents.FunctionArgs.MaxSplitArea.Value=MaxSplitArea;
detectmitoticevents.FunctionArgs.MinSplitEccentricity.Value=MinSplitEccentricity;
detectmitoticevents.FunctionArgs.MaxSplitEccentricity.Value=MaxSplitEccentricity;
detectmitoticevents.FunctionArgs.MaxSplitDistance.Value=MaxSplitDistance;
detectmitoticevents.FunctionArgs.MinTimeForSplit.Value=MinTimeForSplit;
detectmitoticevents.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
detectmitoticevents.FunctionArgs.Tracks.OutputArg='Tracks';
detectmitoticevents.FunctionArgs.UntestedIDs.FunctionInstance='MakeAncestryForFirstFrameCells';
detectmitoticevents.FunctionArgs.UntestedIDs.OutputArg='UntestedIDs';
detectmitoticevents.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
detectmitoticevents.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
functions_list=addToFunctionChain(functions_list,detectmitoticevents);

makeancestryforcellsenteringframes.InstanceName='MakeAncestryForCellsEnteringFrames';
makeancestryforcellsenteringframes.FunctionHandle=@makeAncestryForCellsEnteringFrames;
makeancestryforcellsenteringframes.FunctionArgs.TimeCol.Value=2;
makeancestryforcellsenteringframes.FunctionArgs.TrackIDCol.Value=1;
makeancestryforcellsenteringframes.FunctionArgs.SplitCells.FunctionInstance='DetectMitoticEvents'
;
makeancestryforcellsenteringframes.FunctionArgs.SplitCells.OutputArg='SplitCells';
makeancestryforcellsenteringframes.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDsAfterMerge'
;
makeancestryforcellsenteringframes.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
makeancestryforcellsenteringframes.FunctionArgs.FirstFrameIDs.FunctionInstance='MakeAncestryForFi
rstFrameCells';
makeancestryforcellsenteringframes.FunctionArgs.FirstFrameIDs.OutputArg='FirstFrameIDs';
makeancestryforcellsenteringframes.FunctionArgs.CellsAncestry.FunctionInstance='MakeAncestryForFi
rstFrameCells';
makeancestryforcellsenteringframes.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
makeancestryforcellsenteringframes.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
makeancestryforcellsenteringframes.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,makeancestryforcellsenteringframes);

splittracks.InstanceName='SplitTracks';
splittracks.FunctionHandle=@splitTracks;
splittracks.FunctionArgs.TimeFrame.Value=TimeFrame;
splittracks.FunctionArgs.SplitCells.FunctionInstance='DetectMitoticEvents';
splittracks.FunctionArgs.SplitCells.OutputArg='SplitCells';
splittracks.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
splittracks.FunctionArgs.Tracks.OutputArg='Tracks';
splittracks.FunctionArgs.CellsAncestry.FunctionInstance='MakeAncestryForCellsEnteringFrames';
splittracks.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
splittracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
splittracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
splittracks.FunctionArgs.AncestryLayout.FunctionInstance='LoadAncestryLayout';
splittracks.FunctionArgs.AncestryLayout.OutputArg='AncestryLayout';
functions_list=addToFunctionChain(functions_list,splittracks);

removeshorttracks.InstanceName='RemoveShortTracks';
removeshorttracks.FunctionHandle=@removeShortTracks;
removeshorttracks.FunctionArgs.MinLifespan.Value=MinLifespan;
removeshorttracks.FunctionArgs.Tracks.FunctionInstance='SplitTracks';
removeshorttracks.FunctionArgs.Tracks.OutputArg='Tracks';
removeshorttracks.FunctionArgs.CellsAncestry.FunctionInstance='SplitTracks';
removeshorttracks.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
removeshorttracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
```

```
removeshorttracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
removeshorttracks.FunctionArgs.AncestryLayout.FunctionInstance='LoadAncestryLayout';
removeshorttracks.FunctionArgs.AncestryLayout.OutputArg='AncestryLayout';
functions_list=addToFunctionChain(functions_list,removeshorttracks);

saveupdatedtracks.InstanceName='SaveUpdatedTracks';
saveupdatedtracks.FunctionHandle=@saveTracks;
saveupdatedtracks.FunctionArgs.TracksFileName.Value=[AncestryFolder '/tracks.mat'];
saveupdatedtracks.FunctionArgs.Tracks.FunctionInstance='RemoveShortTracks';
saveupdatedtracks.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,saveupdatedtracks);

saveancestry.InstanceName='SaveAncestry';
saveancestry.FunctionHandle=@saveAncestry;
saveancestry.FunctionArgs.AncestryFileName.Value=[AncestryFolder '/ancestry.mat'];
saveancestry.FunctionArgs.CellsAncestry.FunctionInstance='RemoveShortTracks';
saveancestry.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
functions_list=addToFunctionChain(functions_list,saveancestry);

makeimagenamesinoverlayloop.InstanceName='MakeImageNamesInOverlayLoop';
makeimagenamesinoverlayloop.FunctionHandle=@makeImgFileName;
makeimagenamesinoverlayloop.FunctionArgs.FileBase.Value=ImageFileBase;
makeimagenamesinoverlayloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinoverlayloop.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenamesinoverlayloop.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
makeimagenamesinoverlayloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,makeimagenamesinover
layloop);

readimagesinoverlayloop.InstanceName='ReadImagesInOverlayLoop';
readimagesinoverlayloop.FunctionHandle=@readImage;
readimagesinoverlayloop.FunctionArgs.ImageChannel.Value='';
readimagesinoverlayloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInOverlayLoop';
readimagesinoverlayloop.FunctionArgs.ImageName.OutputArg='FileName';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,readimagesinoverlayl
oop);

getcurrenttracks2.InstanceName='GetCurrentTracks2';
getcurrenttracks2.FunctionHandle=@getCurrentTracks;
getcurrenttracks2.FunctionArgs.OffsetFrame.Value=0;
getcurrenttracks2.FunctionArgs.TimeFrame.Value=TimeFrame;
getcurrenttracks2.FunctionArgs.TimeCol.Value=2;
getcurrenttracks2.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks2.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getcurrenttracks2.FunctionArgs.FrameStep.Value=FrameStep;
getcurrenttracks2.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
getcurrenttracks2.FunctionArgs.CurFrame.OutputArg='LoopCounter';
getcurrenttracks2.FunctionArgs.Tracks.FunctionInstance='ImageOverlayLoop';
getcurrenttracks2.FunctionArgs.Tracks.InputArg='RemoveShortTracks_Tracks';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,getcurrenttracks2);

makematnamesinoverlayloop.InstanceName='MakeMatNamesInOverlayLoop';
makematnamesinoverlayloop.FunctionHandle=@makeImgFileName;
makematnamesinoverlayloop.FunctionArgs.FileBase.Value=SegmentationFilesRoot;
makematnamesinoverlayloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makematnamesinoverlayloop.FunctionArgs.FileExt.Value='.mat';
makematnamesinoverlayloop.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
makematnamesinoverlayloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,makematnamesinoverla
yloop);

loadcellslabel.InstanceName='LoadCellsLabel';
loadcellslabel.FunctionHandle=@loadCellsLabel;
loadcellslabel.FunctionArgs.FileName.FunctionInstance='MakeMatNamesInOverlayLoop';
loadcellslabel.FunctionArgs.FileName.OutputArg='FileName';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,loadcellslabel);

loadcolormap.InstanceName='LoadColormap';
loadcolormap.FunctionHandle=@loadColormap;
loadcolormap.FunctionArgs.FileName.Value='colormap_lines';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,loadcolormap);

displayoverlayingframe.InstanceName='DisplayOverlayingFrame';
displayoverlayingframe.FunctionHandle=@displayVariable;
displayoverlayingframe.FunctionArgs.VariableName.Value='Overlaying Frame';
displayoverlayingframe.FunctionArgs.Variable.FunctionInstance='ImageOverlayLoop';
displayoverlayingframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,displayoverlayingfra
me);

displayancestry.InstanceName='DisplayAncestry';
```

```
displayancestry.FunctionHandle=@displayAncestryData;
displayancestry.FunctionArgs.NumberFormat.Value=NumberFormat;
displayancestry.FunctionArgs.ProlDir.Value=AncestryFolder;
displayancestry.FunctionArgs.ImageFileName.Value=ImageFilesRoot;
displayancestry.FunctionArgs.DS.Value='/';
displayancestry.FunctionArgs.Image.FunctionInstance='ReadImagesInOverlayLoop';
displayancestry.FunctionArgs.Image.OutputArg='Image';
displayancestry.FunctionArgs.CurrentTracks.FunctionInstance='GetCurrentTracks2';
displayancestry.FunctionArgs.CurrentTracks.OutputArg='Tracks';
displayancestry.FunctionArgs.CellsLabel.FunctionInstance='LoadCellsLabel';
displayancestry.FunctionArgs.CellsLabel.OutputArg='LabelMatrix';
displayancestry.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.CurFrame.OutputArg='LoopCounter';
displayancestry.FunctionArgs.ColorMap.FunctionInstance='LoadColormap';
displayancestry.FunctionArgs.ColorMap.OutputArg='Colormap';
displayancestry.FunctionArgs.CellsAncestry.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.CellsAncestry.InputArg='RemoveShortTracks_CellsAncestry';
displayancestry.FunctionArgs.TracksLayout.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
displayancestry.FunctionArgs.AncestryLayout.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.AncestryLayout.InputArg='LoadAncestryLayout_AncestryLayout';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,displayancestry);

imageoverlayloop.InstanceName='ImageOverlayLoop';
imageoverlayloop.FunctionHandle=@forLoop;
imageoverlayloop.FunctionArgs.StartLoop.Value=StartFrame;
imageoverlayloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
imageoverlayloop.FunctionArgs.IncrementLoop.Value=FrameStep;
imageoverlayloop.FunctionArgs.RemoveShortTracks_Tracks.FunctionInstance='RemoveShortTracks';
imageoverlayloop.FunctionArgs.RemoveShortTracks_Tracks.OutputArg='Tracks';
imageoverlayloop.FunctionArgs.RemoveShortTracks_CellsAncestry.FunctionInstance='RemoveShortTracks
';
imageoverlayloop.FunctionArgs.RemoveShortTracks_CellsAncestry.OutputArg='CellsAncestry';
imageoverlayloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='LoadTracksLayout';
imageoverlayloop.FunctionArgs.LoadTracksLayout_TracksLayout.OutputArg='TracksLayout';
imageoverlayloop.FunctionArgs.LoadAncestryLayout_AncestryLayout.FunctionInstance='LoadAncestryLay
out';
imageoverlayloop.FunctionArgs.LoadAncestryLayout_AncestryLayout.OutputArg='AncestryLayout';
imageoverlayloop.LoopFunctions=image_overlay_loop_functions;
functions_list=addToFunctionChain(functions_list,imageoverlayloop);

saveancestryspreadsheets.InstanceName='SaveAncestrySpreadsheets';
saveancestryspreadsheets.FunctionHandle=@saveAncestrySpreadsheets;
saveancestryspreadsheets.FunctionArgs.ShapesXlsFile.Value=ShapesSpreadsheet;
saveancestryspreadsheets.FunctionArgs.ProlXlsFile.Value=AncestrySpreadsheet;
saveancestryspreadsheets.FunctionArgs.Tracks.FunctionInstance='RemoveShortTracks';
saveancestryspreadsheets.FunctionArgs.Tracks.OutputArg='Tracks';
saveancestryspreadsheets.FunctionArgs.CellsAncestry.FunctionInstance='RemoveShortTracks';
saveancestryspreadsheets.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
saveancestryspreadsheets.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
saveancestryspreadsheets.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
functions_list=addToFunctionChain(functions_list,saveancestryspreadsheets);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayFluoNuclTestNN.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayFluoNuclTestNN()
%assayFluoNuclTestNN - This module is used to track cells that have been stained using a
% nuclear stain  using a nearest-neighbor algorithm. The assay tracks the cells, detects
%mitotic  events and records  tracking data, cell shape parameters and ancestry information
%to spreadsheets.  For each tracked image,  an overlayed image is saved to
%disk that displays  the detected cell outlines, cell  IDs and cell generation.
%ImageFolder - String variable  that specifies the absolute location of the directory which
%contains the  time-lapse images.  An example of such a string variable would
%be 'c:/sample images/high-density'. ImageFilesRoot - String  variable specifying the root image
file name.
%The root image file name  for  a set of images is the
%image file name of any of the   images without the number or
%the file extension. For example, if the file name   is 'Experiment-0002_Position(8)_t021.tif'
the
```

```
%root image file name will be 'Experiment-0002_Position(8)_t'. ImageExtension - String  variable
specifying the
%image file extension including the preceding dot. For example  if  the file
%name is 'image003.jpg' the image extension is '.jpg'. StartFrame - Number specifying  the
%first image in the sequence to be analyzed. The minimum  value for
%this variable depends on the numbering of the image sequence so if  the
% first image in the sequence is 'image003.tif' then the minimum value is 3.
%FrameCount  - Number specifying how many images from the image sequence should be
%processed. TimeFrame  - Number specifying the time between consecutive images in minutes.
FrameStep
%- Number specifying  the step size when reading images. Set this variable to
%1  to read  every image in the sequence, 2 to read every
%other image and  so  on. NumberFormat - String value specifying the number
%of digits in the image file  names in  the sequence. For example
%if the image file name is 'image020.jpg'  the value for  the NumberFormat
%is '%03d', while if the file name is  'image000020.jpg' the value should
%be '%06d'. MaxFramesMissing - Number specifying for how many  frames a cell may
%be disappear before its  track is ended. OutputFolder -  The folder where
%the overlayed images and track data will be saved. By   default this
%value is set to a folder named 'output' within the folder where
%the images to be analyzed are located. AncestryFolder - The folder where the
%overlayed images and ancestry data will be saved. By  default this value is
% set to a folder named 'ancestry' within the output folder. AncestrySpreadsheet - The
%path  name to the spreadsheet containing the ancestry data. By default this
%value is  set to a file named 'ancestry.csv' within the ancestry folder. ShapesSpreadsheet
%- The path  name to the spreadsheet containing the position and shape properties
%for  each cell  in the timelapse sequence at every time point. By
%default this is  set  to to a file named 'shapes.csv' within the
%ancestry folder. TracksFolder - The folder  where the label matrixes containing the cell
%outlines are saved. By  default this  value is set to a folder
%named 'track' within the output folder. SegmentationFilesRoot -  The root file name of
%the label matrixes containing the cell outlines. ImageFileBase -  The path name to
%the images. This value is generated from the ImageFolder   and the ImageFilesRoot
%and should not be changed. BrightnessThresholdPct - Number specifying the percentage  threshold
value
%for the image generated by the generateBinImgUsingLocAvg  filter. Any pixel in  the
%original image smaller than the threshold value times the  corresponding value in
%the local average image below this value will be set to  zero while
% the rest will be set to one. ObjectArea - Number specifying the threshold
%area  for the areaFilterLabel, clearSmallObjects, segmentObjectsUsingClusters filters. Objects
below  this value will
%be removed  from the filtered image. Strel - String variable specifying the type
%of filter used  to generate the local average  image in generateBinImgUsingLocAvg. Currently
%'disk' is the only  value supported. StrelSize - Number specifying the size of
%the local neighborhood used to  calculate the average  for each pixel in
%the local average image generated by  the generateBinImgUsingLocAvg module. ClearBorder -
Boolean value
%specifying whether objects next to or touching  the image border in  the
%binary images generated by the generateBinImgUsingGradient module will  be erased (true) or not
% (false). ClearBorderDist - Number specifying how close to  the border objects may
%be and still be  erased if the ClearBorder parameter  is set to
%true in the generateBinImgUsingGradient module. MedianFilterSize - Number specifying the size
of the
%median filter used by the distanceWatershed module.  Setting this to a  higher
%integer value will reduce the number of objects detected  by the module
%and can be used to prevent oversegmentation. MinSolidity - Number specifying a threshold
solidity
% value for the solidityFilterLabelObjects filter. Objects whose solidity  is below this value
%will  be removed from the filtered image. MinAreaOverPerimeter - Number specifying a threshold
%AOP ratio  value for the areaOverPerimeterFilterLabel filter. Objects with  an AOP ratio
%smaller than this  value will be removed. ResizeImageScale - Number specifying by what
%ratio the images will  be resized before they are  processed. By default
%this value is set to  0.5. Used by the resizeImage module. ApproximationDistance -
%Number specifying how close the convex  hull in the getConvexObjects module approximates the
% object outline. By default this value  is set to 2.5. Setting it
%to a  lower value will result in  convex hulls that more closely
%resemble the object outlines  however this increases the  chance of detecting insignificant
%concavities. MaxMergeDistance - Number specifying the maximum distance that one  track may be
%from another track  for the duration and still be considered  for possible
%merging with the other track.  Used by detectMergeCandidatesUsingDistance module. MaxSplitArea -
Number
%specifying the maximum area a nucleus may be and still be considered
%as a part of a possible mitotic event. Used by detectMitoticEvents module. MaxSplitDistance -
% Number specifying the maximum distance a new nucleus may be from another nucleus
%  and still be considered as part of a possible mitotic event. Used
%by detectMitoticEvents   module. MinSplitEccentricity - Number specifying the minimum
eccentricity a new
%nucleus may have  and still be  considered as part of a possible
%mitotic event. Used by  detectMitoticEvents module. MaxSplitEccentricity - Number specifying the
maximum eccentricity
%a new nucleus may have  and still be  considered as part of
```

```
%a possible mitotic event. Used by  detectMitoticEvents module. MinTimeForSplit - Number
specifying the
%minimum time in minutes a track needs  to exist before  it is
%considered for a possible mitotic event. Used by  detectMitoticEvents module. MinLifespan -
Number
%specifying the minimum length in frames a frame has  to be to
%not be removed by the removeShortTracks module. Important Modules -   areaFilterLabel,
areaOverPerimeterFilterLabel,
%assignCellToTrackUsingNN, clearSmallObjects, detectMergeCandidatesUsingDistance,
detectMitoticEvents, distanceWatershed, generateBinImgUsingLocAvg, getConvexObjects,
polygonalAssistedWatershed, removeShortTracks, segmentObjectsUsingMarkers, solidityFilterLabel,
splitTracks.

global functions_list;
functions_list=[];
%script variables
ImageFolder='C:/sample movies/low density';
ImageFilesRoot='low density sample';
ImageExtension='.tif';
StartFrame=1;
FrameCount=10;
TimeFrame=15;
FrameStep=1;
NumberFormat='%06d';
MaxFramesMissing=6;
OutputFolder=[ImageFolder '/output'];
AncestryFolder=[OutputFolder '/ancestry'];
AncestrySpreadsheet=[AncestryFolder 'ancestry.csv'];
ShapesSpreadsheet=[AncestryFolder 'shapes.csv'];
TracksFolder=[OutputFolder '/track'];
SegmentationFilesRoot=[TracksFolder '/grayscale'];
ImageFileBase=[ImageFolder '/' ImageFilesRoot];
BrightnessThresholdPct=1.1;
ObjectArea=30;
Strel='disk';
StrelSize=10;
ClearBorder=true;
ClearBorderDist=2;
MedianFilterSize=3;
MinSolidity=0.69;
MinAreaOverPerimeter=1.5;
ResizeImageScale=0.5;
ApproximationDistance=2.4;
MaxMergeDistance=23;
MaxSplitArea=400;
MaxSplitDistance=45;
MinSplitEccentricity=0.5;
MaxSplitEccentricity=0.95;
MinTimeForSplit=900;
MinLifespan=30;
%end script variables

assign_cells_to_tracks_functions=[];
else_is_empty_cells_label_functions=[];
if_is_empty_cells_label_functions=[];
image_read_loop_functions=[];
image_overlay_loop_functions=[];

loadtrackslayout.InstanceName='LoadTracksLayout';
loadtrackslayout.FunctionHandle=@loadTracksLayout;
loadtrackslayout.FunctionArgs.FileName.Value='tracks_layout.mat';
functions_list=addToFunctionChain(functions_list,loadtrackslayout);

loadancestrylayout.InstanceName='LoadAncestryLayout';
loadancestrylayout.FunctionHandle=@loadAncestryLayout;
loadancestrylayout.FunctionArgs.FileName.Value='ancestry_layout.mat';
functions_list=addToFunctionChain(functions_list,loadancestrylayout);

displaycurframe.InstanceName='DisplayCurFrame';
displaycurframe.FunctionHandle=@displayVariable;
displaycurframe.FunctionArgs.VariableName.Value='Current Tracking Frame';
displaycurframe.FunctionArgs.Variable.FunctionInstance='SegmentationLoop';
displaycurframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,displaycurframe);

makeimagenamesinsegmentationloop.InstanceName='MakeImageNamesInSegmentationLoop';
makeimagenamesinsegmentationloop.FunctionHandle=@makeImgFileName;
makeimagenamesinsegmentationloop.FunctionArgs.FileBase.Value=ImageFileBase;
makeimagenamesinsegmentationloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinsegmentationloop.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
```

```
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makeimagenamesinsegmentati
onloop);

readimagesinsegmentationloop.InstanceName='ReadImagesInSegmentationLoop';
readimagesinsegmentationloop.FunctionHandle=@readImage;
readimagesinsegmentationloop.FunctionArgs.ImageChannel.Value='';
readimagesinsegmentationloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInSegmentatio
nLoop';
readimagesinsegmentationloop.FunctionArgs.ImageName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,readimagesinsegmentationlo
op);

normalizeimageto16bit.InstanceName='NormalizeImageTo16Bit';
normalizeimageto16bit.FunctionHandle=@imNorm;
normalizeimageto16bit.FunctionArgs.IntegerClass.Value='uint16';
normalizeimageto16bit.FunctionArgs.RawImage.FunctionInstance='ReadImagesInSegmentationLoop';
normalizeimageto16bit.FunctionArgs.RawImage.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,normalizeimageto16bit);

resizeimage.InstanceName='ResizeImage';
resizeimage.FunctionHandle=@resizeImage;
resizeimage.FunctionArgs.Scale.Value=ResizeImageScale;
resizeimage.FunctionArgs.Method.Value='bicubic';
resizeimage.FunctionArgs.Image.FunctionInstance='NormalizeImageTo16Bit';
resizeimage.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,resizeimage);

cytobrightnesslocalaveragingfilter.InstanceName='CytoBrightnessLocalAveragingFilter';
cytobrightnesslocalaveragingfilter.FunctionHandle=@generateBinImgUsingLocAvg;
cytobrightnesslocalaveragingfilter.FunctionArgs.Strel.Value=Strel;
cytobrightnesslocalaveragingfilter.FunctionArgs.StrelSize.Value=StrelSize;
cytobrightnesslocalaveragingfilter.FunctionArgs.BrightnessThresholdPct.Value=BrightnessThresholdP
ct;
cytobrightnesslocalaveragingfilter.FunctionArgs.ClearBorder.Value=ClearBorder;
cytobrightnesslocalaveragingfilter.FunctionArgs.ClearBorderDist.Value=ClearBorderDist;
cytobrightnesslocalaveragingfilter.FunctionArgs.Image.FunctionInstance='ResizeImage';
cytobrightnesslocalaveragingfilter.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,cytobrightnesslocalaveragi
ngfilter);

fillholescytoplasmimages.InstanceName='FillHolesCytoplasmImages';
fillholescytoplasmimages.FunctionHandle=@fillHoles;
fillholescytoplasmimages.FunctionArgs.Image.FunctionInstance='CytoBrightnessLocalAveragingFilter'
;
fillholescytoplasmimages.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,fillholescytoplasmimages);

clearsmallcells.InstanceName='ClearSmallCells';
clearsmallcells.FunctionHandle=@clearSmallObjects;
clearsmallcells.FunctionArgs.MinObjectArea.Value=ObjectArea;
clearsmallcells.FunctionArgs.Image.FunctionInstance='FillHolesCytoplasmImages';
clearsmallcells.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,clearsmallcells);

nuclbrightnesslocalaveragingfilter.InstanceName='NuclBrightnessLocalAveragingFilter';
nuclbrightnesslocalaveragingfilter.FunctionHandle=@generateBinImgUsingLocAvg;
nuclbrightnesslocalaveragingfilter.FunctionArgs.Strel.Value=Strel;
nuclbrightnesslocalaveragingfilter.FunctionArgs.StrelSize.Value=StrelSize;
nuclbrightnesslocalaveragingfilter.FunctionArgs.BrightnessThresholdPct.Value=BrightnessThresholdP
ct;
nuclbrightnesslocalaveragingfilter.FunctionArgs.ClearBorder.Value=ClearBorder;
nuclbrightnesslocalaveragingfilter.FunctionArgs.ClearBorderDist.Value=ClearBorderDist;
nuclbrightnesslocalaveragingfilter.FunctionArgs.Image.FunctionInstance='ResizeImage';
nuclbrightnesslocalaveragingfilter.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,nuclbrightnesslocalaveragi
ngfilter);

fillholesnuclearimages.InstanceName='FillHolesNuclearImages';
fillholesnuclearimages.FunctionHandle=@fillHoles;
fillholesnuclearimages.FunctionArgs.Image.FunctionInstance='NuclBrightnessLocalAveragingFilter';
fillholesnuclearimages.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,fillholesnuclearimages);

clearsmallnuclei.InstanceName='ClearSmallNuclei';
clearsmallnuclei.FunctionHandle=@clearSmallObjects;
clearsmallnuclei.FunctionArgs.MinObjectArea.Value=ObjectArea;
clearsmallnuclei.FunctionArgs.Image.FunctionInstance='FillHolesNuclearImages';
clearsmallnuclei.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,clearsmallnuclei);
```

```
combinenuclearandcytoplasmimages.InstanceName='CombineNuclearAndCytoplasmImages';
combinenuclearandcytoplasmimages.FunctionHandle=@combineImages;
combinenuclearandcytoplasmimages.FunctionArgs.CombineOperation.Value='AND';
combinenuclearandcytoplasmimages.FunctionArgs.Image1.FunctionInstance='ClearSmallNuclei';
combinenuclearandcytoplasmimages.FunctionArgs.Image1.OutputArg='Image';
combinenuclearandcytoplasmimages.FunctionArgs.Image2.FunctionInstance='ClearSmallCells';
combinenuclearandcytoplasmimages.FunctionArgs.Image2.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,combinenuclearandcytoplasm
images);

reconstructcytoplasmimage.InstanceName='ReconstructCytoplasmImage';
reconstructcytoplasmimage.FunctionHandle=@reconstructObjects;
reconstructcytoplasmimage.FunctionArgs.GuideImage.FunctionInstance='CombineNuclearAndCytoplasmIma
ges';
reconstructcytoplasmimage.FunctionArgs.GuideImage.OutputArg='Image';
reconstructcytoplasmimage.FunctionArgs.ImageToReconstruct.FunctionInstance='ClearSmallNuclei';
reconstructcytoplasmimage.FunctionArgs.ImageToReconstruct.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,reconstructcytoplasmimage)
;

labelnuclei.InstanceName='LabelNuclei';
labelnuclei.FunctionHandle=@labelObjects;
labelnuclei.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
labelnuclei.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,labelnuclei);

labelcytoplasm.InstanceName='LabelCytoplasm';
labelcytoplasm.FunctionHandle=@labelObjects;
labelcytoplasm.FunctionArgs.Image.FunctionInstance='ReconstructCytoplasmImage';
labelcytoplasm.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,labelcytoplasm);

getconvexobjects.InstanceName='GetConvexObjects';
getconvexobjects.FunctionHandle=@getConvexObjects;
getconvexobjects.FunctionArgs.ApproximationDistance.Value=ApproximationDistance;
getconvexobjects.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
getconvexobjects.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getconvexobjects);

distancewatershed.InstanceName='DistanceWatershed';
distancewatershed.FunctionHandle=@distanceWatershed;
distancewatershed.FunctionArgs.MedianFilterNhood.Value=MedianFilterSize;
distancewatershed.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
distancewatershed.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,distancewatershed);

polygonalassistedwatershed.InstanceName='PolygonalAssistedWatershed';
polygonalassistedwatershed.FunctionHandle=@polygonalAssistedWatershed;
polygonalassistedwatershed.FunctionArgs.MinBlobArea.Value=ObjectArea;
polygonalassistedwatershed.FunctionArgs.ImageLabel.FunctionInstance='LabelNuclei';
polygonalassistedwatershed.FunctionArgs.ImageLabel.OutputArg='LabelMatrix';
polygonalassistedwatershed.FunctionArgs.WatershedLabel.FunctionInstance='DistanceWatershed';
polygonalassistedwatershed.FunctionArgs.WatershedLabel.OutputArg='LabelMatrix';
polygonalassistedwatershed.FunctionArgs.ConvexObjectsIndex.FunctionInstance='GetConvexObjects';
polygonalassistedwatershed.FunctionArgs.ConvexObjectsIndex.OutputArg='ConvexObjectsIndex';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,polygonalassistedwatershed
);

segmentobjectsusingmarkers.InstanceName='SegmentObjectsUsingMarkers';
segmentobjectsusingmarkers.FunctionHandle=@segmentObjectsUsingMarkers;
segmentobjectsusingmarkers.FunctionArgs.MarkersLabel.FunctionInstance='PolygonalAssistedWatershed
';
segmentobjectsusingmarkers.FunctionArgs.MarkersLabel.OutputArg='LabelMatrix';
segmentobjectsusingmarkers.FunctionArgs.ObjectsLabel.FunctionInstance='LabelCytoplasm';
segmentobjectsusingmarkers.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,segmentobjectsusingmarkers
);

areafilter.InstanceName='AreaFilter';
areafilter.FunctionHandle=@areaFilterLabel;
areafilter.FunctionArgs.MinArea.Value=ObjectArea;
areafilter.FunctionArgs.ObjectsLabel.FunctionInstance='SegmentObjectsUsingMarkers';
areafilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,areafilter);

solidityfilter.InstanceName='SolidityFilter';
solidityfilter.FunctionHandle=@solidityFilterLabel;
solidityfilter.FunctionArgs.MinSolidity.Value=MinSolidity;
solidityfilter.FunctionArgs.ObjectsLabel.FunctionInstance='AreaFilter';
solidityfilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,solidityfilter);
```

```
aoverpfilter.InstanceName='AOverPFilter';
aoverpfilter.FunctionHandle=@areaOverPerimeterFilterLabel;
aoverpfilter.FunctionArgs.MinAreaOverPerimeter.Value=MinAreaOverPerimeter;
aoverpfilter.FunctionArgs.ObjectsLabel.FunctionInstance='SolidityFilter';
aoverpfilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,aoverpfilter);

resizecytolabel.InstanceName='ResizeCytoLabel';
resizecytolabel.FunctionHandle=@resizeImage;
resizecytolabel.FunctionArgs.Scale.Value=1/ResizeImageScale;
resizecytolabel.FunctionArgs.Method.Value='nearest';
resizecytolabel.FunctionArgs.Image.FunctionInstance='AOverPFilter';
resizecytolabel.FunctionArgs.Image.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,resizecytolabel);

getshapeparameters.InstanceName='GetShapeParameters';
getshapeparameters.FunctionHandle=@getShapeParams;
getshapeparameters.FunctionArgs.LabelMatrix.FunctionInstance='ResizeCytoLabel';
getshapeparameters.FunctionArgs.LabelMatrix.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getshapeparameters);

isemptypreviouscellslabel.InstanceName='IsEmptyPreviousCellsLabel';
isemptypreviouscellslabel.FunctionHandle=@isEmptyFunction;
isemptypreviouscellslabel.FunctionArgs.TestVariable.Value=[];
isemptypreviouscellslabel.FunctionArgs.TestVariable.FunctionInstance='SaveCellsLabel';
isemptypreviouscellslabel.FunctionArgs.TestVariable.OutputArg='CellsLabel';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,isemptypreviouscellslabel)
;

starttracks.InstanceName='StartTracks';
starttracks.FunctionHandle=@startTracks;
starttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
starttracks.FunctionArgs.CellsLabel.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.CellsLabel.InputArg='ResizeCytoLabel_Image';
starttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
starttracks.FunctionArgs.ShapeParameters.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.ShapeParameters.InputArg='GetShapeParameters_ShapeParameters';
if_is_empty_cells_label_functions=addToFunctionChain(if_is_empty_cells_label_functions,starttrack
s);

getcurrenttracks.InstanceName='GetCurrentTracks';
getcurrenttracks.FunctionHandle=@getCurrentTracks;
getcurrenttracks.FunctionArgs.OffsetFrame.Value=-1;
getcurrenttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getcurrenttracks.FunctionArgs.TimeCol.Value=2;
getcurrenttracks.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getcurrenttracks.FunctionArgs.FrameStep.Value=FrameStep;
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg='Tracks';
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getcurrenttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getcurrenttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getcur
renttracks);

getprevioustracks.InstanceName='GetPreviousTracks';
getprevioustracks.FunctionHandle=@getCurrentTracks;
getprevioustracks.FunctionArgs.OffsetFrame.Value=-2;
getprevioustracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getprevioustracks.FunctionArgs.TimeCol.Value=2;
getprevioustracks.FunctionArgs.TrackIDCol.Value=1;
getprevioustracks.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getprevioustracks.FunctionArgs.FrameStep.Value=FrameStep;
getprevioustracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg='Tracks';
getprevioustracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getprevioustracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getprevioustracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getpre
vioustracks);

makeunassignedcellslist.InstanceName='MakeUnassignedCellsList';
makeunassignedcellslist.FunctionHandle=@makeUnassignedCellsList;
makeunassignedcellslist.FunctionArgs.CellsCentroids.FunctionInstance='IfIsEmptyPreviousCellsLabel
';
makeunassignedcellslist.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
```

```
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,makeun
assignedcellslist);

getcellsmeandisplacement.InstanceName='GetCellsMeanDisplacement';
getcellsmeandisplacement.FunctionHandle=@getObjectsMeanDisplacement;
getcellsmeandisplacement.FunctionArgs.Centroid1Col.Value=3;
getcellsmeandisplacement.FunctionArgs.Centroid2Col.Value=4;
getcellsmeandisplacement.FunctionArgs.CurrentTracks.FunctionInstance='GetCurrentTracks';
getcellsmeandisplacement.FunctionArgs.CurrentTracks.OutputArg='Tracks';
getcellsmeandisplacement.FunctionArgs.ObjectCentroids.FunctionInstance='IfIsEmptyPreviousCellsLab
el';
getcellsmeandisplacement.FunctionArgs.ObjectCentroids.InputArg='GetShapeParameters_Centroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getcel
lsmeandisplacement);

getparamscoefficientofvariation.InstanceName='GetParamsCoefficientOfVariation';
getparamscoefficientofvariation.FunctionHandle=@getParamsCoefficientOfVariation;
getparamscoefficientofvariation.FunctionArgs.AreaCol.Value=5;
getparamscoefficientofvariation.FunctionArgs.SolidityCol.Value=11;
getparamscoefficientofvariation.FunctionArgs.Params.FunctionInstance='IfIsEmptyPreviousCellsLabel
';
getparamscoefficientofvariation.FunctionArgs.Params.InputArg='GetShapeParameters_ShapeParameters'
;
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getpar
amscoefficientofvariation);

getmaxtrackid.InstanceName='GetMaxTrackID';
getmaxtrackid.FunctionHandle=@getMaxTrackID;
getmaxtrackid.FunctionArgs.TrackIDCol.Value=1;
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg='Tracks';
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg2='Tracks';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getmax
trackid);

isnotemptyunassignedcells.InstanceName='IsNotEmptyUnassignedCells';
isnotemptyunassignedcells.FunctionHandle=@isNotEmptyFunction;
isnotemptyunassignedcells.FunctionArgs.TestVariable.FunctionInstance='AssignCellToTrackUsingNN';
isnotemptyunassignedcells.FunctionArgs.TestVariable.OutputArg='UnassignedIDs';
isnotemptyunassignedcells.FunctionArgs.TestVariable.FunctionInstance2='AssignCellsToTracksLoop';
isnotemptyunassignedcells.FunctionArgs.TestVariable.InputArg2='MakeUnassignedCellsList_Unassigned
CellsIDs';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,isnotemptyun
assignedcells);

getcurrentunassignedcell.InstanceName='GetCurrentUnassignedCell';
getcurrentunassignedcell.FunctionHandle=@getCurrentUnassignedCell;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance='AssignCellToTrackUsingNN'
;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTracksLoop'
;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassign
edCellsIDs';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,getcurrentun
assignedcell);

assigncelltotrackusingnn.InstanceName='AssignCellToTrackUsingNN';
assigncelltotrackusingnn.FunctionHandle=@assignCellToTrackUsingNN;
assigncelltotrackusingnn.FunctionArgs.TrackAssignments.Value=[];
assigncelltotrackusingnn.FunctionArgs.UnassignedCells.FunctionInstance='AssignCellToTrackUsingNN'
;
assigncelltotrackusingnn.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
assigncelltotrackusingnn.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellToTrackUsingNN
';
assigncelltotrackusingnn.FunctionArgs.TrackAssignments.OutputArg='TrackAssignments';
assigncelltotrackusingnn.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTracksLoop'
;
assigncelltotrackusingnn.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassign
edCellsIDs';
assigncelltotrackusingnn.FunctionArgs.CellsCentroids.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingnn.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
assigncelltotrackusingnn.FunctionArgs.CurrentTracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingnn.FunctionArgs.CurrentTracks.InputArg='GetCurrentTracks_Tracks';
assigncelltotrackusingnn.FunctionArgs.MaxTrackID.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingnn.FunctionArgs.MaxTrackID.InputArg='GetMaxTrackID_MaxTrackID';
assigncelltotrackusingnn.FunctionArgs.TracksLayout.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingnn.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,assigncellto
trackusingnn);
```

```
setmatchinggroupindex.InstanceName='SetMatchingGroupIndex';
setmatchinggroupindex.FunctionHandle=@setGroupIndex;
setmatchinggroupindex.FunctionArgs.AreaCol.Value=5;
setmatchinggroupindex.FunctionArgs.GroupIDCol.Value=13;
setmatchinggroupindex.FunctionArgs.CellID.FunctionInstance='GetCurrentUnassignedCell';
setmatchinggroupindex.FunctionArgs.CellID.OutputArg='CellID';
setmatchinggroupindex.FunctionArgs.GroupIndex.FunctionInstance='AssignCellToTrackUsingNN';
setmatchinggroupindex.FunctionArgs.GroupIndex.OutputArg='GroupIndex';
setmatchinggroupindex.FunctionArgs.ShapeParameters.FunctionInstance='SetMatchingGroupIndex';
setmatchinggroupindex.FunctionArgs.ShapeParameters.OutputArg='ShapeParameters';
setmatchinggroupindex.FunctionArgs.ShapeParameters.FunctionInstance2='AssignCellsToTracksLoop';
setmatchinggroupindex.FunctionArgs.ShapeParameters.InputArg2='GetShapeParameters_ShapeParameters'
;
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,setmatchingg
roupindex);

assigncellstotracksloop.InstanceName='AssignCellsToTracksLoop';
assigncellstotracksloop.FunctionHandle=@whileLoop;
assigncellstotracksloop.FunctionArgs.TestFunction.FunctionInstance='IsNotEmptyUnassignedCells';
assigncellstotracksloop.FunctionArgs.TestFunction.OutputArg='Boolean';
assigncellstotracksloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.FunctionInstance=
'MakeUnassignedCellsList';
assigncellstotracksloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.OutputArg='Unassi
gnedCellsIDs';
assigncellstotracksloop.FunctionArgs.GetCurrentTracks_Tracks.FunctionInstance='GetCurrentTracks';
assigncellstotracksloop.FunctionArgs.GetCurrentTracks_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.GetMaxTrackID_MaxTrackID.FunctionInstance='GetMaxTrackID';
assigncellstotracksloop.FunctionArgs.GetMaxTrackID_MaxTrackID.OutputArg='MaxTrackID';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_Centroids.FunctionInstance='IfIsEmptyPrev
iousCellsLabel';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_Centroids.InputArg='GetShapeParameters_Ce
ntroids';
assigncellstotracksloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='IfIsEmptyPre
viousCellsLabel';
assigncellstotracksloop.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout_Tra
cksLayout';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_ShapeParameters.FunctionInstance='IfIsEmp
tyPreviousCellsLabel';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_ShapeParameters.InputArg='GetShapeParamet
ers_ShapeParameters';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingNN_TrackAssignments.FunctionInstance='As
signCellToTrackUsingNN';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingNN_TrackAssignments.OutputArg='TrackAssi
gnments';
assigncellstotracksloop.KeepValues.SetMatchingGroupIndex_ShapeParameters.FunctionInstance='SetMat
chingGroupIndex';
assigncellstotracksloop.KeepValues.SetMatchingGroupIndex_ShapeParameters.OutputArg='ShapeParamete
rs';
assigncellstotracksloop.LoopFunctions=assign_cells_to_tracks_functions;
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,assign
cellstotracksloop);

continuetracks.InstanceName='ContinueTracks';
continuetracks.FunctionHandle=@continueTracks;
continuetracks.FunctionArgs.TimeFrame.Value=TimeFrame;
continuetracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
continuetracks.FunctionArgs.Tracks.OutputArg='Tracks';
continuetracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
continuetracks.FunctionArgs.Tracks.OutputArg2='Tracks';
continuetracks.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellsToTracksLoop';
continuetracks.FunctionArgs.TrackAssignments.OutputArg='AssignCellToTrackUsingNN_TrackAssignments
';
continuetracks.FunctionArgs.ShapeParameters.FunctionInstance='AssignCellsToTracksLoop';
continuetracks.FunctionArgs.ShapeParameters.OutputArg='SetMatchingGroupIndex_ShapeParameters';
continuetracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
continuetracks.FunctionArgs.CellsCentroids.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,contin
uetracks);

getmatchinggroupmeans.InstanceName='GetMatchingGroupMeans';
getmatchinggroupmeans.FunctionHandle=@getMatchingGroupMeans;
getmatchinggroupmeans.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getmatchinggroupmeans.FunctionArgs.Tracks.OutputArg='Tracks';
getmatchinggroupmeans.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getmatchinggroupmeans.FunctionArgs.Tracks.OutputArg2='Tracks';
getmatchinggroupmeans.FunctionArgs.TracksLayout.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getmatchinggroupmeans.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
```

```
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getmat
chinggroupmeans);

ifisemptypreviouscellslabel.InstanceName='IfIsEmptyPreviousCellsLabel';
ifisemptypreviouscellslabel.FunctionHandle=@if_statement;
ifisemptypreviouscellslabel.FunctionArgs.PreviousCellsLabel.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.TrackAssignments.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.TestVariable.FunctionInstance='IsEmptyPreviousCellsLabel
';
ifisemptypreviouscellslabel.FunctionArgs.TestVariable.OutputArg='Boolean';
ifisemptypreviouscellslabel.FunctionArgs.ResizeCytoLabel_Image.FunctionInstance='ResizeCytoLabel'
;
ifisemptypreviouscellslabel.FunctionArgs.ResizeCytoLabel_Image.OutputArg='Image';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_LoopCounter.FunctionInstance='Segmentat
ionLoop';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_LoopCounter.OutputArg='LoopCounter';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_ShapeParameters.FunctionInstance='Get
ShapeParameters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_ShapeParameters.OutputArg='ShapeParam
eters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_Centroids.FunctionInstance='GetShapeP
arameters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_Centroids.OutputArg='Centroids';
ifisemptypreviouscellslabel.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='Segmenta
tionLoop';
ifisemptypreviouscellslabel.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout
_TracksLayout';
ifisemptypreviouscellslabel.KeepValues.ContinueTracks_Tracks.FunctionInstance='ContinueTracks';
ifisemptypreviouscellslabel.KeepValues.ContinueTracks_Tracks.OutputArg='Tracks';
ifisemptypreviouscellslabel.ElseFunctions=else_is_empty_cells_label_functions;
ifisemptypreviouscellslabel.IfFunctions=if_is_empty_cells_label_functions;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,ifisemptypreviouscellslabe
l);

savecellslabel.InstanceName='SaveCellsLabel';
savecellslabel.FunctionHandle=@saveCellsLabel;
savecellslabel.FunctionArgs.FileRoot.Value=SegmentationFilesRoot;
savecellslabel.FunctionArgs.NumberFormat.Value=NumberFormat;
savecellslabel.FunctionArgs.CellsLabel.FunctionInstance='ResizeCytoLabel';
savecellslabel.FunctionArgs.CellsLabel.OutputArg='Image';
savecellslabel.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
savecellslabel.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,savecellslabel);

segmentationloop.InstanceName='SegmentationLoop';
segmentationloop.FunctionHandle=@forLoop;
segmentationloop.FunctionArgs.StartLoop.Value=StartFrame;
segmentationloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
segmentationloop.FunctionArgs.IncrementLoop.Value=FrameStep;
segmentationloop.FunctionArgs.Tracks.Value=[];
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='LoadTracksLayout';
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.OutputArg='TracksLayout';
segmentationloop.KeepValues.ContinueTracks_Tracks.FunctionInstance='IfIsEmptyPreviousCellsLabel';
segmentationloop.KeepValues.ContinueTracks_Tracks.OutputArg='ContinueTracks_Tracks';
segmentationloop.LoopFunctions=image_read_loop_functions;
functions_list=addToFunctionChain(functions_list,segmentationloop);

savetracks.InstanceName='SaveTracks';
savetracks.FunctionHandle=@saveTracks;
savetracks.FunctionArgs.TracksFileName.Value=[TracksFolder '/tracks.mat'];
savetracks.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
savetracks.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,savetracks);

gettrackids.InstanceName='GetTrackIDs';
gettrackids.FunctionHandle=@getTrackIDs;
gettrackids.FunctionArgs.TrackIDCol.Value=1;
gettrackids.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
gettrackids.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,gettrackids);

detectmergecandidates.InstanceName='DetectMergeCandidates';
detectmergecandidates.FunctionHandle=@detectMergeCandidatesUsingDistance;
detectmergecandidates.FunctionArgs.MaxMergeDistance.Value=MaxMergeDistance;
detectmergecandidates.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDs';
detectmergecandidates.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
detectmergecandidates.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
detectmergecandidates.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
detectmergecandidates.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
detectmergecandidates.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,detectmergecandidates);
```

```
mergetracks.InstanceName='MergeTracks';
mergetracks.FunctionHandle=@mergeTracks;
mergetracks.FunctionArgs.FrameCount.Value=FrameCount;
mergetracks.FunctionArgs.StartFrame.Value=StartFrame;
mergetracks.FunctionArgs.TimeFrame.Value=TimeFrame;
mergetracks.FunctionArgs.SegFileRoot.Value=SegmentationFilesRoot;
mergetracks.FunctionArgs.FrameStep.Value=FrameStep;
mergetracks.FunctionArgs.NumberFormat.Value=NumberFormat;
mergetracks.FunctionArgs.TracksToBeMerged.FunctionInstance='DetectMergeCandidates';
mergetracks.FunctionArgs.TracksToBeMerged.OutputArg='TracksToBeMerged';
mergetracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
mergetracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
mergetracks.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
mergetracks.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,mergetracks);

gettrackidsaftermerge.InstanceName='GetTrackIDsAfterMerge';
gettrackidsaftermerge.FunctionHandle=@getTrackIDs;
gettrackidsaftermerge.FunctionArgs.TrackIDCol.Value=1;
gettrackidsaftermerge.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
gettrackidsaftermerge.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,gettrackidsaftermerge);

makeancestryforfirstframecells.InstanceName='MakeAncestryForFirstFrameCells';
makeancestryforfirstframecells.FunctionHandle=@makeAncestryForFirstFrameCells;
makeancestryforfirstframecells.FunctionArgs.TimeCol.Value=2;
makeancestryforfirstframecells.FunctionArgs.TrackIDCol.Value=1;
makeancestryforfirstframecells.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
makeancestryforfirstframecells.FunctionArgs.Tracks.OutputArg='Tracks';
makeancestryforfirstframecells.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDsAfterMerge';
makeancestryforfirstframecells.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
functions_list=addToFunctionChain(functions_list,makeancestryforfirstframecells);

detectmitoticevents.InstanceName='DetectMitoticEvents';
detectmitoticevents.FunctionHandle=@detectMitoticEvents;
detectmitoticevents.FunctionArgs.MaxSplitArea.Value=MaxSplitArea;
detectmitoticevents.FunctionArgs.MinSplitEccentricity.Value=MinSplitEccentricity;
detectmitoticevents.FunctionArgs.MaxSplitEccentricity.Value=MaxSplitEccentricity;
detectmitoticevents.FunctionArgs.MaxSplitDistance.Value=MaxSplitDistance;
detectmitoticevents.FunctionArgs.MinTimeForSplit.Value=MinTimeForSplit;
detectmitoticevents.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
detectmitoticevents.FunctionArgs.Tracks.OutputArg='Tracks';
detectmitoticevents.FunctionArgs.UntestedIDs.FunctionInstance='MakeAncestryForFirstFrameCells';
detectmitoticevents.FunctionArgs.UntestedIDs.OutputArg='UntestedIDs';
detectmitoticevents.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
detectmitoticevents.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
functions_list=addToFunctionChain(functions_list,detectmitoticevents);

makeancestryforcellsenteringframes.InstanceName='MakeAncestryForCellsEnteringFrames';
makeancestryforcellsenteringframes.FunctionHandle=@makeAncestryForCellsEnteringFrames;
makeancestryforcellsenteringframes.FunctionArgs.TimeCol.Value=2;
makeancestryforcellsenteringframes.FunctionArgs.TrackIDCol.Value=1;
makeancestryforcellsenteringframes.FunctionArgs.SplitCells.FunctionInstance='DetectMitoticEvents'
;
makeancestryforcellsenteringframes.FunctionArgs.SplitCells.OutputArg='SplitCells';
makeancestryforcellsenteringframes.FunctionArgs.TrackIDs.FunctionInstance='GetTrackIDsAfterMerge'
;
makeancestryforcellsenteringframes.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
makeancestryforcellsenteringframes.FunctionArgs.FirstFrameIDs.FunctionInstance='MakeAncestryForFi
rstFrameCells';
makeancestryforcellsenteringframes.FunctionArgs.FirstFrameIDs.OutputArg='FirstFrameIDs';
makeancestryforcellsenteringframes.FunctionArgs.CellsAncestry.FunctionInstance='MakeAncestryForFi
rstFrameCells';
makeancestryforcellsenteringframes.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
makeancestryforcellsenteringframes.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
makeancestryforcellsenteringframes.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,makeancestryforcellsenteringframes);

splittracks.InstanceName='SplitTracks';
splittracks.FunctionHandle=@splitTracks;
splittracks.FunctionArgs.TimeFrame.Value=TimeFrame;
splittracks.FunctionArgs.SplitCells.FunctionInstance='DetectMitoticEvents';
splittracks.FunctionArgs.SplitCells.OutputArg='SplitCells';
splittracks.FunctionArgs.Tracks.FunctionInstance='MergeTracks';
splittracks.FunctionArgs.Tracks.OutputArg='Tracks';
splittracks.FunctionArgs.CellsAncestry.FunctionInstance='MakeAncestryForCellsEnteringFrames';
splittracks.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
splittracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
splittracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
splittracks.FunctionArgs.AncestryLayout.FunctionInstance='LoadAncestryLayout';
```

```
splittracks.FunctionArgs.AncestryLayout.OutputArg='AncestryLayout';
functions_list=addToFunctionChain(functions_list,splittracks);

removeshorttracks.InstanceName='RemoveShortTracks';
removeshorttracks.FunctionHandle=@removeShortTracks;
removeshorttracks.FunctionArgs.MinLifespan.Value=MinLifespan;
removeshorttracks.FunctionArgs.Tracks.FunctionInstance='SplitTracks';
removeshorttracks.FunctionArgs.Tracks.OutputArg='Tracks';
removeshorttracks.FunctionArgs.CellsAncestry.FunctionInstance='SplitTracks';
removeshorttracks.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
removeshorttracks.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
removeshorttracks.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
removeshorttracks.FunctionArgs.AncestryLayout.FunctionInstance='LoadAncestryLayout';
removeshorttracks.FunctionArgs.AncestryLayout.OutputArg='AncestryLayout';
functions_list=addToFunctionChain(functions_list,removeshorttracks);

saveupdatedtracks.InstanceName='SaveUpdatedTracks';
saveupdatedtracks.FunctionHandle=@saveTracks;
saveupdatedtracks.FunctionArgs.TracksFileName.Value=[AncestryFolder '/tracks.mat'];
saveupdatedtracks.FunctionArgs.Tracks.FunctionInstance='RemoveShortTracks';
saveupdatedtracks.FunctionArgs.Tracks.OutputArg='Tracks';
functions_list=addToFunctionChain(functions_list,saveupdatedtracks);

saveancestry.InstanceName='SaveAncestry';
saveancestry.FunctionHandle=@saveAncestry;
saveancestry.FunctionArgs.AncestryFileName.Value=[AncestryFolder '/ancestry.mat'];
saveancestry.FunctionArgs.CellsAncestry.FunctionInstance='RemoveShortTracks';
saveancestry.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
functions_list=addToFunctionChain(functions_list,saveancestry);

makeimagenamesinoverlayloop.InstanceName='MakeImageNamesInOverlayLoop';
makeimagenamesinoverlayloop.FunctionHandle=@makeImgFileName;
makeimagenamesinoverlayloop.FunctionArgs.FileBase.Value=ImageFileBase;
makeimagenamesinoverlayloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinoverlayloop.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenamesinoverlayloop.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
makeimagenamesinoverlayloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,makeimagenamesinover
layloop);

readimagesinoverlayloop.InstanceName='ReadImagesInOverlayLoop';
readimagesinoverlayloop.FunctionHandle=@readImage;
readimagesinoverlayloop.FunctionArgs.ImageChannel.Value='';
readimagesinoverlayloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInOverlayLoop';
readimagesinoverlayloop.FunctionArgs.ImageName.OutputArg='FileName';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,readimagesinoverlayl
oop);

getcurrenttracks2.InstanceName='GetCurrentTracks2';
getcurrenttracks2.FunctionHandle=@getCurrentTracks;
getcurrenttracks2.FunctionArgs.OffsetFrame.Value=0;
getcurrenttracks2.FunctionArgs.TimeFrame.Value=TimeFrame;
getcurrenttracks2.FunctionArgs.TimeCol.Value=2;
getcurrenttracks2.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks2.FunctionArgs.MaxMissingFrames.Value=0;
getcurrenttracks2.FunctionArgs.FrameStep.Value=FrameStep;
getcurrenttracks2.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
getcurrenttracks2.FunctionArgs.CurFrame.OutputArg='LoopCounter';
getcurrenttracks2.FunctionArgs.Tracks.FunctionInstance='ImageOverlayLoop';
getcurrenttracks2.FunctionArgs.Tracks.InputArg='RemoveShortTracks_Tracks';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,getcurrenttracks2);

makematnamesinoverlayloop.InstanceName='MakeMatNamesInOverlayLoop';
makematnamesinoverlayloop.FunctionHandle=@makeImgFileName;
makematnamesinoverlayloop.FunctionArgs.FileBase.Value=SegmentationFilesRoot;
makematnamesinoverlayloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makematnamesinoverlayloop.FunctionArgs.FileExt.Value='.mat';
makematnamesinoverlayloop.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
makematnamesinoverlayloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,makematnamesinoverla
yloop);

loadcellslabel.InstanceName='LoadCellsLabel';
loadcellslabel.FunctionHandle=@loadCellsLabel;
loadcellslabel.FunctionArgs.FileName.FunctionInstance='MakeMatNamesInOverlayLoop';
loadcellslabel.FunctionArgs.FileName.OutputArg='FileName';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,loadcellslabel);

loadcolormap.InstanceName='LoadColormap';
loadcolormap.FunctionHandle=@loadColormap;
loadcolormap.FunctionArgs.FileName.Value='colormap_lines';
```

```
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,loadcolormap);

displayoverlayingframe.InstanceName='DisplayOverlayingFrame';
displayoverlayingframe.FunctionHandle=@displayVariable;
displayoverlayingframe.FunctionArgs.VariableName.Value='Overlaying Frame';
displayoverlayingframe.FunctionArgs.Variable.FunctionInstance='ImageOverlayLoop';
displayoverlayingframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,displayoverlayingfra
me);

displayancestry.InstanceName='DisplayAncestry';
displayancestry.FunctionHandle=@displayAncestryData;
displayancestry.FunctionArgs.NumberFormat.Value=NumberFormat;
displayancestry.FunctionArgs.ProlDir.Value=AncestryFolder;
displayancestry.FunctionArgs.ImageFileName.Value=ImageFilesRoot;
displayancestry.FunctionArgs.DS.Value='/';
displayancestry.FunctionArgs.Image.FunctionInstance='ReadImagesInOverlayLoop';
displayancestry.FunctionArgs.Image.OutputArg='Image';
displayancestry.FunctionArgs.CurrentTracks.FunctionInstance='GetCurrentTracks2';
displayancestry.FunctionArgs.CurrentTracks.OutputArg='Tracks';
displayancestry.FunctionArgs.CellsLabel.FunctionInstance='LoadCellsLabel';
displayancestry.FunctionArgs.CellsLabel.OutputArg='LabelMatrix';
displayancestry.FunctionArgs.CurFrame.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.CurFrame.OutputArg='LoopCounter';
displayancestry.FunctionArgs.ColorMap.FunctionInstance='LoadColormap';
displayancestry.FunctionArgs.ColorMap.OutputArg='Colormap';
displayancestry.FunctionArgs.CellsAncestry.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.CellsAncestry.InputArg='RemoveShortTracks_CellsAncestry';
displayancestry.FunctionArgs.TracksLayout.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
displayancestry.FunctionArgs.AncestryLayout.FunctionInstance='ImageOverlayLoop';
displayancestry.FunctionArgs.AncestryLayout.InputArg='LoadAncestryLayout_AncestryLayout';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,displayancestry);

imageoverlayloop.InstanceName='ImageOverlayLoop';
imageoverlayloop.FunctionHandle=@forLoop;
imageoverlayloop.FunctionArgs.StartLoop.Value=StartFrame;
imageoverlayloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
imageoverlayloop.FunctionArgs.IncrementLoop.Value=FrameStep;
imageoverlayloop.FunctionArgs.RemoveShortTracks_Tracks.FunctionInstance='RemoveShortTracks';
imageoverlayloop.FunctionArgs.RemoveShortTracks_Tracks.OutputArg='Tracks';
imageoverlayloop.FunctionArgs.RemoveShortTracks_CellsAncestry.FunctionInstance='RemoveShortTracks
';
imageoverlayloop.FunctionArgs.RemoveShortTracks_CellsAncestry.OutputArg='CellsAncestry';
imageoverlayloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='LoadTracksLayout';
imageoverlayloop.FunctionArgs.LoadTracksLayout_TracksLayout.OutputArg='TracksLayout';
imageoverlayloop.FunctionArgs.LoadAncestryLayout_AncestryLayout.FunctionInstance='LoadAncestryLay
out';
imageoverlayloop.FunctionArgs.LoadAncestryLayout_AncestryLayout.OutputArg='AncestryLayout';
imageoverlayloop.LoopFunctions=image_overlay_loop_functions;
functions_list=addToFunctionChain(functions_list,imageoverlayloop);

saveancestryspreadsheets.InstanceName='SaveAncestrySpreadsheets';
saveancestryspreadsheets.FunctionHandle=@saveAncestrySpreadsheets;
saveancestryspreadsheets.FunctionArgs.ShapesXlsFile.Value=ShapesSpreadsheet;
saveancestryspreadsheets.FunctionArgs.ProlXlsFile.Value=AncestrySpreadsheet;
saveancestryspreadsheets.FunctionArgs.Tracks.FunctionInstance='RemoveShortTracks';
saveancestryspreadsheets.FunctionArgs.Tracks.OutputArg='Tracks';
saveancestryspreadsheets.FunctionArgs.CellsAncestry.FunctionInstance='RemoveShortTracks';
saveancestryspreadsheets.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
saveancestryspreadsheets.FunctionArgs.TracksLayout.FunctionInstance='LoadTracksLayout';
saveancestryspreadsheets.FunctionArgs.TracksLayout.OutputArg='TracksLayout';
functions_list=addToFunctionChain(functions_list,saveancestryspreadsheets);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayFluoNuclThresholding.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayFluoNuclThresholding()
%assayFluoNuclThresholding - This module is used to threshold a series of images (no object
segmentation
% and no  tracking are performed).  ImageFolder - String variable that specifies
```

```
%the  absolute location of the directory which contains the  time-lapse images. An
%example of  such a string variable would be 'c:/sample images/high-density'. ImageFilesRoot -
String
%variable specifying the  root image file name. The root image file name
%for a set of  images is the image file name of any of
%the  images without the  number or the file extension. For example, if
%the file name  is 'Experiment-0002_Position(8)_t021.tif'  the root image file name will be
%'Experiment-0002_Position(8)_t'. ImageExtension - String variable specifying the  image file
extension including the preceding
%dot. For example  if the file name  is 'image003.jpg' the image extension
%is '.jpg'. StartFrame - Number specifying the first image  in the sequence to
%be analyzed. The minimum  value for this variable depends  on the numbering
%of the image sequence so if  the first image in  the sequence
%is 'image003.tif' then the minimum value is 3. FrameCount - Number specifying  how
%many images from the image sequence should be processed. FrameStep - Number specifying
%the step size when reading images. Set this variable to 1  to read
% every image in the sequence, 2 to read every other image and
%so  on. NumberFormat - String value specifying the number of digits in the
%image file  names in  the sequence. For example if the image file
%name is 'image020.jpg'  the value for  the NumberFormat is '%03d', while if
%the file name is  'image000020.jpg' the value should  be '%06d'. OutputFolder -
%The folder where the thresholded  images will be saved. By default this value
% is set to a folder  named 'output' within the folder where the
%images to  be analyzed are located.  BrightnessThresholdPct - Number specifying the percentage
%threshold value for the image generated by the  generateBinImgUsingLocAvg  filter. Any pixel
%in the original image smaller than the threshold value  times the  corresponding
%value in the local average image below this value will  be set to
% zero while the rest will be set to one. ClearBorder  - Boolean
%value specifying whether objects next to or touching the image border in
%the binary images generated by the generateBinImgUsingGradient module will be erased (true) or
%not  (false). ClearBorderDist - Number specifying how close to the border objects may
% be and still be  erased if the ClearBorder parameter is set to
%true  in the generateBinImgUsingGradient module. ObjectArea - Number specifying the threshold
area for
%the clearSmallObjects,  polygonalAssistedWatershed filter. Objects below this  value will be
removed from
%the filtered image.  Strel - String variable specifying the type of filter used
%to generate the local  average  image in generateBinImgUsingLocAvg. Currently 'disk' is the
%only value supported. StrelSize -  Number specifying the size of the local neighborhood
%used to calculate the average   for each pixel in the local average
%image generated by the generateBinImgUsingLocAvg module. Important  Modules -
generateBinImgUsingLocAvg.

global functions_list;
functions_list=[];
%script variables
ImageFolder='C:/sample movies/low density';
ImageFilesRoot='low density sample';
ImageExtension='.tif';
StartFrame=1;
FrameCount=10;
FrameStep=1;
NumberFormat='%06d';
OutputFolder=[ImageFolder '/output'];
BrightnessThresholdPct=1.1;
ClearBorder=true;
ClearBorderDist=2;
ObjectArea=30;
Strel='disk';
StrelSize=10;
%end script variables

image_read_loop_functions=[];

makeoutputdir.InstanceName='MakeOutputDir';
makeoutputdir.FunctionHandle=@mkdir_Wrapper;
makeoutputdir.FunctionArgs.DirectoryName.Value=OutputFolder;
functions_list=addToFunctionChain(functions_list,makeoutputdir);

displaycurframe.InstanceName='DisplayCurFrame';
displaycurframe.FunctionHandle=@displayVariable;
displaycurframe.FunctionArgs.VariableName.Value='Current Tracking Frame';
displaycurframe.FunctionArgs.Variable.FunctionInstance='SegmentationLoop';
displaycurframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,displaycurframe);

makeimagenamesinsegmentationloop.InstanceName='MakeImageNamesInSegmentationLoop';
makeimagenamesinsegmentationloop.FunctionHandle=@makeImgFileName;
makeimagenamesinsegmentationloop.FunctionArgs.FileBase.Value=[ImageFolder '/' ImageFilesRoot];
makeimagenamesinsegmentationloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinsegmentationloop.FunctionArgs.FileExt.Value=ImageExtension;
```

```
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makeimagenamesinsegmentati
onloop);

readimagesinsegmentationloop.InstanceName='ReadImagesInSegmentationLoop';
readimagesinsegmentationloop.FunctionHandle=@readImage;
readimagesinsegmentationloop.FunctionArgs.ImageChannel.Value='';
readimagesinsegmentationloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInSegmentatio
nLoop';
readimagesinsegmentationloop.FunctionArgs.ImageName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,readimagesinsegmentationlo
op);

normalizeimageto16bit.InstanceName='NormalizeImageTo16Bit';
normalizeimageto16bit.FunctionHandle=@imNorm;
normalizeimageto16bit.FunctionArgs.IntegerClass.Value='uint16';
normalizeimageto16bit.FunctionArgs.RawImage.FunctionInstance='ReadImagesInSegmentationLoop';
normalizeimageto16bit.FunctionArgs.RawImage.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,normalizeimageto16bit);

cytobrightnesslocalaveragingfilter.InstanceName='CytoBrightnessLocalAveragingFilter';
cytobrightnesslocalaveragingfilter.FunctionHandle=@generateBinImgUsingLocAvg;
cytobrightnesslocalaveragingfilter.FunctionArgs.Strel.Value=Strel;
cytobrightnesslocalaveragingfilter.FunctionArgs.StrelSize.Value=StrelSize;
cytobrightnesslocalaveragingfilter.FunctionArgs.BrightnessThresholdPct.Value=BrightnessThresholdP
ct;
cytobrightnesslocalaveragingfilter.FunctionArgs.ClearBorder.Value=ClearBorder;
cytobrightnesslocalaveragingfilter.FunctionArgs.ClearBorderDist.Value=ClearBorderDist;
cytobrightnesslocalaveragingfilter.FunctionArgs.Image.FunctionInstance='NormalizeImageTo16Bit';
cytobrightnesslocalaveragingfilter.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,cytobrightnesslocalaveragi
ngfilter);

fillholescytoplasmimages.InstanceName='FillHolesCytoplasmImages';
fillholescytoplasmimages.FunctionHandle=@fillHoles;
fillholescytoplasmimages.FunctionArgs.Image.FunctionInstance='CytoBrightnessLocalAveragingFilter'
;
fillholescytoplasmimages.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,fillholescytoplasmimages);

clearsmallcells.InstanceName='ClearSmallCells';
clearsmallcells.FunctionHandle=@clearSmallObjects;
clearsmallcells.FunctionArgs.MinObjectArea.Value=ObjectArea;
clearsmallcells.FunctionArgs.Image.FunctionInstance='FillHolesCytoplasmImages';
clearsmallcells.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,clearsmallcells);

makethresholdimagenames.InstanceName='MakeThresholdImageNames';
makethresholdimagenames.FunctionHandle=@makeImgFileName;
makethresholdimagenames.FunctionArgs.FileBase.Value=[OutputFolder '/' ImageFilesRoot];
makethresholdimagenames.FunctionArgs.FileExt.Value=ImageExtension;
makethresholdimagenames.FunctionArgs.NumberFmt.Value=NumberFormat;
makethresholdimagenames.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
makethresholdimagenames.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makethresholdimagenames);

saveimage.InstanceName='SaveImage';
saveimage.FunctionHandle=@imwrite_Wrapper;
saveimage.FunctionArgs.Format.Value=ImageExtension(2:end);
saveimage.FunctionArgs.Image.FunctionInstance='ClearSmallCells';
saveimage.FunctionArgs.Image.OutputArg='Image';
saveimage.FunctionArgs.FileName.FunctionInstance='MakeThresholdImageNames';
saveimage.FunctionArgs.FileName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,saveimage);

segmentationloop.InstanceName='SegmentationLoop';
segmentationloop.FunctionHandle=@forLoop;
segmentationloop.FunctionArgs.StartLoop.Value=StartFrame;
segmentationloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
segmentationloop.FunctionArgs.IncrementLoop.Value=FrameStep;
segmentationloop.LoopFunctions=image_read_loop_functions;
functions_list=addToFunctionChain(functions_list,segmentationloop);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayFocalAdhesions3DNN.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayFocalAdhesions3DNN()
%assayFocalAdhesions3DNN - This module is used to track fluorescent focal adhesions in 3-D. It
saves
%    a set of images that display the outlines of the
%detected  FAs  as  well  as the ID of each individual
%FA overlyed  over the  original image.  The assay  also saves
%the position and  integrated intensity for  each detected FA  in a
%comma-delimited  file. ImageDirectory  - String variable that  specifies the
%  absolute location of   the directory which
%contains the    time-lapse  images. An   example
%of such a  string variable   would   be  'c:/sample
%  images/high-density'. ImageRoot - String variable     specifying the root
% image file    name. The root    image
%file name   for   a set   of images
%is the  image  file name      of any
% of the    images without   the number
%or the file extension.  For  example,  if the
%file name      is  'Experiment-0002_Position(8)_t021.tif' the  root
% image file name will  be 'Experiment-0002_Position(8)_t'. StartFrame - Number   specifying
%   the first image  in the sequence  to be
%analyzed.        The minimum  value  for
%this variable depends   on  the   numbering   of
%the  image    sequence so  if   the
% first  image in  the  sequence is  'image003.tif'
% then the   minimum  value   is 3.  FrameCount
%- Number   specifying  how   many images  from
% the image   sequence  should  be processed. TimeFrame
% -  Number specifying  the time    between  consecutive
%images in    minutes.  FrameStep -    Number
%specifying the step size when     reading images.  Set
%this variable    to  1   to   read
%every  image in    the  sequence,  2 to
%read every    other image and    so
%  on. NumberFormat  -  String value specifying the   number
%  of   digits   in the  image file names
%    in  the sequence.  For example
% if the     image file name is 'image020.jpg'  the
%value for          the NumberFormat is
%'%03d', while  if the  file name      is
%'image000020.jpg'    the value  should  be '%06d'.  MaxMissingFrames -
%    Number specifying for    how
%many frames a  cell  may   be disappear before its
%   track is       ended. OutputDirectory
%- The folder  where the overlayed  images   and track data
%   will     be saved. By
%default  this value  is set  to   a  folder
%named     'Output' within    the folder where
% the  images   to be analyzed are   located.
% BrightnessThresholdPct   -   Number   specifying the percentage
%threshold  a pixel  has to be brighter than pixels  in its
%neighborhood   to be assigned as a  FA pixel. MinFAArea -
% Number specifying   the minimum area for a FA. Objects  smaller
%than this value   will  not be considered FAs. MaxFAArea -
% Number specifying  the  maximum area for a FA. Objects greater than
%this value  will  not  be considered FAs. GlobalIntensityThresh - Percentage value
%that  determines if a    pixel is assigned as being part
%of the cytoplasm or part  of the    background. GaussStdDev -
%The standard deviation of the Gaussian kernel  used  to blur the image
%for processing. The kernel size of the Gaussian  function  used to blur
%the image.

global functions_list;
functions_list=[];
%script variables
ImageDirectory='C:\donna_webb\2_9_12_3Dmovies_new';
ImageRoot='Set001_s1_t';
StartFrame=1;
FrameCount=10;
TimeFrame=15;
FrameStep=1;
NumberFormat='%02d';
ImageExtension='.tif';
MaxMissingFrames=3;
OutputDirectory=[ImageDirectory '/Output'];
BrightnessThresholdPct=1.3;
```

```
GlobalIntensityThresh=0.2;
MinFAArea=10;
MaxFAArea=200;
GaussStdDev=0.5;
GaussKernSize=3;
ShowIDs=false;
%end script variables

AssignCellsToTrackLoopLoopFunctions=[];
ifisfirstlabelelsefunctions=[];
ifisfirstlabeliffunctions=[];
SegmentationLoopLoopFunctions=[];

loadtrackslayout.InstanceName='LoadTracksLayout';
loadtrackslayout.FunctionHandle=@loadTracksLayout;
loadtrackslayout.FunctionArgs.FileName.Value='tracks_layout_fa_3d.mat';
functions_list=addToFunctionChain(functions_list,loadtrackslayout);

makeoutputfolder.InstanceName='MakeOutputFolder';
makeoutputfolder.FunctionHandle=@mkdir_Wrapper;
makeoutputfolder.FunctionArgs.DirectoryName.Value=OutputDirectory;
functions_list=addToFunctionChain(functions_list,makeoutputfolder);

displaycurframe.InstanceName='DisplayCurFrame';
displaycurframe.FunctionHandle=@displayVariable;
displaycurframe.FunctionArgs.VariableName.Value='Current Tracking Frame';
displaycurframe.FunctionArgs.Variable.FunctionInstance='SegmentationLoop';
displaycurframe.FunctionArgs.Variable.OutputArg='LoopCounter';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,displaycurframe);

makeimagename.InstanceName='MakeImageName';
makeimagename.FunctionHandle=@makeImgFileName;
makeimagename.FunctionArgs.FileBase.Value=[ImageDirectory '/' ImageRoot];
makeimagename.FunctionArgs.FileExt.Value=ImageExtension;
makeimagename.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagename.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
makeimagename.FunctionArgs.CurFrame.OutputArg='LoopCounter';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,makeimagename);

read3dimage.InstanceName='Read3DImage';
read3dimage.FunctionHandle=@readImage3D;
read3dimage.FunctionArgs.ImageChannel.Value='';
read3dimage.FunctionArgs.ImageName.FunctionInstance='MakeImageName';
read3dimage.FunctionArgs.ImageName.OutputArg='FileName';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,read3dimage);

normalizeimageto16bit.InstanceName='NormalizeImageTo16Bit';
normalizeimageto16bit.FunctionHandle=@imNorm;
normalizeimageto16bit.FunctionArgs.IntegerClass.Value='uint16';
normalizeimageto16bit.FunctionArgs.RawImage.FunctionInstance='Read3DImage';
normalizeimageto16bit.FunctionArgs.RawImage.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,normalizeimageto16
bit);

gaussianfilter.InstanceName='GaussianFilter';
gaussianfilter.FunctionHandle=@gaussianFilter;
gaussianfilter.FunctionArgs.KernelSize.Value=GaussKernSize;
gaussianfilter.FunctionArgs.StandardDev.Value=GaussStdDev;
gaussianfilter.FunctionArgs.Image.FunctionInstance='Read3DImage';
gaussianfilter.FunctionArgs.Image.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,gaussianfilter);

normalizefilteredimageto16bit.InstanceName='NormalizeFilteredImageTo16Bit';
normalizefilteredimageto16bit.FunctionHandle=@imNorm;
normalizefilteredimageto16bit.FunctionArgs.IntegerClass.Value='uint16';
normalizefilteredimageto16bit.FunctionArgs.RawImage.FunctionInstance='GaussianFilter';
normalizefilteredimageto16bit.FunctionArgs.RawImage.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,normalizefilteredi
mageto16bit);

localaveragefilter.InstanceName='LocalAverageFilter';
localaveragefilter.FunctionHandle=@generateBinImgUsingLocAvg3D;
localaveragefilter.FunctionArgs.BrightnessThresholdPct.Value=BrightnessThresholdPct;
localaveragefilter.FunctionArgs.Strel.Value='disk';
localaveragefilter.FunctionArgs.StrelSize.Value=10;
localaveragefilter.FunctionArgs.Image.FunctionInstance='NormalizeFilteredImageTo16Bit';
localaveragefilter.FunctionArgs.Image.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,localaveragefilter
);

frequencyfilter.InstanceName='FrequencyFilter';
```

```
frequencyfilter.FunctionHandle=@butterworthFreqFilter3D;
frequencyfilter.FunctionArgs.CutOffFreq.Value=10;
frequencyfilter.FunctionArgs.FilterOrder.Value=6;
frequencyfilter.FunctionArgs.FilterType.Value='LowPass';
frequencyfilter.FunctionArgs.Image.FunctionInstance='GaussianFilter';
frequencyfilter.FunctionArgs.Image.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,frequencyfilter);

flattenbackground.InstanceName='FlattenBackground';
flattenbackground.FunctionHandle=@divideFunction;
flattenbackground.FunctionArgs.ConvertToDouble.Value=true;
flattenbackground.FunctionArgs.Var1.FunctionInstance='GaussianFilter';
flattenbackground.FunctionArgs.Var1.OutputArg='Image';
flattenbackground.FunctionArgs.Var2.FunctionInstance='FrequencyFilter';
flattenbackground.FunctionArgs.Var2.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,flattenbackground)
;

intensityfilter.InstanceName='IntensityFilter';
intensityfilter.FunctionHandle=@generateBinImgUsingGlobInt3D;
intensityfilter.FunctionArgs.IntensityThresholdPct.Value=GlobalIntensityThresh;
intensityfilter.FunctionArgs.Image.FunctionInstance='NormalizeFilteredImageTo16Bit';
intensityfilter.FunctionArgs.Image.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,intensityfilter);

clearnoncell.InstanceName='ClearNonCell';
clearnoncell.FunctionHandle=@clearSmallObjects;
clearnoncell.FunctionArgs.MinObjectArea.Value=5000;
clearnoncell.FunctionArgs.Image.FunctionInstance='IntensityFilter';
clearnoncell.FunctionArgs.Image.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,clearnoncell);

combineimages.InstanceName='CombineImages';
combineimages.FunctionHandle=@combineImages;
combineimages.FunctionArgs.CombineOperation.Value='AND';
combineimages.FunctionArgs.Image1.FunctionInstance='LocalAverageFilter';
combineimages.FunctionArgs.Image1.OutputArg='Image';
combineimages.FunctionArgs.Image2.FunctionInstance='ClearNonCell';
combineimages.FunctionArgs.Image2.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,combineimages);

clearborder.InstanceName='ClearBorder';
clearborder.FunctionHandle=@imclearborderSlices;
clearborder.FunctionArgs.Image.FunctionInstance='CombineImages';
clearborder.FunctionArgs.Image.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,clearborder);

clearsmallobjects.InstanceName='ClearSmallObjects';
clearsmallobjects.FunctionHandle=@clearSmallObjects;
clearsmallobjects.FunctionArgs.MinObjectArea.Value=MinFAArea;
clearsmallobjects.FunctionArgs.Image.FunctionInstance='ClearBorder';
clearsmallobjects.FunctionArgs.Image.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,clearsmallobjects)
;

labelfocaladhesions.InstanceName='LabelFocalAdhesions';
labelfocaladhesions.FunctionHandle=@labelObjects;
labelfocaladhesions.FunctionArgs.Image.FunctionInstance='ClearSmallObjects';
labelfocaladhesions.FunctionArgs.Image.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,labelfocaladhesion
s);

areafilter.InstanceName='AreaFilter';
areafilter.FunctionHandle=@areaFilterLabel;
areafilter.FunctionArgs.MaxArea.Value=MaxFAArea;
areafilter.FunctionArgs.ObjectsLabel.FunctionInstance='LabelFocalAdhesions';
areafilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,areafilter);

getcentroids.InstanceName='GetCentroids';
getcentroids.FunctionHandle=@getCentroids3D;
getcentroids.FunctionArgs.LabelMatrix.FunctionInstance='AreaFilter';
getcentroids.FunctionArgs.LabelMatrix.OutputArg='LabelMatrix';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,getcentroids);

getfasintegratedintensities.InstanceName='GetFAsIntegratedIntensities';
getfasintegratedintensities.FunctionHandle=@getIntegratedIntensities;
getfasintegratedintensities.FunctionArgs.IntensityImage.FunctionInstance='FlattenBackground';
getfasintegratedintensities.FunctionArgs.IntensityImage.OutputArg='Quotient';
getfasintegratedintensities.FunctionArgs.ObjectsLabel.FunctionInstance='AreaFilter';
getfasintegratedintensities.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
```

```
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,getfasintegratedin
tensities);

isemptylabel.InstanceName='IsEmptyLabel';
isemptylabel.FunctionHandle=@isEmptyFunction;
isemptylabel.FunctionArgs.TestVariable.Value=[];
isemptylabel.FunctionArgs.TestVariable.FunctionInstance='SaveFAsLabel';
isemptylabel.FunctionArgs.TestVariable.OutputArg='CellsLabel';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,isemptylabel);

getcurrenttracks.InstanceName='GetCurrentTracks';
getcurrenttracks.FunctionHandle=@getCurrentTracks;
getcurrenttracks.FunctionArgs.FrameStep.Value=FrameStep;
getcurrenttracks.FunctionArgs.MaxMissingFrames.Value=MaxMissingFrames;
getcurrenttracks.FunctionArgs.OffsetFrame.Value=-1;
getcurrenttracks.FunctionArgs.TimeCol.Value=2;
getcurrenttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getcurrenttracks.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg='Tracks';
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getcurrenttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsFirstLabel';
getcurrenttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
ifisfirstlabelelsefunctions=addToFunctionChain(ifisfirstlabelelsefunctions,getcurrenttracks);

getprevioustracks.InstanceName='GetPreviousTracks';
getprevioustracks.FunctionHandle=@getCurrentTracks;
getprevioustracks.FunctionArgs.FrameStep.Value=FrameStep;
getprevioustracks.FunctionArgs.MaxMissingFrames.Value=MaxMissingFrames;
getprevioustracks.FunctionArgs.OffsetFrame.Value=-2;
getprevioustracks.FunctionArgs.TimeCol.Value=2;
getprevioustracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getprevioustracks.FunctionArgs.TrackIDCol.Value=1;
getprevioustracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg='Tracks';
getprevioustracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getprevioustracks.FunctionArgs.CurFrame.FunctionInstance='IfIsFirstLabel';
getprevioustracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
ifisfirstlabelelsefunctions=addToFunctionChain(ifisfirstlabelelsefunctions,getprevioustracks);

makeunassignedcellslist.InstanceName='MakeUnassignedCellsList';
makeunassignedcellslist.FunctionHandle=@makeUnassignedCellsList;
makeunassignedcellslist.FunctionArgs.CellsCentroids.FunctionInstance='IfIsFirstLabel';
makeunassignedcellslist.FunctionArgs.CellsCentroids.InputArg='GetCentroids_Centroids';
ifisfirstlabelelsefunctions=addToFunctionChain(ifisfirstlabelelsefunctions,makeunassignedcellslis
t);

makeexcludedtrackslist.InstanceName='MakeExcludedTracksList';
makeexcludedtrackslist.FunctionHandle=@makeExcludedTracksList;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.FunctionInstance='MakeUnassignedCellsList'
;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.OutputArg='UnassignedCellsIDs';
ifisfirstlabelelsefunctions=addToFunctionChain(ifisfirstlabelelsefunctions,makeexcludedtrackslist
);

getmaxtrackid.InstanceName='GetMaxTrackID';
getmaxtrackid.FunctionHandle=@getMaxTrackID;
getmaxtrackid.FunctionArgs.TrackIDCol.Value=1;
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg='Tracks';
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg2='Tracks';
ifisfirstlabelelsefunctions=addToFunctionChain(ifisfirstlabelelsefunctions,getmaxtrackid);

isnotemptyunassignedlist.InstanceName='IsNotEmptyUnassignedList';
isnotemptyunassignedlist.FunctionHandle=@isNotEmptyFunction;
isnotemptyunassignedlist.FunctionArgs.TestVariable.FunctionInstance='AssignFAsToTracks';
isnotemptyunassignedlist.FunctionArgs.TestVariable.OutputArg='UnassignedIDs';
isnotemptyunassignedlist.FunctionArgs.TestVariable.FunctionInstance2='AssignCellsToTrackLoop';
isnotemptyunassignedlist.FunctionArgs.TestVariable.InputArg2='MakeUnassignedCellsList_UnassignedC
ellsIDs';
AssignCellsToTrackLoopLoopFunctions=addToFunctionChain(AssignCellsToTrackLoopLoopFunctions,isnote
mptyunassignedlist);

getcurrentunassignedcell.InstanceName='GetCurrentUnassignedCell';
getcurrentunassignedcell.FunctionHandle=@getCurrentUnassignedCell;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance='AssignFAsToTracks';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTrackLoop';
```

```
getcurrentunassignedcell.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassign
edCellsIDs';
AssignCellsToTrackLoopLoopFunctions=addToFunctionChain(AssignCellsToTrackLoopLoopFunctions,getcur
rentunassignedcell);

assignfastotracks.InstanceName='AssignFAsToTracks';
assignfastotracks.FunctionHandle=@assignCellToTrackUsingNN_3D;
assignfastotracks.FunctionArgs.TrackAssignments.Value=[];
assignfastotracks.FunctionArgs.TrackAssignments.FunctionInstance='AssignFAsToTracks';
assignfastotracks.FunctionArgs.TrackAssignments.OutputArg='TrackAssignments';
assignfastotracks.FunctionArgs.UnassignedCells.FunctionInstance='AssignFAsToTracks';
assignfastotracks.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
assignfastotracks.FunctionArgs.CellsCentroids.FunctionInstance='AssignCellsToTrackLoop';
assignfastotracks.FunctionArgs.CellsCentroids.InputArg='GetCentroids_Centroids';
assignfastotracks.FunctionArgs.CurrentTracks.FunctionInstance='AssignCellsToTrackLoop';
assignfastotracks.FunctionArgs.CurrentTracks.InputArg='GetCurrentTracks_Tracks';
assignfastotracks.FunctionArgs.MaxTrackID.FunctionInstance='AssignCellsToTrackLoop';
assignfastotracks.FunctionArgs.MaxTrackID.InputArg='GetMaxTrackID_MaxTrackID';
assignfastotracks.FunctionArgs.TracksLayout.FunctionInstance='AssignCellsToTrackLoop';
assignfastotracks.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
assignfastotracks.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTrackLoop';
assignfastotracks.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_UnassignedCells
IDs';
AssignCellsToTrackLoopLoopFunctions=addToFunctionChain(AssignCellsToTrackLoopLoopFunctions,assign
fastotracks);

assigncellstotrackloop.InstanceName='AssignCellsToTrackLoop';
assigncellstotrackloop.FunctionHandle=@whileLoop;
assigncellstotrackloop.FunctionArgs.TestFunction.FunctionInstance='IsNotEmptyUnassignedList';
assigncellstotrackloop.FunctionArgs.TestFunction.OutputArg='Boolean';
assigncellstotrackloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.FunctionInstance='
MakeUnassignedCellsList';
assigncellstotrackloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.OutputArg='Unassig
nedCellsIDs';
assigncellstotrackloop.FunctionArgs.GetCurrentTracks_Tracks.FunctionInstance='GetCurrentTracks';
assigncellstotrackloop.FunctionArgs.GetCurrentTracks_Tracks.OutputArg='Tracks';
assigncellstotrackloop.FunctionArgs.GetMaxTrackID_MaxTrackID.FunctionInstance='GetMaxTrackID';
assigncellstotrackloop.FunctionArgs.GetMaxTrackID_MaxTrackID.OutputArg='MaxTrackID';
assigncellstotrackloop.FunctionArgs.GetCentroids_Centroids.FunctionInstance='IfIsFirstLabel';
assigncellstotrackloop.FunctionArgs.GetCentroids_Centroids.InputArg='GetCentroids_Centroids';
assigncellstotrackloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='IfIsFirstLabe
l';
assigncellstotrackloop.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout_Trac
ksLayout';
assigncellstotrackloop.KeepValues.AssignFAsToTracks_TrackAssignments.FunctionInstance='AssignFAsT
oTracks';
assigncellstotrackloop.KeepValues.AssignFAsToTracks_TrackAssignments.OutputArg='TrackAssignments'
;
assigncellstotrackloop.LoopFunctions=AssignCellsToTrackLoopLoopFunctions;
ifisfirstlabelelsefunctions=addToFunctionChain(ifisfirstlabelelsefunctions,assigncellstotrackloop
);

continuetracks.InstanceName='ContinueTracks';
continuetracks.FunctionHandle=@continueTracks;
continuetracks.FunctionArgs.TimeFrame.Value=TimeFrame;
continuetracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
continuetracks.FunctionArgs.Tracks.OutputArg='Tracks';
continuetracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
continuetracks.FunctionArgs.Tracks.OutputArg2='Tracks';
continuetracks.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellsToTrackLoop';
continuetracks.FunctionArgs.TrackAssignments.OutputArg='AssignFAsToTracks_TrackAssignments';
continuetracks.FunctionArgs.CellsCentroids.FunctionInstance='IfIsFirstLabel';
continuetracks.FunctionArgs.CellsCentroids.InputArg='GetCentroids_Centroids';
continuetracks.FunctionArgs.CurFrame.FunctionInstance='IfIsFirstLabel';
continuetracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
continuetracks.FunctionArgs.ShapeParameters.FunctionInstance='IfIsFirstLabel';
continuetracks.FunctionArgs.ShapeParameters.InputArg='GetFAsIntegratedIntensities_IntegratedInten
sities';
ifisfirstlabelelsefunctions=addToFunctionChain(ifisfirstlabelelsefunctions,continuetracks);

displaytracks.InstanceName='DisplayTracks';
displaytracks.FunctionHandle=@displayTracksData3D;
displaytracks.FunctionArgs.FileRoot.Value=[OutputDirectory '/' ImageRoot];
displaytracks.FunctionArgs.NumberFormat.Value=NumberFormat;
displaytracks.FunctionArgs.TextSize.Value=0.5;
displaytracks.FunctionArgs.ShowIDs.Value=ShowIDs;
displaytracks.FunctionArgs.CurrentTracks.FunctionInstance='ContinueTracks';
displaytracks.FunctionArgs.CurrentTracks.OutputArg='NewTracks';
displaytracks.FunctionArgs.CurFrame.FunctionInstance='IfIsFirstLabel';
displaytracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
displaytracks.FunctionArgs.Image.FunctionInstance='IfIsFirstLabel';
```

```
displaytracks.FunctionArgs.Image.InputArg='NormalizeImageTo16Bit_Image';
displaytracks.FunctionArgs.ObjectsLabel.FunctionInstance='IfIsFirstLabel';
displaytracks.FunctionArgs.ObjectsLabel.InputArg='AreaFilter_LabelMatrix';
displaytracks.FunctionArgs.TracksLayout.FunctionInstance='IfIsFirstLabel';
displaytracks.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
ifisfirstlabelelsefunctions=addToFunctionChain(ifisfirstlabelelsefunctions,displaytracks);

starttracks.InstanceName='StartTracks';
starttracks.FunctionHandle=@startTracks3D;
starttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
starttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsFirstLabel';
starttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
starttracks.FunctionArgs.ObjectCentroids.FunctionInstance='IfIsFirstLabel';
starttracks.FunctionArgs.ObjectCentroids.InputArg='GetCentroids_Centroids';
starttracks.FunctionArgs.ShapeParameters.FunctionInstance='IfIsFirstLabel';
starttracks.FunctionArgs.ShapeParameters.InputArg='GetFAsIntegratedIntensities_IntegratedIntensit
ies';
ifisfirstlabeliffunctions=addToFunctionChain(ifisfirstlabeliffunctions,starttracks);

displayinitialtracks.InstanceName='DisplayInitialTracks';
displayinitialtracks.FunctionHandle=@displayTracksData3D;
displayinitialtracks.FunctionArgs.FileRoot.Value=[OutputDirectory '/' ImageRoot];
displayinitialtracks.FunctionArgs.NumberFormat.Value=NumberFormat;
displayinitialtracks.FunctionArgs.TextSize.Value=0.5;
displayinitialtracks.FunctionArgs.ShowIDs.Value=ShowIDs;
displayinitialtracks.FunctionArgs.CurrentTracks.FunctionInstance='StartTracks';
displayinitialtracks.FunctionArgs.CurrentTracks.OutputArg='Tracks';
displayinitialtracks.FunctionArgs.CurFrame.FunctionInstance='IfIsFirstLabel';
displayinitialtracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
displayinitialtracks.FunctionArgs.Image.FunctionInstance='IfIsFirstLabel';
displayinitialtracks.FunctionArgs.Image.InputArg='NormalizeImageTo16Bit_Image';
displayinitialtracks.FunctionArgs.ObjectsLabel.FunctionInstance='IfIsFirstLabel';
displayinitialtracks.FunctionArgs.ObjectsLabel.InputArg='AreaFilter_LabelMatrix';
displayinitialtracks.FunctionArgs.TracksLayout.FunctionInstance='IfIsFirstLabel';
displayinitialtracks.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
ifisfirstlabeliffunctions=addToFunctionChain(ifisfirstlabeliffunctions,displayinitialtracks);

ifisfirstlabel.InstanceName='IfIsFirstLabel';
ifisfirstlabel.FunctionHandle=@if_statement;
ifisfirstlabel.FunctionArgs.TestVariable.FunctionInstance='IsEmptyLabel';
ifisfirstlabel.FunctionArgs.TestVariable.OutputArg='Boolean';
ifisfirstlabel.FunctionArgs.SegmentationLoop_LoopCounter.FunctionInstance='SegmentationLoop';
ifisfirstlabel.FunctionArgs.SegmentationLoop_LoopCounter.OutputArg='LoopCounter';
ifisfirstlabel.FunctionArgs.GetCentroids_Centroids.FunctionInstance='GetCentroids';
ifisfirstlabel.FunctionArgs.GetCentroids_Centroids.OutputArg='Centroids';
ifisfirstlabel.FunctionArgs.GetFAsIntegratedIntensities_IntegratedIntensities.FunctionInstance='G
etFAsIntegratedIntensities';
ifisfirstlabel.FunctionArgs.GetFAsIntegratedIntensities_IntegratedIntensities.OutputArg='Integrat
edIntensities';
ifisfirstlabel.FunctionArgs.NormalizeImageTo16Bit_Image.FunctionInstance='NormalizeImageTo16Bit';
ifisfirstlabel.FunctionArgs.NormalizeImageTo16Bit_Image.OutputArg='Image';
ifisfirstlabel.FunctionArgs.AreaFilter_LabelMatrix.FunctionInstance='AreaFilter';
ifisfirstlabel.FunctionArgs.AreaFilter_LabelMatrix.OutputArg='LabelMatrix';
ifisfirstlabel.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='SegmentationLoop';
ifisfirstlabel.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout_TracksLayout
';
ifisfirstlabel.KeepValues.ContinueTracks_Tracks.FunctionInstance='ContinueTracks';
ifisfirstlabel.KeepValues.ContinueTracks_Tracks.OutputArg='Tracks';
ifisfirstlabel.ElseFunctions=ifisfirstlabelelsefunctions;
ifisfirstlabel.IfFunctions=ifisfirstlabeliffunctions;
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,ifisfirstlabel);

savefaslabel.InstanceName='SaveFAsLabel';
savefaslabel.FunctionHandle=@saveCellsLabel;
savefaslabel.FunctionArgs.FileRoot.Value=[OutputDirectory '/' ImageRoot];
savefaslabel.FunctionArgs.NumberFormat.Value=NumberFormat;
savefaslabel.FunctionArgs.CellsLabel.FunctionInstance='AreaFilter';
savefaslabel.FunctionArgs.CellsLabel.OutputArg='LabelMatrix';
savefaslabel.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
savefaslabel.FunctionArgs.CurFrame.OutputArg='LoopCounter';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,savefaslabel);

segmentationloop.InstanceName='SegmentationLoop';
segmentationloop.FunctionHandle=@forLoop;
segmentationloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
segmentationloop.FunctionArgs.IncrementLoop.Value=FrameStep;
segmentationloop.FunctionArgs.StartLoop.Value=StartFrame;
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='LoadTracksLayout';
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.OutputArg='TracksLayout';
segmentationloop.KeepValues.ContinueTracks_Tracks.FunctionInstance='IfIsFirstLabel';
segmentationloop.KeepValues.ContinueTracks_Tracks.OutputArg='ContinueTracks_Tracks';
```

```
segmentationloop.LoopFunctions=SegmentationLoopLoopFunctions;
functions_list=addToFunctionChain(functions_list,segmentationloop);

savetracks.InstanceName='SaveTracks';
savetracks.FunctionHandle=@saveTracks;
savetracks.FunctionArgs.TracksFileName.Value=[OutputDirectory '/tracks.mat'];
savetracks.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
savetracks.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,savetracks);

exportfadata.InstanceName='ExportFAData';
exportfadata.FunctionHandle=@saveMatrixToSpreadsheet;
exportfadata.FunctionArgs.SpreadsheetFileName.Value=[OutputDirectory '/' ImageRoot '_FAs.csv'];
exportfadata.FunctionArgs.ColumnNames.Value='Cell ID,Time,Centroid 1,Centroid
2,Centroid3,IntegratedIntensity';
exportfadata.FunctionArgs.Matrix.FunctionInstance='SegmentationLoop';
exportfadata.FunctionArgs.Matrix.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,exportfadata);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayFocalAdhesionsNN.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayFocalAdhesionsNN()
%assayFocalAdhesions3DNN - This module is used to track fluorescent focal adhesions in 2-D. It
saves
%             a
%set  of  images  that  display  the  outlines
%of   the   detected   FAs   as
% well    as the   ID of   each
%individual   FA  overlyed   over  the   original
% image.   The   assay   also   saves
% the position   and   integrated intensity   for
% each detected   FA    in a   comma-delimited
%   file. ImageName   -    String variable that
%  specifies    the
%  absolute location of    the     image
%which          contains  the
%   time-lapse     images.  An
% example      of  such a
% string variable     would
% be  'c:/sample images/adhesionturnover.tif'. StartFrame  -  Number
%  specifying       the first  image
%   in  the sequence   to  be
%    analyzed.
% The minimum       value  for  this
% variable depends        on  the
%   numbering     of     the
%  image      sequence so
%  if   the      first
% image in     the    sequence is
%'image003.tif'            then the
%  minimum    value      is 3.
%    FrameCount   - Number    specifying
%  how        many images
%from     the image      sequence
%    should   be processed. TimeFrame
%   -    Number    specifying
%the  time     between  consecutive    images
%in          minutes.  FrameStep -
%       Number    specifying
% the step   size  when      reading
%images.    Set        this variable
%    to    1
%to    read     every    image
% in     the   sequence,    2
% to      read every
%other image   and        so
%       on.   NumberFormat
%-   String value specifying  the
%number      of     digits
% in   the     image  file names
```

```
%       in    the
%sequence that will be generated.    For example
%   if    the    desired output
%image   file name is   'image020.jpg'    the
%  value for
%      the    NumberFormat should be
%'%03d', while  if  the   file name   wanted
%   is   'image000020.jpg'       the
%  value    should    be '%06d'.
% MaxMissingFrames -          Number
%  specifying  for    how
% many frames     a     cell
%may    be  disappear before   its
%      track   is
%      ended. OutputDirectory     -
%   The  folder  where the overlayed    images
%   and track     data
%   will       be   saved.
%By         default    this
%value      is set  to    a
%   folder      named
% 'Output' within        the folder
%  where      the   images
% to  be analyzed   are
%located.    BrightnessThresholdPct      -
%   Number    specifying the percentage
%threshold      a  pixel  has  to be
%brighter   than  pixels   in its
% neighborhood   to   be assigned  as a
% FA  pixel. MinFAArea   -
% Number specifying   the minimum   area  for a
%    FA. Objects  smaller  than this value
%   will      not  be considered FAs.
% MaxFAArea -       Number  specifying  the
%      maximum area for a   FA. Objects
% greater than    this value   will
% not    be   considered FAs. GlobalIntensityThresh - Percentage value
%  that   determines       if a
%   pixel is assigned    as being  part
%  of  the  cytoplasm or part    of
%the       background. GaussStdDev -
%The   standard deviation of the   Gaussian kernel
%used    to blur  the image   for
%processing.  The  kernel   size of the Gaussian  function
%  used  to blur       the image.
%ShowIDs - boolean value indicating   whether the track IDs of the detected
%focal adhesions should be   displayed in the output images.

global functions_list;
functions_list=[];
%script variables
ImageName='C:\donna_webb\W_movie_4.tiff';
OutputDirectory='C:\donna_webb\Output';
StartFrame=1;
FrameCount=81;
TimeFrame=15;
FrameStep=1;
NumberFormat='%02d';
ImageExtension='.tif';
MaxMissingFrames=3;
BrightnessThresholdPct=1.05;
GlobalIntensityThresh=0.2;
MinFAArea=25;
MaxFAArea=500;
GaussStdDev=2;
GaussKernSize=4;
ShowIDs=false;
IDList=[];
%end script variables

AssignCellsToTrackLoopLoopFunctions=[];
ifisfirstlabelelsefunctions=[];
ifisfirstlabeliffunctions=[];
SegmentationLoopLoopFunctions=[];

loadtrackslayout.InstanceName='LoadTracksLayout';
loadtrackslayout.FunctionHandle=@loadTracksLayout;
loadtrackslayout.FunctionArgs.FileName.Value='tracks_layout_fa_3d.mat';
functions_list=addToFunctionChain(functions_list,loadtrackslayout);

getimageinfo.InstanceName='GetImageInfo';
```

```
getimageinfo.FunctionHandle=@getFileInfo;
getimageinfo.FunctionArgs.PathName.Value=ImageName;
functions_list=addToFunctionChain(functions_list,getimageinfo);

makeoutputfoldername.InstanceName='MakeOutputFolderName';
makeoutputfoldername.FunctionHandle=@concatenateText;
makeoutputfoldername.FunctionArgs.Text3.Value='_Output';
makeoutputfoldername.FunctionArgs.Text1.FunctionInstance='GetImageInfo';
makeoutputfoldername.FunctionArgs.Text1.OutputArg='DirName';
makeoutputfoldername.FunctionArgs.Text2.FunctionInstance='GetImageInfo';
makeoutputfoldername.FunctionArgs.Text2.OutputArg='FileName';
functions_list=addToFunctionChain(functions_list,makeoutputfoldername);

makeoutputfolder.InstanceName='MakeOutputFolder';
makeoutputfolder.FunctionHandle=@mkdir_Wrapper;
makeoutputfolder.FunctionArgs.DirectoryName.FunctionInstance='MakeOutputFolderName';
makeoutputfolder.FunctionArgs.DirectoryName.OutputArg='String';
functions_list=addToFunctionChain(functions_list,makeoutputfolder);

makeimagesrootname.InstanceName='MakeImagesRootName';
makeimagesrootname.FunctionHandle=@concatenateText;
makeimagesrootname.FunctionArgs.Text2.Value='/';
makeimagesrootname.FunctionArgs.Text1.FunctionInstance='MakeOutputFolderName';
makeimagesrootname.FunctionArgs.Text1.OutputArg='String';
makeimagesrootname.FunctionArgs.Text3.FunctionInstance='GetImageInfo';
makeimagesrootname.FunctionArgs.Text3.OutputArg='FileName';
functions_list=addToFunctionChain(functions_list,makeimagesrootname);

displaycurframe.InstanceName='DisplayCurFrame';
displaycurframe.FunctionHandle=@displayVariable;
displaycurframe.FunctionArgs.VariableName.Value='Current Tracking Frame';
displaycurframe.FunctionArgs.Variable.FunctionInstance='SegmentationLoop';
displaycurframe.FunctionArgs.Variable.OutputArg='LoopCounter';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,displaycurframe);

readimageslice.InstanceName='ReadImageSlice';
readimageslice.FunctionHandle=@readImageSlice;
readimageslice.FunctionArgs.ImageChannel.Value='r';
readimageslice.FunctionArgs.ImageName.Value=ImageName;
readimageslice.FunctionArgs.SliceIndex.FunctionInstance='SegmentationLoop';
readimageslice.FunctionArgs.SliceIndex.OutputArg='LoopCounter';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,readimageslice);

normalizeimageto16bit.InstanceName='NormalizeImageTo16Bit';
normalizeimageto16bit.FunctionHandle=@imNorm;
normalizeimageto16bit.FunctionArgs.IntegerClass.Value='uint16';
normalizeimageto16bit.FunctionArgs.RawImage.FunctionInstance='ReadImageSlice';
normalizeimageto16bit.FunctionArgs.RawImage.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,normalizeimageto16
bit);

gaussianfilter.InstanceName='GaussianFilter';
gaussianfilter.FunctionHandle=@gaussianFilter;
gaussianfilter.FunctionArgs.KernelSize.Value=GaussKernSize;
gaussianfilter.FunctionArgs.StandardDev.Value=GaussStdDev;
gaussianfilter.FunctionArgs.Image.FunctionInstance='NormalizeImageTo16Bit';
gaussianfilter.FunctionArgs.Image.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,gaussianfilter);

normalizefilteredimageto16bit.InstanceName='NormalizeFilteredImageTo16Bit';
normalizefilteredimageto16bit.FunctionHandle=@imNorm;
normalizefilteredimageto16bit.FunctionArgs.IntegerClass.Value='uint16';
normalizefilteredimageto16bit.FunctionArgs.RawImage.FunctionInstance='GaussianFilter';
normalizefilteredimageto16bit.FunctionArgs.RawImage.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,normalizefilteredi
mageto16bit);

localaveragefilter.InstanceName='LocalAverageFilter';
localaveragefilter.FunctionHandle=@generateBinImgUsingLocAvg;
localaveragefilter.FunctionArgs.BrightnessThresholdPct.Value=BrightnessThresholdPct;
localaveragefilter.FunctionArgs.ClearBorder.Value=true;
localaveragefilter.FunctionArgs.ClearBorderDist.Value=1;
localaveragefilter.FunctionArgs.Strel.Value='disk';
localaveragefilter.FunctionArgs.StrelSize.Value=10;
localaveragefilter.FunctionArgs.Image.FunctionInstance='NormalizeFilteredImageTo16Bit';
localaveragefilter.FunctionArgs.Image.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,localaveragefilter
);

frequencyfilter.InstanceName='FrequencyFilter';
frequencyfilter.FunctionHandle=@butterworthFreqFilter;
```

```
frequencyfilter.FunctionArgs.CutOffFreq.Value=10;
frequencyfilter.FunctionArgs.FilterOrder.Value=6;
frequencyfilter.FunctionArgs.FilterType.Value='LowPass';
frequencyfilter.FunctionArgs.Image.FunctionInstance='GaussianFilter';
frequencyfilter.FunctionArgs.Image.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,frequencyfilter);

flattenbackground.InstanceName='FlattenBackground';
flattenbackground.FunctionHandle=@divideFunction;
flattenbackground.FunctionArgs.ConvertToDouble.Value=true;
flattenbackground.FunctionArgs.Var1.FunctionInstance='GaussianFilter';
flattenbackground.FunctionArgs.Var1.OutputArg='Image';
flattenbackground.FunctionArgs.Var2.FunctionInstance='FrequencyFilter';
flattenbackground.FunctionArgs.Var2.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,flattenbackground)
;

intensityfilter.InstanceName='IntensityFilter';
intensityfilter.FunctionHandle=@generateBinImgUsingGlobInt;
intensityfilter.FunctionArgs.ClearBorder.Value=false;
intensityfilter.FunctionArgs.ClearBorderDist.Value=1;
intensityfilter.FunctionArgs.IntensityThresholdPct.Value=GlobalIntensityThresh;
intensityfilter.FunctionArgs.Image.FunctionInstance='NormalizeFilteredImageTo16Bit';
intensityfilter.FunctionArgs.Image.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,intensityfilter);

clearnoncell.InstanceName='ClearNonCell';
clearnoncell.FunctionHandle=@clearSmallObjects;
clearnoncell.FunctionArgs.MinObjectArea.Value=5000;
clearnoncell.FunctionArgs.Image.FunctionInstance='IntensityFilter';
clearnoncell.FunctionArgs.Image.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,clearnoncell);

combineimages.InstanceName='CombineImages';
combineimages.FunctionHandle=@combineImages;
combineimages.FunctionArgs.CombineOperation.Value='AND';
combineimages.FunctionArgs.Image1.FunctionInstance='LocalAverageFilter';
combineimages.FunctionArgs.Image1.OutputArg='Image';
combineimages.FunctionArgs.Image2.FunctionInstance='ClearNonCell';
combineimages.FunctionArgs.Image2.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,combineimages);

clearsmallobjects.InstanceName='ClearSmallObjects';
clearsmallobjects.FunctionHandle=@clearSmallObjects;
clearsmallobjects.FunctionArgs.MinObjectArea.Value=MinFAArea;
clearsmallobjects.FunctionArgs.Image.FunctionInstance='CombineImages';
clearsmallobjects.FunctionArgs.Image.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,clearsmallobjects)
;

labelfocaladhesions.InstanceName='LabelFocalAdhesions';
labelfocaladhesions.FunctionHandle=@labelObjects;
labelfocaladhesions.FunctionArgs.Image.FunctionInstance='ClearSmallObjects';
labelfocaladhesions.FunctionArgs.Image.OutputArg='Image';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,labelfocaladhesion
s);

areafilter.InstanceName='AreaFilter';
areafilter.FunctionHandle=@areaFilterLabel;
areafilter.FunctionArgs.MaxArea.Value=MaxFAArea;
areafilter.FunctionArgs.ObjectsLabel.FunctionInstance='LabelFocalAdhesions';
areafilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,areafilter);

getcentroids.InstanceName='GetCentroids';
getcentroids.FunctionHandle=@getCentroids;
getcentroids.FunctionArgs.LabelMatrix.FunctionInstance='AreaFilter';
getcentroids.FunctionArgs.LabelMatrix.OutputArg='LabelMatrix';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,getcentroids);

getfasintegratedintensities.InstanceName='GetFAsIntegratedIntensities';
getfasintegratedintensities.FunctionHandle=@getIntegratedIntensities;
getfasintegratedintensities.FunctionArgs.IntensityImage.FunctionInstance='FlattenBackground';
getfasintegratedintensities.FunctionArgs.IntensityImage.OutputArg='Quotient';
getfasintegratedintensities.FunctionArgs.ObjectsLabel.FunctionInstance='AreaFilter';
getfasintegratedintensities.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,getfasintegratedin
tensities);

isemptylabel.InstanceName='IsEmptyLabel';
isemptylabel.FunctionHandle=@isEmptyFunction;
```

```
isemptylabel.FunctionArgs.TestVariable.Value=[];
isemptylabel.FunctionArgs.TestVariable.FunctionInstance='SaveFAsLabel';
isemptylabel.FunctionArgs.TestVariable.OutputArg='CellsLabel';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,isemptylabel);

getcurrenttracks.InstanceName='GetCurrentTracks';
getcurrenttracks.FunctionHandle=@getCurrentTracks;
getcurrenttracks.FunctionArgs.FrameStep.Value=FrameStep;
getcurrenttracks.FunctionArgs.MaxMissingFrames.Value=MaxMissingFrames;
getcurrenttracks.FunctionArgs.OffsetFrame.Value=-1;
getcurrenttracks.FunctionArgs.TimeCol.Value=2;
getcurrenttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getcurrenttracks.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg='Tracks';
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getcurrenttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsFirstLabel';
getcurrenttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
ifisfirstlabelelsefunctions=addToFunctionChain(ifisfirstlabelelsefunctions,getcurrenttracks);

getprevioustracks.InstanceName='GetPreviousTracks';
getprevioustracks.FunctionHandle=@getCurrentTracks;
getprevioustracks.FunctionArgs.FrameStep.Value=FrameStep;
getprevioustracks.FunctionArgs.MaxMissingFrames.Value=MaxMissingFrames;
getprevioustracks.FunctionArgs.OffsetFrame.Value=-2;
getprevioustracks.FunctionArgs.TimeCol.Value=2;
getprevioustracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getprevioustracks.FunctionArgs.TrackIDCol.Value=1;
getprevioustracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg='Tracks';
getprevioustracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getprevioustracks.FunctionArgs.CurFrame.FunctionInstance='IfIsFirstLabel';
getprevioustracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
ifisfirstlabelelsefunctions=addToFunctionChain(ifisfirstlabelelsefunctions,getprevioustracks);

makeunassignedcellslist.InstanceName='MakeUnassignedCellsList';
makeunassignedcellslist.FunctionHandle=@makeUnassignedCellsList;
makeunassignedcellslist.FunctionArgs.CellsCentroids.FunctionInstance='IfIsFirstLabel';
makeunassignedcellslist.FunctionArgs.CellsCentroids.InputArg='GetCentroids_Centroids';
ifisfirstlabelelsefunctions=addToFunctionChain(ifisfirstlabelelsefunctions,makeunassignedcellslis
t);

makeexcludedtrackslist.InstanceName='MakeExcludedTracksList';
makeexcludedtrackslist.FunctionHandle=@makeExcludedTracksList;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.FunctionInstance='MakeUnassignedCellsList'
;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.OutputArg='UnassignedCellsIDs';
ifisfirstlabelelsefunctions=addToFunctionChain(ifisfirstlabelelsefunctions,makeexcludedtrackslist
);

getmaxtrackid.InstanceName='GetMaxTrackID';
getmaxtrackid.FunctionHandle=@getMaxTrackID;
getmaxtrackid.FunctionArgs.TrackIDCol.Value=1;
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg='Tracks';
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg2='Tracks';
ifisfirstlabelelsefunctions=addToFunctionChain(ifisfirstlabelelsefunctions,getmaxtrackid);

isnotemptyunassignedlist.InstanceName='IsNotEmptyUnassignedList';
isnotemptyunassignedlist.FunctionHandle=@isNotEmptyFunction;
isnotemptyunassignedlist.FunctionArgs.TestVariable.FunctionInstance='AssignFAsToTracks';
isnotemptyunassignedlist.FunctionArgs.TestVariable.OutputArg='UnassignedIDs';
isnotemptyunassignedlist.FunctionArgs.TestVariable.FunctionInstance2='AssignCellsToTrackLoop';
isnotemptyunassignedlist.FunctionArgs.TestVariable.InputArg2='MakeUnassignedCellsList_UnassignedC
ellsIDs';
AssignCellsToTrackLoopLoopFunctions=addToFunctionChain(AssignCellsToTrackLoopLoopFunctions,isnote
mptyunassignedlist);

getcurrentunassignedcell.InstanceName='GetCurrentUnassignedCell';
getcurrentunassignedcell.FunctionHandle=@getCurrentUnassignedCell;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance='AssignFAsToTracks';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTrackLoop';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassign
edCellsIDs';
AssignCellsToTrackLoopLoopFunctions=addToFunctionChain(AssignCellsToTrackLoopLoopFunctions,getcur
rentunassignedcell);
```

```
assignfastotracks.InstanceName='AssignFAsToTracks';
assignfastotracks.FunctionHandle=@assignCellToTrackUsingNN;
assignfastotracks.FunctionArgs.TrackAssignments.Value=[];
assignfastotracks.FunctionArgs.TrackAssignments.FunctionInstance='AssignFAsToTracks';
assignfastotracks.FunctionArgs.TrackAssignments.OutputArg='TrackAssignments';
assignfastotracks.FunctionArgs.UnassignedCells.FunctionInstance='AssignFAsToTracks';
assignfastotracks.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
assignfastotracks.FunctionArgs.CellsCentroids.FunctionInstance='AssignCellsToTrackLoop';
assignfastotracks.FunctionArgs.CellsCentroids.InputArg='GetCentroids_Centroids';
assignfastotracks.FunctionArgs.CurrentTracks.FunctionInstance='AssignCellsToTrackLoop';
assignfastotracks.FunctionArgs.CurrentTracks.InputArg='GetCurrentTracks_Tracks';
assignfastotracks.FunctionArgs.MaxTrackID.FunctionInstance='AssignCellsToTrackLoop';
assignfastotracks.FunctionArgs.MaxTrackID.InputArg='GetMaxTrackID_MaxTrackID';
assignfastotracks.FunctionArgs.TracksLayout.FunctionInstance='AssignCellsToTrackLoop';
assignfastotracks.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
assignfastotracks.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTrackLoop';
assignfastotracks.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_UnassignedCells
IDs';
AssignCellsToTrackLoopLoopFunctions=addToFunctionChain(AssignCellsToTrackLoopLoopFunctions,assign
fastotracks);

assigncellstotrackloop.InstanceName='AssignCellsToTrackLoop';
assigncellstotrackloop.FunctionHandle=@whileLoop;
assigncellstotrackloop.FunctionArgs.TestFunction.FunctionInstance='IsNotEmptyUnassignedList';
assigncellstotrackloop.FunctionArgs.TestFunction.OutputArg='Boolean';
assigncellstotrackloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.FunctionInstance='
MakeUnassignedCellsList';
assigncellstotrackloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.OutputArg='Unassig
nedCellsIDs';
assigncellstotrackloop.FunctionArgs.GetCurrentTracks_Tracks.FunctionInstance='GetCurrentTracks';
assigncellstotrackloop.FunctionArgs.GetCurrentTracks_Tracks.OutputArg='Tracks';
assigncellstotrackloop.FunctionArgs.GetMaxTrackID_MaxTrackID.FunctionInstance='GetMaxTrackID';
assigncellstotrackloop.FunctionArgs.GetMaxTrackID_MaxTrackID.OutputArg='MaxTrackID';
assigncellstotrackloop.FunctionArgs.GetCentroids_Centroids.FunctionInstance='IfIsFirstLabel';
assigncellstotrackloop.FunctionArgs.GetCentroids_Centroids.InputArg='GetCentroids_Centroids';
assigncellstotrackloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='IfIsFirstLabe
l';
assigncellstotrackloop.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout_Trac
ksLayout';
assigncellstotrackloop.KeepValues.AssignFAsToTracks_TrackAssignments.FunctionInstance='AssignFAsT
oTracks';
assigncellstotrackloop.KeepValues.AssignFAsToTracks_TrackAssignments.OutputArg='TrackAssignments'
;
assigncellstotrackloop.LoopFunctions=AssignCellsToTrackLoopLoopFunctions;
ifisfirstlabelelsefunctions=addToFunctionChain(ifisfirstlabelelsefunctions,assigncellstotrackloop
);

continuetracks.InstanceName='ContinueTracks';
continuetracks.FunctionHandle=@continueTracks;
continuetracks.FunctionArgs.TimeFrame.Value=TimeFrame;
continuetracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
continuetracks.FunctionArgs.Tracks.OutputArg='Tracks';
continuetracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
continuetracks.FunctionArgs.Tracks.OutputArg2='Tracks';
continuetracks.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellsToTrackLoop';
continuetracks.FunctionArgs.TrackAssignments.OutputArg='AssignFAsToTracks_TrackAssignments';
continuetracks.FunctionArgs.CellsCentroids.FunctionInstance='IfIsFirstLabel';
continuetracks.FunctionArgs.CellsCentroids.InputArg='GetCentroids_Centroids';
continuetracks.FunctionArgs.CurFrame.FunctionInstance='IfIsFirstLabel';
continuetracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
continuetracks.FunctionArgs.ShapeParameters.FunctionInstance='IfIsFirstLabel';
continuetracks.FunctionArgs.ShapeParameters.InputArg='GetFAsIntegratedIntensities_IntegratedInten
sities';
ifisfirstlabelelsefunctions=addToFunctionChain(ifisfirstlabelelsefunctions,continuetracks);

displaytracks.InstanceName='DisplayTracks';
displaytracks.FunctionHandle=@displayTracksData;
displaytracks.FunctionArgs.IDList.Value=IDList;
displaytracks.FunctionArgs.LabelColorRGB.Value=[0 255 0];
displaytracks.FunctionArgs.NumberFormat.Value=NumberFormat;
displaytracks.FunctionArgs.ShowIDs.Value=true;
displaytracks.FunctionArgs.CurrentTracks.FunctionInstance='ContinueTracks';
displaytracks.FunctionArgs.CurrentTracks.OutputArg='NewTracks';
displaytracks.FunctionArgs.CurFrame.FunctionInstance='IfIsFirstLabel';
displaytracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
displaytracks.FunctionArgs.Image.FunctionInstance='IfIsFirstLabel';
displaytracks.FunctionArgs.Image.InputArg='NormalizeImageTo16Bit_Image';
displaytracks.FunctionArgs.TracksLayout.FunctionInstance='IfIsFirstLabel';
displaytracks.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
displaytracks.FunctionArgs.FileRoot.FunctionInstance='IfIsFirstLabel';
displaytracks.FunctionArgs.FileRoot.InputArg='MakeImagesRootName_String';
```

```
displaytracks.FunctionArgs.CellsLabel.FunctionInstance='IfIsFirstLabel';
displaytracks.FunctionArgs.CellsLabel.InputArg='AreaFilter_LabelMatrix';
ifisfirstlabelelsefunctions=addToFunctionChain(ifisfirstlabelelsefunctions,displaytracks);

starttracks.InstanceName='StartTracks';
starttracks.FunctionHandle=@startTracks3D;
starttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
starttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsFirstLabel';
starttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
starttracks.FunctionArgs.ObjectCentroids.FunctionInstance='IfIsFirstLabel';
starttracks.FunctionArgs.ObjectCentroids.InputArg='GetCentroids_Centroids';
starttracks.FunctionArgs.ShapeParameters.FunctionInstance='IfIsFirstLabel';
starttracks.FunctionArgs.ShapeParameters.InputArg='GetFAsIntegratedIntensities_IntegratedIntensit
ies';
ifisfirstlabeliffunctions=addToFunctionChain(ifisfirstlabeliffunctions,starttracks);

displayinitialtracks.InstanceName='DisplayInitialTracks';
displayinitialtracks.FunctionHandle=@displayTracksData;
displayinitialtracks.FunctionArgs.LabelColorRGB.Value=[0 255 0];
displayinitialtracks.FunctionArgs.NumberFormat.Value=NumberFormat;
displayinitialtracks.FunctionArgs.ShowIDs.Value=true;
displayinitialtracks.FunctionArgs.IDList.Value=IDList;
displayinitialtracks.FunctionArgs.CurrentTracks.FunctionInstance='StartTracks';
displayinitialtracks.FunctionArgs.CurrentTracks.OutputArg='Tracks';
displayinitialtracks.FunctionArgs.CurFrame.FunctionInstance='IfIsFirstLabel';
displayinitialtracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
displayinitialtracks.FunctionArgs.Image.FunctionInstance='IfIsFirstLabel';
displayinitialtracks.FunctionArgs.Image.InputArg='NormalizeImageTo16Bit_Image';
displayinitialtracks.FunctionArgs.TracksLayout.FunctionInstance='IfIsFirstLabel';
displayinitialtracks.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
displayinitialtracks.FunctionArgs.FileRoot.FunctionInstance='IfIsFirstLabel';
displayinitialtracks.FunctionArgs.FileRoot.InputArg='MakeImagesRootName_String';
displayinitialtracks.FunctionArgs.CellsLabel.FunctionInstance='IfIsFirstLabel';
displayinitialtracks.FunctionArgs.CellsLabel.InputArg='AreaFilter_LabelMatrix';
ifisfirstlabeliffunctions=addToFunctionChain(ifisfirstlabeliffunctions,displayinitialtracks);

ifisfirstlabel.InstanceName='IfIsFirstLabel';
ifisfirstlabel.FunctionHandle=@if_statement;
ifisfirstlabel.FunctionArgs.TestVariable.FunctionInstance='IsEmptyLabel';
ifisfirstlabel.FunctionArgs.TestVariable.OutputArg='Boolean';
ifisfirstlabel.FunctionArgs.SegmentationLoop_LoopCounter.FunctionInstance='SegmentationLoop';
ifisfirstlabel.FunctionArgs.SegmentationLoop_LoopCounter.OutputArg='LoopCounter';
ifisfirstlabel.FunctionArgs.GetCentroids_Centroids.FunctionInstance='GetCentroids';
ifisfirstlabel.FunctionArgs.GetCentroids_Centroids.OutputArg='Centroids';
ifisfirstlabel.FunctionArgs.GetFAsIntegratedIntensities_IntegratedIntensities.FunctionInstance='G
etFAsIntegratedIntensities';
ifisfirstlabel.FunctionArgs.GetFAsIntegratedIntensities_IntegratedIntensities.OutputArg='Integrat
edIntensities';
ifisfirstlabel.FunctionArgs.NormalizeImageTo16Bit_Image.FunctionInstance='NormalizeImageTo16Bit';
ifisfirstlabel.FunctionArgs.NormalizeImageTo16Bit_Image.OutputArg='Image';
ifisfirstlabel.FunctionArgs.AreaFilter_LabelMatrix.FunctionInstance='AreaFilter';
ifisfirstlabel.FunctionArgs.AreaFilter_LabelMatrix.OutputArg='LabelMatrix';
ifisfirstlabel.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='SegmentationLoop';
ifisfirstlabel.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout_TracksLayout
';
ifisfirstlabel.FunctionArgs.MakeImagesRootName_String.FunctionInstance='SegmentationLoop';
ifisfirstlabel.FunctionArgs.MakeImagesRootName_String.InputArg='MakeImagesRootName_String';
ifisfirstlabel.KeepValues.ContinueTracks_Tracks.FunctionInstance='ContinueTracks';
ifisfirstlabel.KeepValues.ContinueTracks_Tracks.OutputArg='Tracks';
ifisfirstlabel.ElseFunctions=ifisfirstlabelelsefunctions;
ifisfirstlabel.IfFunctions=ifisfirstlabeliffunctions;
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,ifisfirstlabel);

savefaslabel.InstanceName='SaveFAsLabel';
savefaslabel.FunctionHandle=@saveCellsLabel;
savefaslabel.FunctionArgs.NumberFormat.Value=NumberFormat;
savefaslabel.FunctionArgs.CellsLabel.FunctionInstance='AreaFilter';
savefaslabel.FunctionArgs.CellsLabel.OutputArg='LabelMatrix';
savefaslabel.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
savefaslabel.FunctionArgs.CurFrame.OutputArg='LoopCounter';
savefaslabel.FunctionArgs.FileRoot.FunctionInstance='SegmentationLoop';
savefaslabel.FunctionArgs.FileRoot.InputArg='MakeImagesRootName_String';
SegmentationLoopLoopFunctions=addToFunctionChain(SegmentationLoopLoopFunctions,savefaslabel);

segmentationloop.InstanceName='SegmentationLoop';
segmentationloop.FunctionHandle=@forLoop;
segmentationloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
segmentationloop.FunctionArgs.IncrementLoop.Value=FrameStep;
segmentationloop.FunctionArgs.StartLoop.Value=StartFrame;
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='LoadTracksLayout';
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.OutputArg='TracksLayout';
```

```
segmentationloop.FunctionArgs.MakeImagesRootName_String.FunctionInstance='MakeImagesRootName';
segmentationloop.FunctionArgs.MakeImagesRootName_String.OutputArg='String';
segmentationloop.KeepValues.ContinueTracks_Tracks.FunctionInstance='IfIsFirstLabel';
segmentationloop.KeepValues.ContinueTracks_Tracks.OutputArg='ContinueTracks_Tracks';
segmentationloop.LoopFunctions=SegmentationLoopLoopFunctions;
functions_list=addToFunctionChain(functions_list,segmentationloop);

maketracksfilename.InstanceName='MakeTracksFileName';
maketracksfilename.FunctionHandle=@concatenateText;
maketracksfilename.FunctionArgs.Text2.Value='/tracks.mat';
maketracksfilename.FunctionArgs.Text1.FunctionInstance='MakeOutputFolderName';
maketracksfilename.FunctionArgs.Text1.OutputArg='String';
functions_list=addToFunctionChain(functions_list,maketracksfilename);

savetracks.InstanceName='SaveTracks';
savetracks.FunctionHandle=@saveTracks;
savetracks.FunctionArgs.TracksFileName.FunctionInstance='MakeTracksFileName';
savetracks.FunctionArgs.TracksFileName.OutputArg='String';
savetracks.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
savetracks.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,savetracks);

makespreadsheetname.InstanceName='MakeSpreadsheetName';
makespreadsheetname.FunctionHandle=@concatenateText;
makespreadsheetname.FunctionArgs.Text2.Value='_FAs.csv';
makespreadsheetname.FunctionArgs.Text1.FunctionInstance='MakeImagesRootName';
makespreadsheetname.FunctionArgs.Text1.OutputArg='String';
functions_list=addToFunctionChain(functions_list,makespreadsheetname);

exportfadata.InstanceName='ExportFAData';
exportfadata.FunctionHandle=@saveMatrixToSpreadsheet;
exportfadata.FunctionArgs.ColumnNames.Value='Cell ID,Time,Centroid 1,Centroid
2,IntegratedIntensity';
exportfadata.FunctionArgs.SpreadsheetFileName.FunctionInstance='MakeSpreadsheetName';
exportfadata.FunctionArgs.SpreadsheetFileName.OutputArg='String';
exportfadata.FunctionArgs.Matrix.FunctionInstance='SegmentationLoop';
exportfadata.FunctionArgs.Matrix.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,exportfadata);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayGetStageOffset.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayGetStageOffset()
%assayGetStageOffset - This assay is used to calculate the microscope stage offset at each frame
%  by manually  clicking on a fixed point in every image. ImageFolder
%-  String  variable that specifies the absolute location of the directory which
%contains the   time-lapse  images. An example of such a string variable
%would be 'c:/sample  images/high-density'. ImageFilesRoot -  String variable specifying the root
image
%file name. The root  image file name   for a set of
%images is the image file  name of any of the   images
%without the number or the file  extension. For example, if the file
%name  is 'Experiment-0002_Position(8)_t021.tif' the root image  file name will be 'Experiment-
0002_Position(8)_t'. ImageFileBase
%-  The path name to the images.  This value is generated from
%the ImageFolder   and the ImageFilesRoot and should  not be changed. ImageExtension
%- String variable specifying the  image file extension including  the preceding dot.
%For example  if the file name  is 'image003.jpg' the  image extension
%is '.jpg'. NumberFormat - String value specifying the number  of digits  in
%the image file names in  the sequence. For example if  the
%image file name is 'image020.jpg' the value for  the NumberFormat is '%03d',
% while if the file name is 'image000020.jpg' the value should  be '%06d'.
%StartFrame   - Number specifying the first image in the sequence to be
%analyzed. The  minimum   value for this variable depends on the numbering
%of the image  sequence so  if  the first image in the
%sequence is 'image003.tif' then  the minimum value  is 3. FrameStep - Number
%specifying the step size when  reading images. Set this  variable to 1
% to read every image in  the sequence, 2 to read  every
%other image and  so on. FrameCount  - Number specifying how many images
% from the image sequence should be processed.  XYOffsets - Initial value for
%the stage  offsets array. Set to zero by  default. Important Modules -
```

```
%None.

global functions_list;
functions_list=[];
%script variables
ImageFolder='C:/peter/cropped';
ImageFilesRoot='peter';
ImageFileBase=[ImageFolder '/' ImageFilesRoot];
ImageExtension='.tif';
NumberFormat='%06d';
StartFrame=1;
FrameStep=1;
FrameCount=10;
XYOffsets=zeros(FrameCount,2);
%end script variables

image_read_loop_functions=[];

displaycurframe.InstanceName='DisplayCurFrame';
displaycurframe.FunctionHandle=@displayVariable;
displaycurframe.FunctionArgs.VariableName.Value='Current Tracking Frame';
displaycurframe.FunctionArgs.Variable.FunctionInstance='ProcessingLoop';
displaycurframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,displaycurframe);

makeimagenamesinprocessingloop.InstanceName='MakeImageNamesInProcessingLoop';
makeimagenamesinprocessingloop.FunctionHandle=@makeImgFileName;
makeimagenamesinprocessingloop.FunctionArgs.FileBase.Value=ImageFileBase;
makeimagenamesinprocessingloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinprocessingloop.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenamesinprocessingloop.FunctionArgs.CurFrame.FunctionInstance='ProcessingLoop';
makeimagenamesinprocessingloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makeimagenamesinprocessing
loop);

readimagesinprocessingloop.InstanceName='ReadImagesInProcessingLoop';
readimagesinprocessingloop.FunctionHandle=@readImage;
readimagesinprocessingloop.FunctionArgs.ImageChannel.Value='';
readimagesinprocessingloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInProcessingLoo
p';
readimagesinprocessingloop.FunctionArgs.ImageName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,readimagesinprocessingloop
);

displayimage.InstanceName='DisplayImage';
displayimage.FunctionHandle=@displayImage;
displayimage.FunctionArgs.FigureNr.Value=1;
displayimage.FunctionArgs.Image.FunctionInstance='ReadImagesInProcessingLoop';
displayimage.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,displayimage);

getxycoordinates.InstanceName='GetXYCoordinates';
getxycoordinates.FunctionHandle=@ginput_Wrapper;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getxycoordinates);

setoffset.InstanceName='SetOffset';
setoffset.FunctionHandle=@setArrayVar;
setoffset.FunctionArgs.Array.Value=XYOffsets;
setoffset.FunctionArgs.Index.FunctionInstance='ProcessingLoop';
setoffset.FunctionArgs.Index.OutputArg='LoopCounter';
setoffset.FunctionArgs.Var.FunctionInstance='GetXYCoordinates';
setoffset.FunctionArgs.Var.OutputArg='XYCoords';
setoffset.FunctionArgs.Array.FunctionInstance='SetOffset';
setoffset.FunctionArgs.Array.OutputArg='Array';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,setoffset);

processingloop.InstanceName='ProcessingLoop';
processingloop.FunctionHandle=@forLoop;
processingloop.FunctionArgs.StartLoop.Value=StartFrame;
processingloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
processingloop.FunctionArgs.IncrementLoop.Value=FrameStep;
processingloop.FunctionArgs.OffsetArray.Value=XYOffsets;
processingloop.KeepValues.SetOffset_Array.FunctionInstance='SetOffset';
processingloop.KeepValues.SetOffset_Array.OutputArg='Array';
processingloop.LoopFunctions=image_read_loop_functions;
functions_list=addToFunctionChain(functions_list,processingloop);

getfirstoffset.InstanceName='GetFirstOffset';
getfirstoffset.FunctionHandle=@getArrayVal;
getfirstoffset.FunctionArgs.Index.Value=1;
getfirstoffset.FunctionArgs.Array.FunctionInstance='ProcessingLoop';
```

```
getfirstoffset.FunctionArgs.Array.OutputArg='SetOffset_Array';
functions_list=addToFunctionChain(functions_list,getfirstoffset);

replicatefirstoffset.InstanceName='ReplicateFirstOffset';
replicatefirstoffset.FunctionHandle=@repmat_Wrapper;
replicatefirstoffset.FunctionArgs.RepeatDim.Value=[FrameCount 1];
replicatefirstoffset.FunctionArgs.Matrix.FunctionInstance='GetFirstOffset';
replicatefirstoffset.FunctionArgs.Matrix.OutputArg='ArrayVal';
functions_list=addToFunctionChain(functions_list,replicatefirstoffset);

subtractfirstoffset.InstanceName='SubtractFirstOffset';
subtractfirstoffset.FunctionHandle=@subtractFunction;
subtractfirstoffset.FunctionArgs.Number2.FunctionInstance='ReplicateFirstOffset';
subtractfirstoffset.FunctionArgs.Number2.OutputArg='Matrix';
subtractfirstoffset.FunctionArgs.Number1.FunctionInstance='ProcessingLoop';
subtractfirstoffset.FunctionArgs.Number1.OutputArg='SetOffset_Array';
functions_list=addToFunctionChain(functions_list,subtractfirstoffset);

saveoffsets.InstanceName='SaveOffsets';
saveoffsets.FunctionHandle=@saveOffsets;
saveoffsets.FunctionArgs.FileName.Value=[ImageFolder '/xyoffsets.mat'];
saveoffsets.FunctionArgs.XYOffsets.FunctionInstance='SubtractFirstOffset';
saveoffsets.FunctionArgs.XYOffsets.OutputArg='Difference';
functions_list=addToFunctionChain(functions_list,saveoffsets);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayOffsetFrames.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayOffsetFrames()
%Usage  This module is used to offset and crop frames to remove incorrect offsets
% due to errors in automatic stage return. Use with offset data acquired using
%an  assay such as assayGetStageOffset.  The original images will be untouched and
%the cropped  images will be saved in a ÿcropped framesÿ directory under the
%original image folder.   Script Variables  ImageFolder - String variable that specifies
%the absolute location of  the directory which contains the time-lapse images. An example
%of such a string variable  would be ÿc:/sample images/high-densityÿ.  ImageFilesRoot - String
%variable specifying the root image file  name. The root image file name for
%a set of images is the image  file name of any of the
%images without the number or the file extension.  For example, if the file
%name is 'Experiment-0002_Position(8)_t021.tif' the root image file name will  be 'Experiment-
0002_Position(8)_t'.  ImageFileBase
%ÿ String variable specifying the path name to the images.  This value is
%generated from the ImageFolder and the ImageFilesRoot and should not be  changed.
%CroppedImagesFolder ÿ String variable specifying the folder where the cropped images will  be
%saved. Set by default to a folder named ÿcroppedÿ within the original images
%folder.  ImageExtension ÿ String variable specifying the image file extension including the
preceding
% dot. For example, if the file name is ÿimage003.jpgÿ the image extension is
%'.jpg'.   NumberFormat ÿ String value specifying the number of digits in the
%image file  names in the sequence. For example, if the image file name
%is ÿimage020.jpgÿ the  value for the NumberFormat is '%03d', while if the file
%name is ÿimage000020.jpgÿ the  value should be ÿ%06dÿ.  StartFrame ÿ Number specifying
%the first image in the  sequence to be analyzed. The minimum value for
%this variable depends on the numbering  of the image sequence, so if the
%first image in the sequence is ÿimage003.tifÿ  then the minimum value is 3.
% FrameStep - Number specifying the step size  when reading images. Set this
%variable to 1 to read every image in the  sequence, 2 to read
%every other image and so on.  FrameCount ÿ Number  specifying how many
%images from the image sequence should be processed.   Important  Module -
%None.

global functions_list;
functions_list=[];
%script variables
ImagesFolder='C:/peter/cropped';
ImageFilesRoot='peter';
ImageFileBase=[ImagesFolder '/' ImageFilesRoot];
CroppedImagesFolder=[ImagesFolder '/cropped frames'];
ImageExtension='.tif';
NumberFormat='%06d';
StartFrame=1;
```

```
FrameStep=1;
FrameCount=10;
%end script variables

iffirstframeelsefunctions=[];
if_first_frame_functions=[];
image_read_loop_functions=[];

makedirectory.InstanceName='MakeDirectory';
makedirectory.FunctionHandle=@mkdir_Wrapper;
makedirectory.FunctionArgs.DirectoryName.Value=CroppedImagesFolder;
functions_list=addToFunctionChain(functions_list,makedirectory);

displaycurframe.InstanceName='DisplayCurFrame';
displaycurframe.FunctionHandle=@displayVariable;
displaycurframe.FunctionArgs.VariableName.Value='Current Tracking Frame';
displaycurframe.FunctionArgs.Variable.FunctionInstance='ProcessingLoop';
displaycurframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,displaycurframe);

makeimagenamesinprocessingloop.InstanceName='MakeImageNamesInProcessingLoop';
makeimagenamesinprocessingloop.FunctionHandle=@makeImgFileName;
makeimagenamesinprocessingloop.FunctionArgs.FileBase.Value=ImageFileBase;
makeimagenamesinprocessingloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinprocessingloop.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenamesinprocessingloop.FunctionArgs.CurFrame.FunctionInstance='ProcessingLoop';
makeimagenamesinprocessingloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makeimagenamesinprocessing
loop);

readimagesinprocessingloop.InstanceName='ReadImagesInProcessingLoop';
readimagesinprocessingloop.FunctionHandle=@readImage;
readimagesinprocessingloop.FunctionArgs.ImageChannel.Value='';
readimagesinprocessingloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInProcessingLoo
p';
readimagesinprocessingloop.FunctionArgs.ImageName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,readimagesinprocessingloop
);

loadoffsets.InstanceName='LoadOffsets';
loadoffsets.FunctionHandle=@loadOffsets;
loadoffsets.FunctionArgs.FileName.Value=[ImagesFolder '/xyoffsets.mat'];
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,loadoffsets);

isfirstframe.InstanceName='IsFirstFrame';
isfirstframe.FunctionHandle=@isEqualFunction;
isfirstframe.FunctionArgs.Var2.Value=StartFrame;
isfirstframe.FunctionArgs.Var1.FunctionInstance='ProcessingLoop';
isfirstframe.FunctionArgs.Var1.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,isfirstframe);

calculatecropsize.InstanceName='CalculateCropSize';
calculatecropsize.FunctionHandle=@calculateCropSize;
calculatecropsize.FunctionArgs.Image.FunctionInstance='IfFirstFrame';
calculatecropsize.FunctionArgs.Image.InputArg='ReadImagesInProcessingLoop_Image';
calculatecropsize.FunctionArgs.XYOffsets.FunctionInstance='IfFirstFrame';
calculatecropsize.FunctionArgs.XYOffsets.InputArg='LoadOffsets_Offsets';
if_first_frame_functions=addToFunctionChain(if_first_frame_functions,calculatecropsize);

iffirstframe.InstanceName='IfFirstFrame';
iffirstframe.FunctionHandle=@if_statement;
iffirstframe.FunctionArgs.TestVariable.FunctionInstance='IsFirstFrame';
iffirstframe.FunctionArgs.TestVariable.OutputArg='Boolean';
iffirstframe.FunctionArgs.ReadImagesInProcessingLoop_Image.FunctionInstance='ReadImagesInProcessi
ngLoop';
iffirstframe.FunctionArgs.ReadImagesInProcessingLoop_Image.OutputArg='Image';
iffirstframe.FunctionArgs.LoadOffsets_Offsets.FunctionInstance='LoadOffsets';
iffirstframe.FunctionArgs.LoadOffsets_Offsets.OutputArg='Offsets';
iffirstframe.KeepValues.CalculateCropSize_CropSize.FunctionInstance='CalculateCropSize';
iffirstframe.KeepValues.CalculateCropSize_CropSize.OutputArg='CropSize';
iffirstframe.ElseFunctions=iffirstframeelsefunctions;
iffirstframe.IfFunctions=if_first_frame_functions;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,iffirstframe);

getcurrentoffset.InstanceName='GetCurrentOffset';
getcurrentoffset.FunctionHandle=@getArrayVal;
getcurrentoffset.FunctionArgs.Array.FunctionInstance='LoadOffsets';
getcurrentoffset.FunctionArgs.Array.OutputArg='Offsets';
getcurrentoffset.FunctionArgs.Index.FunctionInstance='ProcessingLoop';
getcurrentoffset.FunctionArgs.Index.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getcurrentoffset);
```

```
cropimage.InstanceName='CropImage';
cropimage.FunctionHandle=@cropImageWithOffset;
cropimage.FunctionArgs.Image.FunctionInstance='ReadImagesInProcessingLoop';
cropimage.FunctionArgs.Image.OutputArg='Image';
cropimage.FunctionArgs.XYOffset.FunctionInstance='GetCurrentOffset';
cropimage.FunctionArgs.XYOffset.OutputArg='ArrayVal';
cropimage.FunctionArgs.CropSize.FunctionInstance='IfFirstFrame';
cropimage.FunctionArgs.CropSize.OutputArg='CalculateCropSize_CropSize';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,cropimage);

makeoutputimagenames.InstanceName='MakeOutputImageNames';
makeoutputimagenames.FunctionHandle=@makeImgFileName;
makeoutputimagenames.FunctionArgs.FileBase.Value=[CroppedImagesFolder '/' ImageFilesRoot];
makeoutputimagenames.FunctionArgs.NumberFmt.Value=NumberFormat;
makeoutputimagenames.FunctionArgs.FileExt.Value=ImageExtension;
makeoutputimagenames.FunctionArgs.CurFrame.FunctionInstance='ProcessingLoop';
makeoutputimagenames.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makeoutputimagenames);

saveimage.InstanceName='SaveImage';
saveimage.FunctionHandle=@imwrite_Wrapper;
saveimage.FunctionArgs.Format.Value=ImageExtension(2:end);
saveimage.FunctionArgs.Image.FunctionInstance='CropImage';
saveimage.FunctionArgs.Image.OutputArg='Image';
saveimage.FunctionArgs.FileName.FunctionInstance='MakeOutputImageNames';
saveimage.FunctionArgs.FileName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,saveimage);

processingloop.InstanceName='ProcessingLoop';
processingloop.FunctionHandle=@forLoop;
processingloop.FunctionArgs.StartLoop.Value=StartFrame;
processingloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
processingloop.FunctionArgs.IncrementLoop.Value=FrameStep;
processingloop.LoopFunctions=image_read_loop_functions;
functions_list=addToFunctionChain(functions_list,processingloop);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayOverlayCellProfilerData.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayOverlayCellProfilerData()

global functions_list;
functions_list=[];
%script variables
ImageFolder='C:\darren\movie2';
ImageRoot='DsRed - Confocal - n';
ImageExtension='.tif';
NumberFormat='%06d';
StartFrame=1;
FrameCount=72;
CPFileName='C:\Users\hubertus\Documents\CellProfiler\DefaultOUT.mat';
OutputFolder=[ImageFolder '/cellprofiler output'];
%end script variables

OverlayLoopLoopFunctions=[];

loadcpdata.InstanceName='LoadCPData';
loadcpdata.FunctionHandle=@loadCellProfilerTracks;
loadcpdata.FunctionArgs.CPDataFile.Value=CPFileName;
functions_list=addToFunctionChain(functions_list,loadcpdata);

makeoutputfolder.InstanceName='MakeOutputFolder';
makeoutputfolder.FunctionHandle=@mkdir_Wrapper;
makeoutputfolder.FunctionArgs.DirectoryName.Value=OutputFolder;
functions_list=addToFunctionChain(functions_list,makeoutputfolder);

displaycurframe.InstanceName='DisplayCurFrame';
displaycurframe.FunctionHandle=@displayVariable;
displaycurframe.FunctionArgs.VariableName.Value='CurFrame';
displaycurframe.FunctionArgs.Variable.FunctionInstance='OverlayLoop';
```

```
displaycurframe.FunctionArgs.Variable.OutputArg='LoopCounter';
OverlayLoopLoopFunctions=addToFunctionChain(OverlayLoopLoopFunctions,displaycurframe);

makeimagenames.InstanceName='MakeImageNames';
makeimagenames.FunctionHandle=@makeImgFileName;
makeimagenames.FunctionArgs.FileBase.Value=[ImageFolder '/' ImageRoot];
makeimagenames.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenames.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenames.FunctionArgs.CurFrame.FunctionInstance='OverlayLoop';
makeimagenames.FunctionArgs.CurFrame.OutputArg='LoopCounter';
OverlayLoopLoopFunctions=addToFunctionChain(OverlayLoopLoopFunctions,makeimagenames);

readimage.InstanceName='ReadImage';
readimage.FunctionHandle=@readImage;
readimage.FunctionArgs.ImageChannel.Value='';
readimage.FunctionArgs.ImageName.FunctionInstance='MakeImageNames';
readimage.FunctionArgs.ImageName.OutputArg='FileName';
OverlayLoopLoopFunctions=addToFunctionChain(OverlayLoopLoopFunctions,readimage);

adjustcontrast.InstanceName='AdjustContrast';
adjustcontrast.FunctionHandle=@imadjust_Wrapper;
adjustcontrast.FunctionArgs.Image.FunctionInstance='ReadImage';
adjustcontrast.FunctionArgs.Image.OutputArg='Image';
OverlayLoopLoopFunctions=addToFunctionChain(OverlayLoopLoopFunctions,adjustcontrast);

getcurrenttracks.InstanceName='GetCurrentTracks';
getcurrenttracks.FunctionHandle=@getCurrentTracks;
getcurrenttracks.FunctionArgs.FrameStep.Value=1;
getcurrenttracks.FunctionArgs.MaxMissingFrames.Value=1;
getcurrenttracks.FunctionArgs.OffsetFrame.Value=0;
getcurrenttracks.FunctionArgs.TimeCol.Value=2;
getcurrenttracks.FunctionArgs.TimeFrame.Value=1;
getcurrenttracks.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks.FunctionArgs.CurFrame.FunctionInstance='OverlayLoop';
getcurrenttracks.FunctionArgs.CurFrame.OutputArg='LoopCounter';
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance='OverlayLoop';
getcurrenttracks.FunctionArgs.Tracks.InputArg='LoadCPData_Tracks';
OverlayLoopLoopFunctions=addToFunctionChain(OverlayLoopLoopFunctions,getcurrenttracks);

overlayids.InstanceName='OverlayIDs';
overlayids.FunctionHandle=@displayTracksData;
overlayids.FunctionArgs.FileRoot.Value=[OutputFolder '/' ImageRoot];
overlayids.FunctionArgs.NumberFormat.Value=NumberFormat;
overlayids.FunctionArgs.ShowIDs.Value=true;
overlayids.FunctionArgs.LabelColorRGB.Value=[255 0 0];
overlayids.FunctionArgs.CurFrame.FunctionInstance='OverlayLoop';
overlayids.FunctionArgs.CurFrame.OutputArg='LoopCounter';
overlayids.FunctionArgs.CurrentTracks.FunctionInstance='GetCurrentTracks';
overlayids.FunctionArgs.CurrentTracks.OutputArg='Tracks';
overlayids.FunctionArgs.Image.FunctionInstance='AdjustContrast';
overlayids.FunctionArgs.Image.OutputArg='Image';
overlayids.FunctionArgs.TracksLayout.FunctionInstance='OverlayLoop';
overlayids.FunctionArgs.TracksLayout.InputArg='LoadCPData_TracksLayout';
OverlayLoopLoopFunctions=addToFunctionChain(OverlayLoopLoopFunctions,overlayids);

overlayloop.InstanceName='OverlayLoop';
overlayloop.FunctionHandle=@forLoop;
overlayloop.FunctionArgs.EndLoop.Value=StartFrame+FrameCount-1;
overlayloop.FunctionArgs.IncrementLoop.Value=1;
overlayloop.FunctionArgs.StartLoop.Value=StartFrame;
overlayloop.FunctionArgs.LoadCPData_Tracks.FunctionInstance='LoadCPData';
overlayloop.FunctionArgs.LoadCPData_Tracks.OutputArg='Tracks';
overlayloop.FunctionArgs.LoadCPData_TracksLayout.FunctionInstance='LoadCPData';
overlayloop.FunctionArgs.LoadCPData_TracksLayout.OutputArg='TracksLayout';
overlayloop.LoopFunctions=OverlayLoopLoopFunctions;
functions_list=addToFunctionChain(functions_list,overlayloop);


global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assayTestOnlyNNWG.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function []=assayTestOnlyNNWG()
```

```
%This assay is only to test a simple NN algorithm. Don't use it as it's
% not applying any of the usual track refinements and it will not perform
%well.

global functions_list;
functions_list=[];
%script variables
ImageFolder='C:/darren/movie2';
ImageFilesRoot='DsRed - Confocal - n';
ImageExtension='.tif';
StartFrame=1;
FrameCount=72;
TimeFrame=15;
FrameStep=1;
NumberFormat='%06d';
MaxFramesMissing=6;
OutputFolder=[ImageFolder '/output'];
AncestryFolder=[OutputFolder '/ancestry'];
AncestrySpreadsheet=[AncestryFolder 'ancestry.csv'];
ShapesSpreadsheet=[AncestryFolder 'shapes.csv'];
TracksFolder=[OutputFolder '/track'];
SegmentationFilesRoot=[TracksFolder '/grayscale'];
ImageFileBase=[ImageFolder '/' ImageFilesRoot];
BrightnessThresholdPct=1.1;
ObjectArea=30;
Strel='disk';
StrelSize=10;
ClearBorder=true;
ClearBorderDist=2;
MedianFilterSize=3;
MinSolidity=0.69;
MinAreaOverPerimeter=1.5;
ResizeImageScale=0.5;
ApproximationDistance=2.4;
MaxSearchRadius=Inf;
MinSearchRadius=0;
MinSecondDistance=5;
MaxDistRatio=0.6;
MaxAngleDiff=0.35;
NrParamsForSureMatch=5;
SearchRadiusPct=1.5;
MaxMergeDistance=23;
MaxSplitArea=400;
MaxSplitDistance=45;
MinSplitEccentricity=0.5;
MaxSplitEccentricity=0.95;
MinTimeForSplit=900;
MinLifespan=30;
FrontParams=[];
DefaultParamWeights=[34 21 13 8 5 3 2 2 2];
DistanceRankingOrder=[1 3 4 5 6 7 8 9 2];
DirectionRankingOrder=[2 3 4 5 6 7 8 9 1];
RelevantParametersIndex=[true true true false true false true true false];
UnknownParamWeights=[5 3 1 1 1 1 1 1 1];
UnknownRankingOrder=[1 2 3 4 5 6 7 8 9];
%end script variables

assign_cells_to_tracks_functions=[];
else_is_empty_cells_label_functions=[];
if_is_empty_cells_label_functions=[];
image_read_loop_functions=[];

loadtrackslayout.InstanceName='LoadTracksLayout';
loadtrackslayout.FunctionHandle=@loadTracksLayout;
loadtrackslayout.FunctionArgs.FileName.Value='tracks_layout.mat';
functions_list=addToFunctionChain(functions_list,loadtrackslayout);

loadancestrylayout.InstanceName='LoadAncestryLayout';
loadancestrylayout.FunctionHandle=@loadAncestryLayout;
loadancestrylayout.FunctionArgs.FileName.Value='ancestry_layout.mat';
functions_list=addToFunctionChain(functions_list,loadancestrylayout);

displaycurframe.InstanceName='DisplayCurFrame';
displaycurframe.FunctionHandle=@displayVariable;
displaycurframe.FunctionArgs.VariableName.Value='Current Tracking Frame';
displaycurframe.FunctionArgs.Variable.FunctionInstance='SegmentationLoop';
displaycurframe.FunctionArgs.Variable.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,displaycurframe);

makeimagenamesinsegmentationloop.InstanceName='MakeImageNamesInSegmentationLoop';
makeimagenamesinsegmentationloop.FunctionHandle=@makeImgFileName;
```

```
makeimagenamesinsegmentationloop.FunctionArgs.FileBase.Value=ImageFileBase;
makeimagenamesinsegmentationloop.FunctionArgs.NumberFmt.Value=NumberFormat;
makeimagenamesinsegmentationloop.FunctionArgs.FileExt.Value=ImageExtension;
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
makeimagenamesinsegmentationloop.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makeimagenamesinsegmentati
onloop);

readimagesinsegmentationloop.InstanceName='ReadImagesInSegmentationLoop';
readimagesinsegmentationloop.FunctionHandle=@readImage;
readimagesinsegmentationloop.FunctionArgs.ImageChannel.Value='';
readimagesinsegmentationloop.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesInSegmentatio
nLoop';
readimagesinsegmentationloop.FunctionArgs.ImageName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,readimagesinsegmentationlo
op);

normalizeimageto16bit.InstanceName='NormalizeImageTo16Bit';
normalizeimageto16bit.FunctionHandle=@imNorm;
normalizeimageto16bit.FunctionArgs.IntegerClass.Value='uint16';
normalizeimageto16bit.FunctionArgs.RawImage.FunctionInstance='ReadImagesInSegmentationLoop';
normalizeimageto16bit.FunctionArgs.RawImage.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,normalizeimageto16bit);

resizeimage.InstanceName='ResizeImage';
resizeimage.FunctionHandle=@resizeImage;
resizeimage.FunctionArgs.Scale.Value=ResizeImageScale;
resizeimage.FunctionArgs.Method.Value='bicubic';
resizeimage.FunctionArgs.Image.FunctionInstance='NormalizeImageTo16Bit';
resizeimage.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,resizeimage);

cytobrightnesslocalaveragingfilter.InstanceName='CytoBrightnessLocalAveragingFilter';
cytobrightnesslocalaveragingfilter.FunctionHandle=@generateBinImgUsingLocAvg;
cytobrightnesslocalaveragingfilter.FunctionArgs.Strel.Value=Strel;
cytobrightnesslocalaveragingfilter.FunctionArgs.StrelSize.Value=StrelSize;
cytobrightnesslocalaveragingfilter.FunctionArgs.BrightnessThresholdPct.Value=BrightnessThresholdP
ct;
cytobrightnesslocalaveragingfilter.FunctionArgs.ClearBorder.Value=ClearBorder;
cytobrightnesslocalaveragingfilter.FunctionArgs.ClearBorderDist.Value=ClearBorderDist;
cytobrightnesslocalaveragingfilter.FunctionArgs.Image.FunctionInstance='ResizeImage';
cytobrightnesslocalaveragingfilter.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,cytobrightnesslocalaveragi
ngfilter);

fillholescytoplasmimages.InstanceName='FillHolesCytoplasmImages';
fillholescytoplasmimages.FunctionHandle=@fillHoles;
fillholescytoplasmimages.FunctionArgs.Image.FunctionInstance='CytoBrightnessLocalAveragingFilter'
;
fillholescytoplasmimages.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,fillholescytoplasmimages);

clearsmallcells.InstanceName='ClearSmallCells';
clearsmallcells.FunctionHandle=@clearSmallObjects;
clearsmallcells.FunctionArgs.MinObjectArea.Value=ObjectArea;
clearsmallcells.FunctionArgs.Image.FunctionInstance='FillHolesCytoplasmImages';
clearsmallcells.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,clearsmallcells);

nuclbrightnesslocalaveragingfilter.InstanceName='NuclBrightnessLocalAveragingFilter';
nuclbrightnesslocalaveragingfilter.FunctionHandle=@generateBinImgUsingLocAvg;
nuclbrightnesslocalaveragingfilter.FunctionArgs.Strel.Value=Strel;
nuclbrightnesslocalaveragingfilter.FunctionArgs.StrelSize.Value=StrelSize;
nuclbrightnesslocalaveragingfilter.FunctionArgs.BrightnessThresholdPct.Value=BrightnessThresholdP
ct;
nuclbrightnesslocalaveragingfilter.FunctionArgs.ClearBorder.Value=ClearBorder;
nuclbrightnesslocalaveragingfilter.FunctionArgs.ClearBorderDist.Value=ClearBorderDist;
nuclbrightnesslocalaveragingfilter.FunctionArgs.Image.FunctionInstance='ResizeImage';
nuclbrightnesslocalaveragingfilter.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,nuclbrightnesslocalaveragi
ngfilter);

fillholesnuclearimages.InstanceName='FillHolesNuclearImages';
fillholesnuclearimages.FunctionHandle=@fillHoles;
fillholesnuclearimages.FunctionArgs.Image.FunctionInstance='NuclBrightnessLocalAveragingFilter';
fillholesnuclearimages.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,fillholesnuclearimages);

clearsmallnuclei.InstanceName='ClearSmallNuclei';
clearsmallnuclei.FunctionHandle=@clearSmallObjects;
clearsmallnuclei.FunctionArgs.MinObjectArea.Value=ObjectArea;
```

```
clearsmallnuclei.FunctionArgs.Image.FunctionInstance='FillHolesNuclearImages';
clearsmallnuclei.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,clearsmallnuclei);

combinenuclearandcytoplasmimages.InstanceName='CombineNuclearAndCytoplasmImages';
combinenuclearandcytoplasmimages.FunctionHandle=@combineImages;
combinenuclearandcytoplasmimages.FunctionArgs.CombineOperation.Value='AND';
combinenuclearandcytoplasmimages.FunctionArgs.Image1.FunctionInstance='ClearSmallNuclei';
combinenuclearandcytoplasmimages.FunctionArgs.Image1.OutputArg='Image';
combinenuclearandcytoplasmimages.FunctionArgs.Image2.FunctionInstance='ClearSmallCells';
combinenuclearandcytoplasmimages.FunctionArgs.Image2.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,combinenuclearandcytoplasm
images);

reconstructcytoplasmimage.InstanceName='ReconstructCytoplasmImage';
reconstructcytoplasmimage.FunctionHandle=@reconstructObjects;
reconstructcytoplasmimage.FunctionArgs.GuideImage.FunctionInstance='CombineNuclearAndCytoplasmIma
ges';
reconstructcytoplasmimage.FunctionArgs.GuideImage.OutputArg='Image';
reconstructcytoplasmimage.FunctionArgs.ImageToReconstruct.FunctionInstance='ClearSmallNuclei';
reconstructcytoplasmimage.FunctionArgs.ImageToReconstruct.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,reconstructcytoplasmimage)
;

labelnuclei.InstanceName='LabelNuclei';
labelnuclei.FunctionHandle=@labelObjects;
labelnuclei.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
labelnuclei.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,labelnuclei);

labelcytoplasm.InstanceName='LabelCytoplasm';
labelcytoplasm.FunctionHandle=@labelObjects;
labelcytoplasm.FunctionArgs.Image.FunctionInstance='ReconstructCytoplasmImage';
labelcytoplasm.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,labelcytoplasm);

getconvexobjects.InstanceName='GetConvexObjects';
getconvexobjects.FunctionHandle=@getConvexObjects;
getconvexobjects.FunctionArgs.ApproximationDistance.Value=ApproximationDistance;
getconvexobjects.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
getconvexobjects.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getconvexobjects);

distancewatershed.InstanceName='DistanceWatershed';
distancewatershed.FunctionHandle=@distanceWatershed;
distancewatershed.FunctionArgs.MedianFilterNhood.Value=MedianFilterSize;
distancewatershed.FunctionArgs.Image.FunctionInstance='ClearSmallNuclei';
distancewatershed.FunctionArgs.Image.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,distancewatershed);

polygonalassistedwatershed.InstanceName='PolygonalAssistedWatershed';
polygonalassistedwatershed.FunctionHandle=@polygonalAssistedWatershed;
polygonalassistedwatershed.FunctionArgs.MinBlobArea.Value=ObjectArea;
polygonalassistedwatershed.FunctionArgs.ImageLabel.FunctionInstance='LabelNuclei';
polygonalassistedwatershed.FunctionArgs.ImageLabel.OutputArg='LabelMatrix';
polygonalassistedwatershed.FunctionArgs.WatershedLabel.FunctionInstance='DistanceWatershed';
polygonalassistedwatershed.FunctionArgs.WatershedLabel.OutputArg='LabelMatrix';
polygonalassistedwatershed.FunctionArgs.ConvexObjectsIndex.FunctionInstance='GetConvexObjects';
polygonalassistedwatershed.FunctionArgs.ConvexObjectsIndex.OutputArg='ConvexObjectsIndex';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,polygonalassistedwatershed
);

segmentobjectsusingmarkers.InstanceName='SegmentObjectsUsingMarkers';
segmentobjectsusingmarkers.FunctionHandle=@segmentObjectsUsingMarkers;
segmentobjectsusingmarkers.FunctionArgs.MarkersLabel.FunctionInstance='PolygonalAssistedWatershed
';
segmentobjectsusingmarkers.FunctionArgs.MarkersLabel.OutputArg='LabelMatrix';
segmentobjectsusingmarkers.FunctionArgs.ObjectsLabel.FunctionInstance='LabelCytoplasm';
segmentobjectsusingmarkers.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,segmentobjectsusingmarkers
);

areafilter.InstanceName='AreaFilter';
areafilter.FunctionHandle=@areaFilterLabel;
areafilter.FunctionArgs.MinArea.Value=ObjectArea;
areafilter.FunctionArgs.ObjectsLabel.FunctionInstance='SegmentObjectsUsingMarkers';
areafilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,areafilter);

solidityfilter.InstanceName='SolidityFilter';
solidityfilter.FunctionHandle=@solidityFilterLabel;
```

```
solidityfilter.FunctionArgs.MinSolidity.Value=MinSolidity;
solidityfilter.FunctionArgs.ObjectsLabel.FunctionInstance='AreaFilter';
solidityfilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,solidityfilter);

aoverpfilter.InstanceName='AOverPFilter';
aoverpfilter.FunctionHandle=@areaOverPerimeterFilterLabel;
aoverpfilter.FunctionArgs.MinAreaOverPerimeter.Value=MinAreaOverPerimeter;
aoverpfilter.FunctionArgs.ObjectsLabel.FunctionInstance='SolidityFilter';
aoverpfilter.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,aoverpfilter);

resizecytolabel.InstanceName='ResizeCytoLabel';
resizecytolabel.FunctionHandle=@resizeImage;
resizecytolabel.FunctionArgs.Scale.Value=1/ResizeImageScale;
resizecytolabel.FunctionArgs.Method.Value='nearest';
resizecytolabel.FunctionArgs.Image.FunctionInstance='AOverPFilter';
resizecytolabel.FunctionArgs.Image.OutputArg='LabelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,resizecytolabel);

makeexportfilenames.InstanceName='MakeExportFileNames';
makeexportfilenames.FunctionHandle=@makeImgFileName;
makeexportfilenames.FunctionArgs.FileBase.Value='C:\darren\movie2\cellanimation output\exported
labels\CALabel';
makeexportfilenames.FunctionArgs.FileExt.Value='.tif';
makeexportfilenames.FunctionArgs.NumberFmt.Value=NumberFormat;
makeexportfilenames.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
makeexportfilenames.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,makeexportfilenames);

exportlabeltoimage.InstanceName='ExportLabelToImage';
exportlabeltoimage.FunctionHandle=@exportLabelToImage;
exportlabeltoimage.FunctionArgs.Format.Value='tif';
exportlabeltoimage.FunctionArgs.Image.FunctionInstance='ResizeCytoLabel';
exportlabeltoimage.FunctionArgs.Image.OutputArg='Image';
exportlabeltoimage.FunctionArgs.FileName.FunctionInstance='MakeExportFileNames';
exportlabeltoimage.FunctionArgs.FileName.OutputArg='FileName';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,exportlabeltoimage);

getshapeparameters.InstanceName='GetShapeParameters';
getshapeparameters.FunctionHandle=@getShapeParams;
getshapeparameters.FunctionArgs.LabelMatrix.FunctionInstance='ResizeCytoLabel';
getshapeparameters.FunctionArgs.LabelMatrix.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,getshapeparameters);

isemptypreviouscellslabel.InstanceName='IsEmptyPreviousCellsLabel';
isemptypreviouscellslabel.FunctionHandle=@isEmptyFunction;
isemptypreviouscellslabel.FunctionArgs.TestVariable.Value=[];
isemptypreviouscellslabel.FunctionArgs.TestVariable.FunctionInstance='SaveCellsLabel';
isemptypreviouscellslabel.FunctionArgs.TestVariable.OutputArg='CellsLabel';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,isemptypreviouscellslabel)
;

starttracks.InstanceName='StartTracks';
starttracks.FunctionHandle=@startTracks;
starttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
starttracks.FunctionArgs.CellsLabel.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.CellsLabel.InputArg='ResizeCytoLabel_Image';
starttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
starttracks.FunctionArgs.ShapeParameters.FunctionInstance='IfIsEmptyPreviousCellsLabel';
starttracks.FunctionArgs.ShapeParameters.InputArg='GetShapeParameters_ShapeParameters';
if_is_empty_cells_label_functions=addToFunctionChain(if_is_empty_cells_label_functions,starttrack
s);

displayinitialtracks.InstanceName='DisplayInitialTracks';
displayinitialtracks.FunctionHandle=@displayTracksData;
displayinitialtracks.FunctionArgs.LabelColorRGB.Value=[255 140 0];
displayinitialtracks.FunctionArgs.NumberFormat.Value=NumberFormat;
displayinitialtracks.FunctionArgs.ShowIDs.Value=true;
displayinitialtracks.FunctionArgs.FileRoot.Value=[TracksFolder '/' ImageFilesRoot];
displayinitialtracks.FunctionArgs.CurrentTracks.FunctionInstance='StartTracks';
displayinitialtracks.FunctionArgs.CurrentTracks.OutputArg='Tracks';
displayinitialtracks.FunctionArgs.CellsLabel.FunctionInstance='IfIsEmptyPreviousCellsLabel';
displayinitialtracks.FunctionArgs.CellsLabel.InputArg='ResizeCytoLabel_Image';
displayinitialtracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
displayinitialtracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
displayinitialtracks.FunctionArgs.Image.FunctionInstance='IfIsEmptyPreviousCellsLabel';
displayinitialtracks.FunctionArgs.Image.InputArg='NormalizeImageTo16Bit_Image';
displayinitialtracks.FunctionArgs.TracksLayout.FunctionInstance='IfIsEmptyPreviousCellsLabel';
displayinitialtracks.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
```

```
if_is_empty_cells_label_functions=addToFunctionChain(if_is_empty_cells_label_functions,displayini
tialtracks);

getcurrenttracks.InstanceName='GetCurrentTracks';
getcurrenttracks.FunctionHandle=@getCurrentTracks;
getcurrenttracks.FunctionArgs.OffsetFrame.Value=-1;
getcurrenttracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getcurrenttracks.FunctionArgs.TimeCol.Value=2;
getcurrenttracks.FunctionArgs.TrackIDCol.Value=1;
getcurrenttracks.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getcurrenttracks.FunctionArgs.FrameStep.Value=FrameStep;
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg='Tracks';
getcurrenttracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getcurrenttracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getcurrenttracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getcurrenttracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getcur
renttracks);

getprevioustracks.InstanceName='GetPreviousTracks';
getprevioustracks.FunctionHandle=@getCurrentTracks;
getprevioustracks.FunctionArgs.OffsetFrame.Value=-2;
getprevioustracks.FunctionArgs.TimeFrame.Value=TimeFrame;
getprevioustracks.FunctionArgs.TimeCol.Value=2;
getprevioustracks.FunctionArgs.TrackIDCol.Value=1;
getprevioustracks.FunctionArgs.MaxMissingFrames.Value=MaxFramesMissing;
getprevioustracks.FunctionArgs.FrameStep.Value=FrameStep;
getprevioustracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg='Tracks';
getprevioustracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getprevioustracks.FunctionArgs.Tracks.OutputArg2='Tracks';
getprevioustracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
getprevioustracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getpre
vioustracks);

makeunassignedcellslist.InstanceName='MakeUnassignedCellsList';
makeunassignedcellslist.FunctionHandle=@makeUnassignedCellsList;
makeunassignedcellslist.FunctionArgs.CellsCentroids.FunctionInstance='IfIsEmptyPreviousCellsLabel
';
makeunassignedcellslist.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,makeun
assignedcellslist);

makeexcludedtrackslist.InstanceName='MakeExcludedTracksList';
makeexcludedtrackslist.FunctionHandle=@makeExcludedTracksList;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.FunctionInstance='MakeUnassignedCellsList'
;
makeexcludedtrackslist.FunctionArgs.UnassignedCellsIDs.OutputArg='UnassignedCellsIDs';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,makeex
cludedtrackslist);

getmaxtrackid.InstanceName='GetMaxTrackID';
getmaxtrackid.FunctionHandle=@getMaxTrackID;
getmaxtrackid.FunctionArgs.TrackIDCol.Value=1;
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance='StartTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg='Tracks';
getmaxtrackid.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
getmaxtrackid.FunctionArgs.Tracks.OutputArg2='Tracks';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,getmax
trackid);

isnotemptyunassignedcells.InstanceName='IsNotEmptyUnassignedCells';
isnotemptyunassignedcells.FunctionHandle=@isNotEmptyFunction;
isnotemptyunassignedcells.FunctionArgs.TestVariable.FunctionInstance='AssignCellToTrackUsingAll';
isnotemptyunassignedcells.FunctionArgs.TestVariable.OutputArg='UnassignedIDs';
isnotemptyunassignedcells.FunctionArgs.TestVariable.FunctionInstance2='AssignCellsToTracksLoop';
isnotemptyunassignedcells.FunctionArgs.TestVariable.InputArg2='MakeUnassignedCellsList_Unassigned
CellsIDs';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,isnotemptyun
assignedcells);

getcurrentunassignedcell.InstanceName='GetCurrentUnassignedCell';
getcurrentunassignedcell.FunctionHandle=@getCurrentUnassignedCell;
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance='AssignCellToTrackUsingAll
';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
getcurrentunassignedcell.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTracksLoop'
;
```

```
getcurrentunassignedcell.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassign
edCellsIDs';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,getcurrentun
assignedcell);

assigncelltotrackusingall.InstanceName='AssignCellToTrackUsingAll';
assigncelltotrackusingall.FunctionHandle=@assignCellToTrackUsingNN;
assigncelltotrackusingall.FunctionArgs.TrackAssignments.Value=[];
assigncelltotrackusingall.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellToTrackUsingA
ll';
assigncelltotrackusingall.FunctionArgs.TrackAssignments.OutputArg='TrackAssignments';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.FunctionInstance='AssignCellToTrackUsingAl
l';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.OutputArg='UnassignedIDs';
assigncelltotrackusingall.FunctionArgs.CellsCentroids.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
assigncelltotrackusingall.FunctionArgs.CurrentTracks.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.CurrentTracks.InputArg='GetCurrentTracks_Tracks';
assigncelltotrackusingall.FunctionArgs.MaxTrackID.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.MaxTrackID.InputArg='GetMaxTrackID_MaxTrackID';
assigncelltotrackusingall.FunctionArgs.TracksLayout.FunctionInstance='AssignCellsToTracksLoop';
assigncelltotrackusingall.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.FunctionInstance2='AssignCellsToTracksLoop
';
assigncelltotrackusingall.FunctionArgs.UnassignedCells.InputArg2='MakeUnassignedCellsList_Unassig
nedCellsIDs';
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_functions,assigncellto
trackusingall);

assigncellstotracksloop.InstanceName='AssignCellsToTracksLoop';
assigncellstotracksloop.FunctionHandle=@whileLoop;
assigncellstotracksloop.FunctionArgs.TestFunction.FunctionInstance='IsNotEmptyUnassignedCells';
assigncellstotracksloop.FunctionArgs.TestFunction.OutputArg='Boolean';
assigncellstotracksloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.FunctionInstance=
'MakeUnassignedCellsList';
assigncellstotracksloop.FunctionArgs.MakeUnassignedCellsList_UnassignedCellsIDs.OutputArg='Unassi
gnedCellsIDs';
assigncellstotracksloop.FunctionArgs.GetCurrentTracks_Tracks.FunctionInstance='GetCurrentTracks';
assigncellstotracksloop.FunctionArgs.GetCurrentTracks_Tracks.OutputArg='Tracks';
assigncellstotracksloop.FunctionArgs.GetMaxTrackID_MaxTrackID.FunctionInstance='GetMaxTrackID';
assigncellstotracksloop.FunctionArgs.GetMaxTrackID_MaxTrackID.OutputArg='MaxTrackID';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_Centroids.FunctionInstance='IfIsEmptyPrev
iousCellsLabel';
assigncellstotracksloop.FunctionArgs.GetShapeParameters_Centroids.InputArg='GetShapeParameters_Ce
ntroids';
assigncellstotracksloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='IfIsEmptyPre
viousCellsLabel';
assigncellstotracksloop.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout_Tra
cksLayout';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_TrackAssignments.FunctionInstance='A
ssignCellToTrackUsingAll';
assigncellstotracksloop.KeepValues.AssignCellToTrackUsingAll_TrackAssignments.OutputArg='TrackAss
ignments';
assigncellstotracksloop.LoopFunctions=assign_cells_to_tracks_functions;
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,assign
cellstotracksloop);

continuetracks.InstanceName='ContinueTracks';
continuetracks.FunctionHandle=@continueTracks;
continuetracks.FunctionArgs.TimeFrame.Value=TimeFrame;
continuetracks.FunctionArgs.Tracks.FunctionInstance='StartTracks';
continuetracks.FunctionArgs.Tracks.OutputArg='Tracks';
continuetracks.FunctionArgs.Tracks.FunctionInstance2='ContinueTracks';
continuetracks.FunctionArgs.Tracks.OutputArg2='Tracks';
continuetracks.FunctionArgs.TrackAssignments.FunctionInstance='AssignCellsToTracksLoop';
continuetracks.FunctionArgs.TrackAssignments.OutputArg='AssignCellToTrackUsingAll_TrackAssignment
s';
continuetracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
continuetracks.FunctionArgs.CellsCentroids.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.CellsCentroids.InputArg='GetShapeParameters_Centroids';
continuetracks.FunctionArgs.ShapeParameters.FunctionInstance='IfIsEmptyPreviousCellsLabel';
continuetracks.FunctionArgs.ShapeParameters.InputArg='GetShapeParameters_ShapeParameters';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,contin
uetracks);

displaytracks.InstanceName='DisplayTracks';
displaytracks.FunctionHandle=@displayTracksData;
displaytracks.FunctionArgs.LabelColorRGB.Value=[255 140 0];
displaytracks.FunctionArgs.NumberFormat.Value=NumberFormat;
displaytracks.FunctionArgs.ShowIDs.Value=true;
```

```
displaytracks.FunctionArgs.FileRoot.Value=[TracksFolder '/' ImageFilesRoot];
displaytracks.FunctionArgs.CurrentTracks.FunctionInstance='ContinueTracks';
displaytracks.FunctionArgs.CurrentTracks.OutputArg='NewTracks';
displaytracks.FunctionArgs.CellsLabel.FunctionInstance='IfIsEmptyPreviousCellsLabel';
displaytracks.FunctionArgs.CellsLabel.InputArg='ResizeCytoLabel_Image';
displaytracks.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPreviousCellsLabel';
displaytracks.FunctionArgs.CurFrame.InputArg='SegmentationLoop_LoopCounter';
displaytracks.FunctionArgs.Image.FunctionInstance='IfIsEmptyPreviousCellsLabel';
displaytracks.FunctionArgs.Image.InputArg='NormalizeImageTo16Bit_Image';
displaytracks.FunctionArgs.TracksLayout.FunctionInstance='IfIsEmptyPreviousCellsLabel';
displaytracks.FunctionArgs.TracksLayout.InputArg='LoadTracksLayout_TracksLayout';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_label_functions,displa
ytracks);

ifisemptypreviouscellslabel.InstanceName='IfIsEmptyPreviousCellsLabel';
ifisemptypreviouscellslabel.FunctionHandle=@if_statement;
ifisemptypreviouscellslabel.FunctionArgs.PreviousCellsLabel.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.MatchingGroupsStats.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.TrackAssignments.Value=[];
ifisemptypreviouscellslabel.FunctionArgs.TestVariable.FunctionInstance='IsEmptyPreviousCellsLabel
';
ifisemptypreviouscellslabel.FunctionArgs.TestVariable.OutputArg='Boolean';
ifisemptypreviouscellslabel.FunctionArgs.ResizeCytoLabel_Image.FunctionInstance='ResizeCytoLabel'
;
ifisemptypreviouscellslabel.FunctionArgs.ResizeCytoLabel_Image.OutputArg='Image';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_LoopCounter.FunctionInstance='Segmentat
ionLoop';
ifisemptypreviouscellslabel.FunctionArgs.SegmentationLoop_LoopCounter.OutputArg='LoopCounter';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_ShapeParameters.FunctionInstance='Get
ShapeParameters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_ShapeParameters.OutputArg='ShapeParam
eters';
ifisemptypreviouscellslabel.FunctionArgs.NormalizeImageTo16Bit_Image.FunctionInstance='NormalizeI
mageTo16Bit';
ifisemptypreviouscellslabel.FunctionArgs.NormalizeImageTo16Bit_Image.OutputArg='Image';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_Centroids.FunctionInstance='GetShapeP
arameters';
ifisemptypreviouscellslabel.FunctionArgs.GetShapeParameters_Centroids.OutputArg='Centroids';
ifisemptypreviouscellslabel.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='Segmenta
tionLoop';
ifisemptypreviouscellslabel.FunctionArgs.LoadTracksLayout_TracksLayout.InputArg='LoadTracksLayout
_TracksLayout';
ifisemptypreviouscellslabel.KeepValues.ContinueTracks_Tracks.FunctionInstance='ContinueTracks';
ifisemptypreviouscellslabel.KeepValues.ContinueTracks_Tracks.OutputArg='Tracks';
ifisemptypreviouscellslabel.ElseFunctions=else_is_empty_cells_label_functions;
ifisemptypreviouscellslabel.IfFunctions=if_is_empty_cells_label_functions;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,ifisemptypreviouscellslabe
l);

savecellslabel.InstanceName='SaveCellsLabel';
savecellslabel.FunctionHandle=@saveCellsLabel;
savecellslabel.FunctionArgs.FileRoot.Value=SegmentationFilesRoot;
savecellslabel.FunctionArgs.NumberFormat.Value=NumberFormat;
savecellslabel.FunctionArgs.CellsLabel.FunctionInstance='ResizeCytoLabel';
savecellslabel.FunctionArgs.CellsLabel.OutputArg='Image';
savecellslabel.FunctionArgs.CurFrame.FunctionInstance='SegmentationLoop';
savecellslabel.FunctionArgs.CurFrame.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,savecellslabel);

segmentationloop.InstanceName='SegmentationLoop';
segmentationloop.FunctionHandle=@forLoop;
segmentationloop.FunctionArgs.StartLoop.Value=StartFrame;
segmentationloop.FunctionArgs.EndLoop.Value=(StartFrame+FrameCount-1)*FrameStep;
segmentationloop.FunctionArgs.IncrementLoop.Value=FrameStep;
segmentationloop.FunctionArgs.MatchingGroups.Value=[];
segmentationloop.FunctionArgs.Tracks.Value=[];
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.FunctionInstance='LoadTracksLayout';
segmentationloop.FunctionArgs.LoadTracksLayout_TracksLayout.OutputArg='TracksLayout';
segmentationloop.KeepValues.ContinueTracks_Tracks.FunctionInstance='IfIsEmptyPreviousCellsLabel';
segmentationloop.KeepValues.ContinueTracks_Tracks.OutputArg='ContinueTracks_Tracks';
segmentationloop.LoopFunctions=image_read_loop_functions;
functions_list=addToFunctionChain(functions_list,segmentationloop);

savetracks.InstanceName='SaveTracks';
savetracks.FunctionHandle=@saveTracks;
savetracks.FunctionArgs.TracksFileName.Value=[TracksFolder '/tracks.mat'];
savetracks.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
savetracks.FunctionArgs.Tracks.OutputArg='ContinueTracks_Tracks';
functions_list=addToFunctionChain(functions_list,savetracks);
```

```
global dependencies_list;
global dependencies_index;
dependencies_list={};
dependencies_index=java.util.Hashtable;
makeDependencies([]);
runFunctions();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assignCellToTrackUsingAll.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=assignCellToTrackUsingAll(input_args)
%Usage
%This module tracks objects in a time-lapse sequence of label matrices. It uses distance, speed,
%direction and shape parameters to determine which candidate object best matches a track. It
%can be optimized for tracking fast moving directional objects or slow moving objects.
%
%Input Structure Members
%CheckCellPath - To allow paths that go through another cell set this value to true otherwise set
%it to false.
%CellsCentroids - Current object centroids.
%CellsLabel - The current time step label matrix.
%CurrentTracks - Matrix containing the track assignments and shape parameters for the objects
%in the previous frame.
%DefaultParamWeights - A set of weights is assigned to each parameter based on its prediction
%power. Parameters with high prediction power are assigned high weights and parameters with
%low prediction power are assigned lower weights. If an object can be assigned to a matching
%group or is a sure match for a track the weights listed in this variable are used.
%DirectionRankingOrder - When the order of the parameters based on prediction power cannot
%be determined using a group match a set of default parameter ranks are used. The parameter
%order for fast directional objects is provided in this variable.
%DistanceRankingOrder - When the order of the parameters based on prediction power cannot
%be determined using a group match a set of default parameter ranks are used. The parameter
%order for slow-moving objects is provided in this variable.
%ExcludedTracks - List of track assignments that should not be changed.
%FrontParams - To force a set of parameters to the front so they are heavily weighted enter their
%column indices in the variable, otherwise set the variable to an empty vector.
%MatchingGroups - Matrix of current matching groups (vectors of shape or motility indices
%ordered by their prediction power).
%MatchingGroupsStats - Matrix of mean values for each parameter in each group.
%MaxAngleDiff - The maximum allowed angle difference between a track and a candidate object.
%If the angle is larger than this value direction ranking will not be used for this object.
%MaxDistRatio - The maximum allowed distance ratio between the two nearest candidate
%objects. If the ratio is higher than this value distance ranking will not be used.
%MaxSearchRadius - Sets an absolute lower bound for the search radius to prevent selecting too
%few candidate objects for a track.
%MaxTrackID - Current maximum track ID.
%MinSecondDistance - Minimum significant distance between the closest candidate object to a
%track and the second closest. Used to determine when distance should be used as a ranking
%parameter.
%MinSearchRadius - Sets an absolute higher bound for the search radius to prevent selecting
%too many candidate objects for a track.
%NrParamsForSureMatch - This value is used to indicate the minimum number of closest
%matches between a candidate object parameters and a track's object parameters that make the
%candidate object a sure match to the track.
%ParamsCoeffOfVariation - Matrix containing the coefficient of variation for each parameter.
%PreviousCellsLabel - The matrix label from the previous time step.
%PreviousTracks - Matrix containing the track assignments and shape parameters for the objects
%in the frame previous to those in CurrentTracks.
%RelevantParametersIndex - Shape or motility parameters that have been determined to be
%irrelevant for tracking the objects can be eliminated by setting the corresponding index in the
%variable to false. This indicates to the module not to use the parameters in computing track
%assignment probabilities.
%SearchRadiusPct - This value determines the size of the neighborhood from which candidate
%objects for matching the track are selected. It is a multiple of the distance to the nearest
%candidate in the current frame. Setting this variable equal to 1 turns this module into a
nearest-
%neighbor algorithm (only the nearest cell can be a candidate). It does not make sense to have a
%value lower than 1.
%ShapeParameters - Shape parameters (area, eccentricity, perimeter, etc.) extracted from the
%current label.
%TrackAssignments - List of track assignments that have already been completed.
%TracksLayout - Matrix describing the order of the columns in the tracks matrix.
%UnassignedCells - List of object IDs currently unassigned.
%UnknownParamWeights - A set of weights is assigned to each parameter based on its
%prediction power. Parameters with high prediction power are assigned high weights and
%parameters with low prediction power are assigned lower weights. If an object cannot be
%assigned to a matching group and is not a sure match for a track the weights listed in this
%variable are used.
%UnknownRankingOrder - When the order of the parameters based on prediction power cannot
```

```
%be determined using a group match a set of default parameter ranks are used. If the objects
%cannot be categorized as either slow-moving or fast directional the parameter order provided in
%this variable is used.
%
%Output Structure Members
%UnassignedIDs - List of object IDs currently unassigned.
%TrackAssignments - List of track assignments that have already been completed.
%MatchingGroups - Matrix of mean values for each parameter in each group.
%GroupIndex - Indicates the index of the group to which the object has been assigned.
%ExcludedTracks - List of track assignments that should not be changed.

unassignedIDs=input_args.UnassignedCells.Value;
cells_lbl=input_args.CellsLabel.Value;
prev_cells_lbl=input_args.PreviousCellsLabel.Value;
shape_params=input_args.ShapeParameters.Value;
cells_centroids=input_args.CellsCentroids.Value;
cur_tracks=input_args.CurrentTracks.Value;
prev_tracks=input_args.PreviousTracks.Value;
search_radius_pct=input_args.SearchRadiusPct.Value;
trackAssignments=input_args.TrackAssignments.Value;
tracks_layout=input_args.TracksLayout.Value;
max_tracks=input_args.MaxTrackID.Value;
matching_groups=input_args.MatchingGroups.Value;
matching_groups_stats=input_args.MatchingGroupsStats.Value;
params_coeff_var=input_args.ParamsCoeffOfVariation.Value;
excluded_tracks=input_args.ExcludedTracks.Value;
relevant_params_idx=input_args.RelevantParametersIndex.Value;
params_for_sure_match=input_args.NrParamsForSureMatch.Value;
param_weights=input_args.DefaultParamWeights.Value;
unknown_param_weights=input_args.UnknownParamWeights.Value;
distance_ranking_order=input_args.DistanceRankingOrder.Value;
direction_ranking_order=input_args.DirectionRankingOrder.Value;
unknown_ranking_order=input_args.UnknownRankingOrder.Value;
min_second_distance=input_args.MinSecondDistance.Value;
max_dist_ratio=input_args.MaxDistRatio.Value;
max_angle_diff=input_args.MaxAngleDiff.Value;
b_check_path=input_args.CheckCellPath.Value;
max_search_dist=input_args.MaxSearchRadius.Value;
min_search_dist=input_args.MinSearchRadius.Value;

%assign current cell to a track
cur_id=unassignedIDs(1);
%first get a list of all tracks in the current search radius
centroid1Col=tracks_layout.Centroid1Col;
centroid2Col=tracks_layout.Centroid2Col;
trackIDCol=tracks_layout.TrackIDCol;
front_params=input_args.FrontParams.Value;

[nearby_tracks_sorted group_idx matching_groups]=getNearbyTracksSorted(cur_id,
cells_centroids,shape_params,tracks_layout,cur_tracks...

,prev_tracks,search_radius_pct,matching_groups,params_coeff_var,relevant_params_idx,matching_grou
ps_stats,params_for_sure_match,...

param_weights,unknown_param_weights,distance_ranking_order,direction_ranking_order,unknown_rankin
g_order,min_second_distance,...
    max_dist_ratio,max_angle_diff,max_search_dist,min_search_dist,front_params);
if (isempty(nearby_tracks_sorted))
    nearby_tracks_nr=0;
else
    nearby_tracks_ids=nearby_tracks_sorted(:,trackIDCol);
    %does list have at least one track?
    nearby_tracks_nr=length(nearby_tracks_ids);
end
for i=1:nearby_tracks_nr
    %pick the best track for current cell
    best_track_id=nearby_tracks_ids(i,trackIDCol);
    track_lbl_id=getLabelId(prev_cells_lbl, nearby_tracks_sorted(i,centroid1Col:centroid2Col));
    if (max(excluded_tracks{cur_id}==best_track_id)==1)
        %can't get this track
        continue;
    end
    if (b_check_path)
        if (pathGoesThroughACell(cells_lbl, prev_cells_lbl,cur_id,track_lbl_id,0))
            %resulting path would go through another cell - this track cannot match this cell
            continue;
        end
    end
    if (isempty(trackAssignments))
        track_idx=[];
        competing_id=[];
```

```
    else
        track_idx=find(trackAssignments(:,1)==best_track_id,1);
        competing_id=trackAssignments(track_idx,2);
    end
    %is the track this cell wants claimed?
    if (isempty(track_idx))
        %track is not claimed-assign it to this cell
        trackAssignments=[trackAssignments; [best_track_id cur_id]];
        %remove cell from unassigned list
        unassignedIDs(1)=[];
        output_args.UnassignedIDs=unassignedIDs;
        output_args.TrackAssignments=trackAssignments;
        output_args.MatchingGroups=matching_groups;
        output_args.GroupIndex=group_idx;
        output_args.ExcludedTracks=excluded_tracks;
        return;
    else
        %which cell is prefered by the track?
        competing_shape_params=[shape_params(cur_id,:); shape_params(competing_id,:)];
        competing_cells_centroids=[cells_centroids(cur_id,:); cells_centroids(competing_id,:)];
        %sort the two cells with respect of their goodness-of-fit to the
        %track

preferred_cell_id=getBetterMatchToTrack(nearby_tracks_sorted(i,:),competing_shape_params,competin
g_cells_centroids,[cur_id;competing_id]...
            ,prev_tracks,matching_groups,tracks_layout, cells_lbl, prev_cells_lbl,
relevant_params_idx, param_weights,...
            unknown_param_weights,unknown_ranking_order);
        if (isempty(preferred_cell_id))
            continue;
        end
        if (preferred_cell_id==competing_id)
            %the competing cell is preferred does this cell have other
            %tracks it can get?
            available_tracks_sorted=nearby_tracks_sorted(i+1:nearby_tracks_nr,:);

excluded_tracks_idx=ismember(available_tracks_sorted(:,trackIDCol),excluded_tracks{cur_id});
            available_tracks_sorted(excluded_tracks_idx,:)=[];
            if
(canCellGetAnotherTrack(cur_id,available_tracks_sorted,prev_cells_lbl,cells_lbl,...

tracks_layout,trackAssignments,shape_params,cells_centroids,prev_tracks,matching_groups,true,rele
vant_params_idx,...
                    param_weights,unknown_param_weights,unknown_ranking_order,b_check_path))
                %it does. we'll have to leave this track to the
                 %cell with the stronger claim
                 continue;
            end
            %this cell has no other tracks it can connect to. does the
            %competing cell have other tracks it can get?
            other_tracks_sorted=getNearbyTracksSorted(competing_id,
cells_centroids,shape_params,tracks_layout,cur_tracks,...

prev_tracks,search_radius_pct,matching_groups,params_coeff_var,relevant_params_idx,matching_group
s_stats,params_for_sure_match,...

param_weights,unknown_param_weights,distance_ranking_order,direction_ranking_order,unknown_rankin
g_order,...

min_second_distance,max_dist_ratio,max_angle_diff,max_search_dist,min_search_dist,front_params);
            %remove the current track

other_tracks_sorted(other_tracks_sorted(:,trackIDCol)==nearby_tracks_sorted(i,trackIDCol),:)=[];
            %remove any tracks that have already been excluded

excluded_tracks_idx=ismember(other_tracks_sorted(:,trackIDCol),excluded_tracks{competing_id});
            other_tracks_sorted(excluded_tracks_idx,:)=[];
            if
(isempty(other_tracks_sorted)||(~canCellGetAnotherTrack(competing_id,other_tracks_sorted,prev_cel
ls_lbl,cells_lbl,...

tracks_layout,trackAssignments,shape_params,cells_centroids,prev_tracks,matching_groups,false,rel
evant_params_idx,...
                    param_weights,unknown_param_weights,unknown_ranking_order,b_check_path)))
                %this track is the last option for the competing cell
                 %as well. we'll have to leave it to it since it is
                 %preferred by the track
                 excluded_tracks{cur_id}=[excluded_tracks{cur_id}; best_track_id];
                 continue;
            end
            %the competing cell has other options this cell doesn't so take
```

```
            %the track even though it has a weaker claim
            unassignedIDs(1)=competing_id;
            trackAssignments(track_idx,2)=cur_id;
            output_args.UnassignedIDs=unassignedIDs;
            output_args.TrackAssignments=trackAssignments;
            output_args.MatchingGroups=matching_groups;
            output_args.GroupIndex=group_idx;
            output_args.ExcludedTracks=excluded_tracks;
            return;
        else
            %this cell is preferred by the track
            available_tracks_sorted=nearby_tracks_sorted(i+1:nearby_tracks_nr,:);

excluded_tracks_idx=ismember(available_tracks_sorted(:,trackIDCol),excluded_tracks{cur_id});
            available_tracks_sorted(excluded_tracks_idx,:)=[];
            if
(isempty(available_tracks_sorted)||(~canCellGetAnotherTrack(cur_id,available_tracks_sorted,prev_c
ells_lbl,cells_lbl,...

tracks_layout,trackAssignments,shape_params,cells_centroids,prev_tracks,matching_groups,false,rel
evant_params_idx,...
                    param_weights,unknown_param_weights,unknown_ranking_order,b_check_path)))
                %this cell has no other tracks it can get
                 %bump the cell with the weaker claim
                 unassignedIDs(1)=trackAssignments(track_idx,2);
                 trackAssignments(track_idx,2)=cur_id;
                 output_args.UnassignedIDs=unassignedIDs;
                 output_args.TrackAssignments=trackAssignments;
                 output_args.MatchingGroups=matching_groups;
                 output_args.GroupIndex=group_idx;
                 excluded_tracks{competing_id}=[excluded_tracks{competing_id}; best_track_id];
                 output_args.ExcludedTracks=excluded_tracks;
                 return;
            end
            %does the competing cell have other options?
            other_tracks_sorted=getNearbyTracksSorted(competing_id,
cells_centroids,shape_params,tracks_layout,cur_tracks,...

prev_tracks,search_radius_pct,matching_groups,params_coeff_var,relevant_params_idx,matching_group
s_stats,params_for_sure_match,...

param_weights,unknown_param_weights,distance_ranking_order,direction_ranking_order,unknown_rankin
g_order,min_second_distance,...
                max_dist_ratio,max_angle_diff,max_search_dist,min_search_dist,front_params);
            %remove the current track

other_tracks_sorted(other_tracks_sorted(:,trackIDCol)==nearby_tracks_sorted(i,trackIDCol),:)=[];
            %remove any tracks that have already been excluded

excluded_tracks_idx=ismember(other_tracks_sorted(:,trackIDCol),excluded_tracks{competing_id});
            other_tracks_sorted(excluded_tracks_idx,:)=[];
            if
(canCellGetAnotherTrack(competing_id,other_tracks_sorted,prev_cells_lbl,cells_lbl,tracks_layout,t
rackAssignments,shape_params,...

cells_centroids,prev_tracks,matching_groups,false,relevant_params_idx,param_weights,unknown_param
_weights,...
                    unknown_ranking_order,b_check_path))
                %yes relinquish the track to this cell with the stronger
                %claim
                unassignedIDs(1)=trackAssignments(track_idx,2);
                trackAssignments(track_idx,2)=cur_id;
                output_args.UnassignedIDs=unassignedIDs;
                output_args.TrackAssignments=trackAssignments;
                output_args.MatchingGroups=matching_groups;
                output_args.GroupIndex=group_idx;
                excluded_tracks{competing_id}=[excluded_tracks{competing_id}; best_track_id];
                output_args.ExcludedTracks=excluded_tracks;
                return;
            else
                %this cell can get other tracks, the competing cell
                %can't. let it keep the track even though it has a
                %weaker claim
                continue;
            end
        end
    end
end


%list of potential tracks is empty
```

```
%start new track
if isempty(trackAssignments)
    max_track_id=max([cur_tracks(:,trackIDCol);max_tracks]);
else
    max_track_id=max([cur_tracks(:,trackIDCol); trackAssignments(:,1); max_tracks]);
end
trackAssignments=[trackAssignments; [max_track_id+1 cur_id]];
%remove cell from unassigned list
unassignedIDs(1)=[];

output_args.UnassignedIDs=unassignedIDs;
output_args.TrackAssignments=trackAssignments;
output_args.MatchingGroups=matching_groups;
output_args.GroupIndex=group_idx;
output_args.ExcludedTracks=excluded_tracks;

%end assignCellToTrackUsingAll
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assignCellToTrackUsingNN.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=assignCellToTrackUsingNN(input_args)
%Usage
%This module tracks objects in a time-lapse sequence of label matrices using a nearest-neighbor
%algorithm.
%
%Input Structure Members
%CellsCentroids - Current object centroids.
%CurrentTracks - Matrix containing the track assignments for the objects in the previous frame.
%MaxTrackID - Current maximum track ID.
%TrackAssignments - List of track assignments that have already been completed.
%TracksLayout - Matrix describing the order of the columns in the tracks matrix.
%UnassignedCells - List of object IDs currently unassigned.
%
%Output Structure Members
%ExcludedTracks - Returns empty value. Included for compatibility only.
%GroupIndex - Returns zero. Included for compatibility only.
%MatchingGroups - Returns empty value. Included for compatibility only.
%TrackAssignments - List of track assignments that have already been completed.
%UnassignedIDs - List of object IDs currently unassigned.

unassignedIDs=input_args.UnassignedCells.Value;
cells_centroids=input_args.CellsCentroids.Value;
cur_tracks=input_args.CurrentTracks.Value;
trackAssignments=input_args.TrackAssignments.Value;
tracks_layout=input_args.TracksLayout.Value;
max_tracks=input_args.MaxTrackID.Value;
centroid1Col=tracks_layout.Centroid1Col;
centroid2Col=tracks_layout.Centroid2Col;
trackIDCol=tracks_layout.TrackIDCol;

%assign current cell to a track
cur_id=unassignedIDs(1);
unassignedIDs(1)=[];
cur_cell_centroid=cells_centroids(cur_id,:);
dist_to_existing_tracks=hypot(cur_tracks(:,centroid1Col)-
cur_cell_centroid(1),cur_tracks(:,centroid2Col)-cur_cell_centroid(2));
[min_dist nearest_track_idx]=min(dist_to_existing_tracks);
nearest_track_id=cur_tracks(nearest_track_idx,trackIDCol);
if (isempty(trackAssignments))
    track_idx=[];
    competing_id=[];
else
    track_idx=find(trackAssignments(:,1)==nearest_track_id,1);
    competing_id=trackAssignments(track_idx,2);
end
%is the track this cell wants claimed?
if (isempty(track_idx))
    %track is not claimed-assign it to this cell
    trackAssignments=[trackAssignments; [nearest_track_id cur_id]];
else
    if isempty(trackAssignments)
        max_track_id=max([cur_tracks(:,trackIDCol);max_tracks]);
    else
        max_track_id=max([cur_tracks(:,trackIDCol); trackAssignments(:,1); max_tracks]);
    end
    %which cell is the better match?
    track_centroid=cur_tracks(nearest_track_idx,centroid1Col:centroid2Col);
```

```
    competing_centroid=cells_centroids(competing_id,:);
    competing_dist=hypot(track_centroid(1)-competing_centroid(1),track_centroid(2)-
competing_centroid(2));
    if (competing_dist<min_dist)
        %the competing cell is preferred
        trackAssignments=[trackAssignments; [max_track_id+1 cur_id]];
    else
        %this cell is preferred by the track
        trackAssignments(track_idx,2)=cur_id;
        trackAssignments=[trackAssignments; [max_track_id+1 competing_id]];
    end
end

output_args.UnassignedIDs=unassignedIDs;
output_args.TrackAssignments=trackAssignments;
output_args.MatchingGroups=[];
output_args.GroupIndex=0;
output_args.ExcludedTracks=[];

%end assignCellToTracksUsingNN
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% assignCellToTrackUsingNN_3D.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=assignCellToTrackUsingNN_3D(input_args)
%Usage
%This module tracks objects in a time-lapse sequence of 3D label matrices using a nearest-
neighbor
%algorithm.
%
%Input Structure Members
%CellsCentroids - Current object centroids.
%CurrentTracks - Matrix containing the track assignments for the objects in the previous frame.
%MaxTrackID - Current maximum track ID.
%TrackAssignments - List of track assignments that have already been completed.
%TracksLayout - Matrix describing the order of the columns in the tracks matrix.
%UnassignedCells - List of object IDs currently unassigned.
%
%Output Structure Members
%ExcludedTracks - Returns empty value. Included for compatibility only.
%GroupIndex - Returns zero. Included for compatibility only.
%MatchingGroups - Returns empty value. Included for compatibility only.
%TrackAssignments - List of track assignments that have already been completed.
%UnassignedIDs - List of object IDs currently unassigned.

unassignedIDs=input_args.UnassignedCells.Value;
cells_centroids=input_args.CellsCentroids.Value;
cur_tracks=input_args.CurrentTracks.Value;
trackAssignments=input_args.TrackAssignments.Value;
tracks_layout=input_args.TracksLayout.Value;
max_tracks=input_args.MaxTrackID.Value;
centroid1Col=tracks_layout.Centroid1Col;
centroid3Col=tracks_layout.Centroid3Col;
nr_tracks=size(cur_tracks,1);

trackIDCol=tracks_layout.TrackIDCol;

%assign current cell to a track
cur_id=unassignedIDs(1);
unassignedIDs(1)=[];
cur_cell_centroid=cells_centroids(cur_id,:);
dist_to_existing_tracks=sqrt(sum((cur_tracks(:,centroid1Col:centroid3Col)-
repmat(cur_cell_centroid,nr_tracks,1)).^2,2));
[min_dist nearest_track_idx]=min(dist_to_existing_tracks);
nearest_track_id=cur_tracks(nearest_track_idx,trackIDCol);
if (isempty(trackAssignments))
    track_idx=[];
    competing_id=[];
else
    track_idx=find(trackAssignments(:,1)==nearest_track_id,1);
    competing_id=trackAssignments(track_idx,2);
end
%is the track this cell wants claimed?
if (isempty(track_idx))
    %track is not claimed-assign it to this cell
    trackAssignments=[trackAssignments; [nearest_track_id cur_id]];
else
    if isempty(trackAssignments)
```

```
        max_track_id=max([cur_tracks(:,trackIDCol);max_tracks]);
    else
        max_track_id=max([cur_tracks(:,trackIDCol); trackAssignments(:,1); max_tracks]);
    end
    %which cell is the better match?
    track_centroid=cur_tracks(nearest_track_idx,centroid1Col:centroid3Col);
    competing_centroid=cells_centroids(competing_id,:);
    competing_dist=sqrt(sum((track_centroid-competing_centroid).^2));
    if (competing_dist<min_dist)
        %the competing cell is preferred
        trackAssignments=[trackAssignments; [max_track_id+1 cur_id]];
    else
        %this cell is preferred by the track
        trackAssignments(track_idx,2)=cur_id;
        trackAssignments=[trackAssignments; [max_track_id+1 competing_id]];
    end
end

output_args.UnassignedIDs=unassignedIDs;
output_args.TrackAssignments=trackAssignments;
output_args.MatchingGroups=[];
output_args.GroupIndex=0;
output_args.ExcludedTracks=[];

%end assignCellToTracksUsingNN
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% breakTrack.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function breakTrack()
%used by the manual tracking GUI to break a track in two at the current time
global mtr_gui_struct;

track_id=mtr_gui_struct.SelectedCellID;
if (track_id==0)
    warndlg('No cell is selected!');
    return;
end
cur_time=(mtr_gui_struct.CurFrame-1)*mtr_gui_struct.TimeFrame;
ancestry_layout=mtr_gui_struct.AncestryLayout;
ancestry_record=mtr_gui_struct.CurrentAncestryRecord;
if (cur_time==ancestry_record(ancestry_layout.StartTimeCol))
    warndlg('Can''t break a track at its start point!');
    return;
end
ancestry_records=mtr_gui_struct.CellsAncestry;
new_track_id=max(ancestry_records(:,ancestry_layout.TrackIDCol))+1;
tracks_layout=mtr_gui_struct.TracksLayout;
tracks=mtr_gui_struct.Tracks;
new_track_idx=(tracks(:,tracks_layout.TrackIDCol)==track_id)&(tracks(:,tracks_layout.TimeCol)>=cu
r_time);
tracks(new_track_idx,tracks_layout.TrackIDCol)=new_track_id;
mtr_gui_struct.Tracks=tracks;

%add ancestry record
new_ancestry_record=zeros(1,size(ancestry_records,2));
new_ancestry_record(ancestry_layout.TrackIDCol)=new_track_id;
new_ancestry_record(ancestry_layout.StartTimeCol)=cur_time;
new_ancestry_record(ancestry_layout.StopTimeCol)=ancestry_record(ancestry_layout.StopTimeCol);
new_ancestry_record(ancestry_layout.GenerationCol)=1;
ancestry_records=[ancestry_records; new_ancestry_record];

%update the ancestry record of the old track
ancestry_idx=ancestry_records(:,ancestry_layout.TrackIDCol)==track_id;
ancestry_records(ancestry_idx,ancestry_layout.StopTimeCol)=cur_time-mtr_gui_struct.TimeFrame;
%find any children of the new track and update their parent ids
ancestry_idx=(ancestry_records(:,ancestry_layout.ParentIDCol)==track_id)&...
    (ancestry_records(:,ancestry_layout.StartTimeCol)>=cur_time);
ancestry_records(ancestry_idx,ancestry_layout.ParentIDCol)=new_track_id;
mtr_gui_struct.CellsAncestry=ancestry_records;
mtr_gui_struct.SelectedCellID=0;
mtr_gui_struct.SelectedCellLabelID=0;
updateTrackImage(mtr_gui_struct.CurFrame,mtr_gui_struct.ShowLabels,mtr_gui_struct.ShowOutlines);
addSelectionLayers();

%end breakTrack
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% butterworthFreqFilter.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=butterworthFreqFilter(input_args)
% Usage
% This module is used to filter a 2D image in the frequency domain
% Input Structure Members
% Image - Grayscale image to be filtered.
% CutOffFreq - The cut-off frequency of the Butterworth filter
% FilterOrder - The order of the filter
% FilterType - Can be either 'HighPass' or 'LowPass'
%
% Output Structure Members
% Image - Filtered 2D grayscale image.
img=input_args.Image.Value;
original_img_sz=size(img);
img_sz(1)=2^nextpow2(2*original_img_sz(1)-1);
img_sz(2)=2^nextpow2(2*original_img_sz(2)-1);
%calculate the square distance matrix from the center point
dist_matrix=false(img_sz(1:2));
dist_matrix(floor(img_sz(1)/2)+1,floor(img_sz(2)/2)+1)=true;
dist_matrix=bwdist(dist_matrix).^2;
%arrange it so it's in the proper form for fft2
dist_matrix=ifftshift(rot90(dist_matrix,2));
%setup the filter
cutoff_freq=input_args.CutOffFreq.Value;
filter_order=input_args.FilterOrder.Value;
filter_type=input_args.FilterType.Value;
switch(filter_type)
    case 'LowPass'
        butterworth_filter=1./(1+ (dist_matrix./cutoff_freq).^(2*filter_order));
    case 'HighPass'
        butterworth_filter=1-1./(1+ (dist_matrix./cutoff_freq).^(2*filter_order));
end
%filter the image in the frequency domain

img_fft=fft2(double(img),img_sz(1),img_sz(2));
img_filtered=real(ifft2(butterworth_filter.*img_fft));

output_args.Image=img_filtered(1:original_img_sz(1),1:original_img_sz(2));

%butterworthFilter
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% butterworthFreqFilter3D.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=butterworthFreqFilter3D(input_args)
% Usage
% This module is used to filter a 3D image in the frequency domain one
% z-slice at a time
% Input Structure Members
% Image - Grayscale image to be converted.
% CutOffFreq - The cut-off frequency of the butterworth filter
% FilterOrder - The order of the filter
% FilterType - Can be either 'HighPass' or 'LowPass'
%
% Output Structure Members
% Image - Filtered 3D grayscale image.

img=input_args.Image.Value;
original_img_sz=size(img);
img_sz(1)=2^nextpow2(2*original_img_sz(1)-1);
img_sz(2)=2^nextpow2(2*original_img_sz(2)-1);
%calculate the square distance matrix from the center point
dist_matrix=false(img_sz(1:2));
dist_matrix(floor(img_sz(1)/2)+1,floor(img_sz(2)/2)+1)=true;
dist_matrix=bwdist(dist_matrix).^2;
%arrange it so it's in the proper form for fft2
dist_matrix=ifftshift(rot90(dist_matrix,2));
%setup the filter
cutoff_freq=input_args.CutOffFreq.Value;
filter_order=input_args.FilterOrder.Value;
filter_type=input_args.FilterType.Value;
switch(filter_type)
    case 'LowPass'
        butterworth_filter=1./(1+ (dist_matrix./cutoff_freq).^(2*filter_order));
    case 'HighPass'
        butterworth_filter=1-1./(1+ (dist_matrix./cutoff_freq).^(2*filter_order));
end
```

```
%filter the image in the frequency domain
nr_slices=original_img_sz(3);
img_filtered=zeros(original_img_sz);
for i=1:nr_slices
    slice_fft=fft2(double(img(:,:,i)),img_sz(1),img_sz(2));
    filtered_slice=real(ifft2(butterworth_filter.*slice_fft));
    img_filtered(:,:,i)=filtered_slice(1:original_img_sz(1),1:original_img_sz(2));
end
output_args.Image=img_filtered;

%butterworthFilter
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%% calculateCropSize.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=calculateCropSize(input_args)
%This module is used to calculate how much a image will need to be cropped to remove the
%microscope stage offsets
%Input Structure Members
%Image - Image in the sequence to be cropped.
%XYOffsets - Array containing the offsets for all the images in the series.
%Output Structure Members
%CropSize - Array indicating how much the image will need to be cropped in
%each direction.

img=input_args.Image.Value;
img_sz=size(img);
xy_offsets=input_args.XYOffsets.Value;
max_offsets=2*(max(abs(xy_offsets))+1);
output_args.CropSize=img_sz-[max_offsets(2) max_offsets(1)];

%end calculateCropRectangle
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%% callFunction.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function function_output=callFunction(instance_name,b_clear_args)
%CellAnimation core function used to execute the modules and sub-modules in
%an assay
global dependencies_list;
global dependencies_index;

cur_idx=dependencies_index.get(instance_name);
dependency_item=dependencies_list{cur_idx};

functionHandle=dependency_item.FunctionHandle;
switch(char(functionHandle))
    case {'forLoop','if_statement','whileLoop'}
        function_output=functionHandle(dependency_item);
    otherwise
        field_names=fieldnames(dependency_item);
        if max(strcmp(field_names,'FunctionArgs'))
            function_output=functionHandle(dependency_item.FunctionArgs);
        else
            function_output=functionHandle();
        end
end
instance_name=dependency_item.InstanceName;
updateArgs(instance_name,function_output,'output');
if (b_clear_args)
    clearArgs(instance_name);
end

%end callFunction
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%% canCellGetAnotherTrack.m %%%%%%%%%%%%%%%%%%%%%%%%
function
b_cell_can_get_another_track=canCellGetAnotherTrack(cur_id,nearby_tracks_sorted,prev_cells_lbl,ce
lls_lbl,...

tracks_layout,trackAssignments,shape_params,cells_centroids,prev_tracks,matching_groups,b_bumping
_allowed,relevant_params_idx,...
```

```
    param_weights,unknown_param_weights,unknown_ranking_order,b_check_path)
%helper function for the CA tracking module used to determine if a cell has
%another track that it can use if the track currently assigned to it is
%assigned to another cell
if (isempty(nearby_tracks_sorted))
    b_cell_can_get_another_track=false;
    return;
end
centroid1Col=tracks_layout.Centroid1Col;
centroid2Col=tracks_layout.Centroid2Col;
trackIDCol=tracks_layout.TrackIDCol;
b_found_track=false;
for i=1:size(nearby_tracks_sorted,1)
    track_lbl_id=getLabelId(prev_cells_lbl,nearby_tracks_sorted(i,centroid1Col:centroid2Col));
    if (b_check_path)
        if (pathGoesThroughACell(cells_lbl,prev_cells_lbl,cur_id,track_lbl_id,0))
            continue;
        end
    end
    if (isempty(trackAssignments))
        track_idx=[];
    else
        track_idx=find(trackAssignments(:,1)==nearby_tracks_sorted(i,trackIDCol),1);
    end
    if (isempty(track_idx))
        b_found_track=true;
        break;
    else
        if (~b_bumping_allowed)
            %not allowed to bump other cells
            continue;
        end
        test_id=trackAssignments(track_idx,2);
        %which cell is prefered by the track?
        test_shape_params=[shape_params(cur_id,:); shape_params(test_id,:)];
        test_cells_centroids=[cells_centroids(cur_id,:); cells_centroids(test_id,:)];
        %sort the two cells with respect of their goodness-of-fit to the
        %track

preferred_cell_id=getBetterMatchToTrack(nearby_tracks_sorted(i,:),test_shape_params,test_cells_ce
ntroids,...

[cur_id;test_id],prev_tracks,matching_groups,tracks_layout,cells_lbl,prev_cells_lbl,relevant_para
ms_idx,param_weights,...
            unknown_param_weights,unknown_ranking_order);
        if (isempty(preferred_cell_id)||(preferred_cell_id==test_id))
            continue;
        else
            %found another track which this cell can use
            b_found_track=true;
            break;
        end
    end
end
if (b_found_track)
    b_cell_can_get_another_track=true;
else
    b_cell_can_get_another_track=false;
end

%end canCellGetAnotherTrack
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% checkBoxLabelsEvent.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function checkBoxLabelsEvent()
%helper function for the manual tracking review module. used to turn the
%label display on and off

global mtr_gui_struct;
mtr_gui_struct.ShowLabels=get(mtr_gui_struct.CheckBoxLabelsHandle,'Value');
updateTrackImage(mtr_gui_struct.CurFrame,mtr_gui_struct.ShowLabels,mtr_gui_struct.ShowOutlines);
addSelectionLayers();

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% checkBoxOutlinesEvent.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function checkBoxOutlinesEvent()
%helper function for the manual tracking review module. used to turn the
%display of detected cell outlines on and off
global mtr_gui_struct;
mtr_gui_struct.ShowOutlines=get(mtr_gui_struct.CheckBoxOutlinesHandle,'Value');
updateTrackImage(mtr_gui_struct.CurFrame,mtr_gui_struct.ShowLabels,mtr_gui_struct.ShowOutlines);
addSelectionLayers();

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% clearArgs.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function clearArgs(instance_name)
%core CellAnimation function. clear argument values to free up memory

global dependencies_list;
global dependencies_index;

cur_idx=dependencies_index.get(instance_name);
dependency_item=dependencies_list{cur_idx};
function_args=dependency_item.FunctionArgs;
field_names=fieldnames(function_args);
for i=1:size(field_names,1)
    arg_struct=function_args.(field_names{i});
    arg_field_names=fieldnames(arg_struct);
    %if the field contains a value field that is set by a function clear it
    if ((size(arg_field_names,1)>1)&&(max(strcmp('Value',arg_field_names))==1))
        dependencies_list{cur_idx}.FunctionArgs.(field_names{i}).Value=[];
    end
end

%end clearArgs
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% clearBorderObjectsInLabel.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=clearBorderObjectsInLabel(input_args)
% Usage
% This module is used to remove objects touching the border of a label
% matrix
% Input Structure Members
% LabelMatrix - The label matrix to be processed.
% Output Structure Members
% LabelMatrix - The resulting label matrix.

lbl_matrix=input_args.LabelMatrix.Value;
lbl_sz=size(lbl_matrix);
%get indexes of the objects touching the sides
remove_obj_idx=lbl_matrix(1,1:lbl_sz(2));
remove_obj_idx=[remove_obj_idx lbl_matrix(lbl_sz(1),1:lbl_sz(2))];
remove_obj_idx=[remove_obj_idx lbl_matrix(1:lbl_sz(1),1)'];
remove_obj_idx=[remove_obj_idx lbl_matrix(1:lbl_sz(1),lbl_sz(2))'];
remove_obj_idx=unique(remove_obj_idx);
if (remove_obj_idx(1)==0)
    remove_obj_idx(1)=[];
end
if isempty(remove_obj_idx)
    %nothing to remove
    output_args.LabelMatrix=lbl_matrix;
end

remove_obj_idx=ismember(lbl_matrix,remove_obj_idx);
lbl_matrix(remove_obj_idx)=0;
output_args.LabelMatrix=makeContinuousLabelMatrix(lbl_matrix);

%end clearBorderObjectsInLabel
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% clearSmallObjects.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=clearSmallObjects(input_args)
% Simple wrapper module for bwareaopen MATLAB function
% Input Structure Members
```

```
% Image - Binary image from which objects will be removed.
% MinObjectArea - Objects with an area smaller than this value will be removed.
% Output Structure Members
% Image - Filtered binary image.

output_args.Image=bwareaopen(input_args.Image.Value,input_args.MinObjectArea.Value);

%end clearSmallObjects
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% closeImage.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=closeImage(input_args)
% Simple wrapper for MATLAB close function
% Input Members
% FigureNr - the figure number to be closed

close(input_args.FigureNr.Value);
output_args=[];

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% collapseText.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function new_list=collapseText(listbox_list,selection_idx,sublevel_depth)
%helper function for assayEditorGUI. collapse the submodules belonging to the current module
list_head=listbox_list(1:selection_idx);
list_tail=listbox_list((selection_idx+1):end);
items_level=cellfun(@getSelectionLevel,list_tail);
%find any items at this level that are chains
i=1;
tail_len=length(list_tail);
while((i<=tail_len)&&(items_level(i)>sublevel_depth))
    selection_text=list_tail{i};
    if ((items_level(i)==sublevel_depth)&&strcmp(selection_text(1:9),'<html><i>'))
        %prevent the chain from being collapsed
        items_level(i:end)=0;
        break;
    end
    i=i+1;
end
keep_idx=find(items_level<=sublevel_depth);
if ~isempty(keep_idx)
    items_level(keep_idx(1):end)=0;
end
new_list=[list_head; list_tail(items_level<=sublevel_depth)];

%end collapseText
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% combineImages.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=combineImages(input_args)
%Usage
%This module is used to combine two binary images using logical operations.
%
%Input Structure Members
%CombineOperation - String value indicating the logical operation used to combine the images.
%Currently, only AND and OR are supported.
%Image1 - First binary image.
%Image2 - Second binary image.
%
%Output Structure Members
%Image - Binary image resulting from the logical operation.

switch (input_args.CombineOperation.Value)
    case 'AND'
        output_args.Image=input_args.Image1.Value&input_args.Image2.Value;
    case 'OR'
        output_args.Image=input_args.Image1.Value|input_args.Image2.Value;
end

%end combineImages
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% compareValues.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=compareValues(input_args)
%Usage
%This module is used to compare two numerical values using the specified operation.
%
%Input Structure Members
%Arg1 - First numerical value.
%Arg2 - Second numerical value.
%Operation - String representing the mathematical operation to be performed.
%Currently, ">","<",">=","<=" and "==" are supported.
%
%Output Structure Members
%BooleanOut - The result of the operation. Can be either 1 (true) or 0 (false).

switch input_args.Operation.Value
    case '>'
        output_args.BooleanOut=input_args.Arg1.Value>input_args.Arg2.Value;
    case '<'
        output_args.BooleanOut=input_args.Arg1.Value<input_args.Arg2.Value;
    case '>='
        output_args.BooleanOut=input_args.Arg1.Value>=input_args.Arg2.Value;
    case '<='
        output_args.BooleanOut=input_args.Arg1.Value<=input_args.Arg2.Value;
    case '=='
        output_args.BooleanOut=input_args.Arg1.Value==input_args.Arg2.Value;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% concatenateText.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=concatenateText(input_args)
% Usage
% This module is used to cobine several pieces of text together to form a
% string. It can be used for example to generate an absolute path name from a
% directory name, a file name and an extension.
% Input Structure Members
% Text 1-10 - The text items to be combined. Text3-9 are optional.
% Output Structure Members
% String - The combined string.
string_out=[input_args.Text1.Value input_args.Text2.Value];
field_names=fieldnames(input_args);
if (max(strcmp(field_names,'Text3')))
    string_out=[string_out input_args.Text3.Value];
end
if (max(strcmp(field_names,'Text4')))
    string_out=[string_out input_args.Text4.Value];
end
if (max(strcmp(field_names,'Text5')))
    string_out=[string_out input_args.Text5.Value];
end
if (max(strcmp(field_names,'Text6')))
    string_out=[string_out input_args.Text6.Value];
end
if (max(strcmp(field_names,'Text7')))
    string_out=[string_out input_args.Text7.Value];
end
if (max(strcmp(field_names,'Text8')))
    string_out=[string_out input_args.Text8.Value];
end
if (max(strcmp(field_names,'Text9')))
    string_out=[string_out input_args.Text9.Value];
end

output_args.String=string_out;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% connectBlobs.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function connected_blob=connectBlobs(unconnected_blobs, nr_blobs)
%helper function to deal with fragmented blobs. get a bunch of unconnected blobs and connect them
using shortest distance
```

```
%lines

unconnected_blobs=imfill(unconnected_blobs,'holes');
%get the perimeter of the blobs
blobs_perim=bwperim(unconnected_blobs);
blobs_perim_lbl=bwlabeln(blobs_perim);
[connected_blob_1 connected_blob_2]=find(blobs_perim_lbl==1);
for i=2:nr_blobs
    [cur_blob_1 cur_blob_2]=find(blobs_perim_lbl==i);
    min_dist=Inf;
    for j=1:size(cur_blob_1,1)
        dist_to_cur_blob=hypot(connected_blob_1-cur_blob_1(j),connected_blob_2-cur_blob_2(j));
        [cur_min_dist cur_min_dist_idx]=min(dist_to_cur_blob);
        if (cur_min_dist<min_dist)
            min_dist=cur_min_dist;
            cur_blob_min_dist_idx=j;
            connected_blob_min_dist_idx=cur_min_dist_idx;
        end
    end

    connected_blob_closest_point_1=connected_blob_1(connected_blob_min_dist_idx);
    connected_blob_closest_point_2=connected_blob_2(connected_blob_min_dist_idx);
    cur_blob_closest_point_1=cur_blob_1(cur_blob_min_dist_idx);
    cur_blob_closest_point_2=cur_blob_2(cur_blob_min_dist_idx);

    cur_point=[connected_blob_closest_point_1 connected_blob_closest_point_2];
    min_dist=round(min_dist);
    connecting_line=zeros(min_dist,2);
    %progressively fill in the shortest path between the two
    %blobs
    for j=1:min_dist
        diff_1=cur_point(1)-cur_blob_closest_point_1;
        if (diff_1<0)
            cur_point(1)=cur_point(1)+1;
        elseif (diff_1>0)
            cur_point(1)=cur_point(1)-1;
        end
        diff_2=cur_point(2)-cur_blob_closest_point_2;
        if (diff_2<0)
            cur_point(2)=cur_point(2)+1;
        elseif (diff_2>0)
            cur_point(2)=cur_point(2)-1;
        end
        %add the cur_point to the list of pixels in the line connecting the
        %current blob to the connected blob
        connecting_line(j,:)=cur_point;
    end

    %add the connecting line and the current blob to the connected blob
    connected_blob_1=[connected_blob_1; connecting_line(:,1); cur_blob_1];
    connected_blob_2=[connected_blob_2; connecting_line(:,2); cur_blob_2];
end

connected_blob_lin=sub2ind(size(unconnected_blobs),connected_blob_1,connected_blob_2);
connected_blob=unconnected_blobs;
connected_blob(connected_blob_lin)=true;

%end connectBlobs
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% continueTrack.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function continueTrack()
%helper function for the manual tracking review module. use to start the
%process of continuing a track
global mtr_gui_struct;

mtr_gui_struct.TrackToContinueID=mtr_gui_struct.SelectedCellID;
mtr_gui_struct.TrackToContinueRecord=mtr_gui_struct.CurrentTrackRecord;
mtr_gui_struct.TrackToContinueAncestry=mtr_gui_struct.CurrentAncestryRecord;
mtr_gui_struct.ContinueTrack=true;

%end continueTrack
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% continueTracks.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function output_args=continueTracks(input_args)
% continueTracks
% Usage
% This module is used to continue the tracks with the new track assignments as tracking
progresses from frame to frame.
% Input Structure Members
% CurFrame - Integer value representing the current frame number.
% TrackAssignments - Matrix containing the new track assignments.
% TimeFrame - Integer value representing the current time frame.
% Output Structure Members
% NewTracks - Matrix containing the new tracks for the current frame.
% Tracks - Matrix containing the tracks including the new track assignments.

trackAssignments=input_args.TrackAssignments.Value;
[dummy tracks_sort_idx]=sort(trackAssignments(:,2));
tracks_ids_sorted=trackAssignments(tracks_sort_idx,1);
cur_time=(input_args.CurFrame.Value-1)*input_args.TimeFrame.Value;
output_args.NewTracks=[tracks_ids_sorted repmat(cur_time,size(tracks_ids_sorted,1),1)
input_args.CellsCentroids.Value...
    input_args.ShapeParameters.Value];
output_args.Tracks=[input_args.Tracks.Value; output_args.NewTracks];

%end continueTracks
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% createCheckerBoardPattern.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function checkerboard_pattern=createCheckerBoardPattern(selected_object)
%helper function for manual segmentation review. create a checkerboard
%pattern to indicate the selected object
cur_obj_size=sum(selected_object(:));
checkerboard_pattern=repmat([0;intmax('uint8')],floor(cur_obj_size/2),1);
if (rem(cur_obj_size,2))
    checkerboard_pattern=[checkerboard_pattern;0];
end
checkerboard_pattern=repmat(checkerboard_pattern,3,1);

%end createCheckerBoardPattern
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% cropImage.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=cropImage(input_args)
% cropImage
% Usage
% This module is used to crop an image from a point specified from the (0,0) coordinate to the
specified size.
% Input Structure Members
% CropSize - Two number vector containing the desired dimensions for the cropped image.
% Image - The image to be processed.
% XYOffset - Offset point (from the (0,0) coordinate) from which the image should be cropped.
% Output Structure Members
% Image - Cropped image.

img=input_args.Image.Value;
xy_offset=input_args.XYOffset.Value;
crop_size=input_args.CropSize.Value;
output_args.Image=img(xy_offset(2):(xy_offset(2)+crop_size(2)),xy_offset(1):(xy_offset(1)+crop_si
ze(1)));

%end cropImageWithOffset
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% cropImageWithOffset.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=cropImageWithOffset(input_args)
% cropImageWithOffset
% Usage
% This module is used to crop an image from a point specified from the center of the image to the
specified size.
% Input Structure Members
% CropSize - Two number vector containing the desired dimensions for the cropped image.
% Image - The image to be processed.
```

```
% XYOffset - Offset point (from the center of the original image) at which the cropped image
should be centered.
% Output Structure Members
% Image - Cropped image.

img=input_args.Image.Value;
xy_offset=input_args.XYOffset.Value;
crop_size=input_args.CropSize.Value;
img_sz=size(img);
center_pixel=floor((img_sz+1)/2);
x_low=center_pixel(1)-round(crop_size(1)/2+xy_offset(2));
x_high=x_low+crop_size(1)-1;
y_low=center_pixel(2)-round(crop_size(2)/2+xy_offset(1));
y_high=y_low+crop_size(2)-1;
output_args.Image=img(x_low:x_high,y_low:y_high);

%end cropImageWithOffset
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% deleteTrack.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function deleteTrack()
%helper function for manual tracking review. used to delete a selected
%track.
global mtr_gui_struct;

track_id=mtr_gui_struct.SelectedCellID;
if (track_id==0)
    warndlg('No cell is selected!');
    return;
end
tracks_layout=mtr_gui_struct.TracksLayout;
tracks=mtr_gui_struct.Tracks;
mtr_gui_struct.SelectedCellID=0;
mtr_gui_struct.SelectedCellLabelID=0;
%remove track
track_idx=tracks(:,tracks_layout.TrackIDCol)==track_id;
tracks(track_idx,:)=[];
mtr_gui_struct.Tracks=tracks;
%update ancestry records
ancestry_records=mtr_gui_struct.CellsAncestry;
ancestry_layout=mtr_gui_struct.AncestryLayout;
ancestry_idx=ancestry_records(:,ancestry_layout.TrackIDCol)==track_id;
ancestry_records(ancestry_idx,:)=[];
mtr_gui_struct.CellsAncestry=ancestry_records;
offsetGenerationNumber(track_id,-1);
updateTrackImage(mtr_gui_struct.CurFrame,mtr_gui_struct.ShowLabels,mtr_gui_struct.ShowOutlines);
addSelectionLayers();

%end deleteTrack
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% detectMergeCandidatesUsingDistance.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=detectMergeCandidatesUsingDistance(input_args)
% Usage
% This module is used to detect tracks that are never further than a small distance apart for
possible merging.
% Input Structure Members
% MaxMergeDistance - Maximum distance between tracks. Any tracks that drift further apart than
this distance at any time point will be merged.
% TrackIDs - Track IDs to be tested for merging.
% Tracks - Tracks matrix including time stamps and object centroids.
% TracksLayout - Matrix describing the order of the columns in the tracks matrix.
% Output Structure Members
% TracksToBeMerged - Paired list of the track IDs to be merged.

untested_ids=input_args.TrackIDs.Value;
tracks=input_args.Tracks.Value;
max_merge_dist=input_args.MaxMergeDistance.Value;
tracks_layout=input_args.TracksLayout.Value;
trackIDCol=tracks_layout.TrackIDCol;
timeCol=tracks_layout.TimeCol;
centroid1Col=tracks_layout.Centroid1Col;
centroid2Col=tracks_layout.Centroid2Col;
tracks_to_be_merged=[];
```

```
while (~isempty(untested_ids))
    curID=untested_ids(1);
    cur_track_idx=(tracks(:,trackIDCol)==curID);
    cur_track=tracks(cur_track_idx,:);
    cur_track_times=cur_track(:,timeCol);
    track_start_time=cur_track_times(1);
    %get the tracks that exist when this track appears for now just tracks
    %in the current time for the future we might add other times
    existing_tracks_idx=(tracks(:,timeCol)==track_start_time)&(~cur_track_idx);
    existing_tracks=tracks(existing_tracks_idx,:);
    cur_track_centroid=cur_track(1,centroid1Col:centroid2Col);
    existing_tracks_centroids=existing_tracks(:,centroid1Col:centroid2Col);
    %get the tracks that are near our cell
    dist_to_existing_tracks=hypot(existing_tracks_centroids(:,1)-cur_track_centroid(1),...
        existing_tracks_centroids(:,2)-cur_track_centroid(2));
    merge_candidates_idx=dist_to_existing_tracks<max_merge_dist;
    merge_candidates=existing_tracks(merge_candidates_idx,:);
    if isempty(merge_candidates)
        %no possible candidates to merge with so move on to next track
        untested_ids(1)=[];
        continue;
    end
    dist_to_existing_tracks=dist_to_existing_tracks(merge_candidates_idx);
    %sort the merge candidates by distance
    [dummy sort_idx]=sort(dist_to_existing_tracks);
    merge_candidates=merge_candidates(sort_idx,trackIDCol);

    %we have some tracks that may need to be merged with this track
    candidates_nr=size(merge_candidates,1);
    for j=1:candidates_nr
        candidateID=merge_candidates(j);
        candidate_track=tracks(tracks(:,trackIDCol)==candidateID,:);
        %get the times at which track exists
        candidate_times=candidate_track(:,timeCol);
        %get the times when both tracks exist
        [dummy cur_track_common_idx
candidate_track_common_idx]=intersect(cur_track_times,candidate_times);
        %get the centroids at those times

cur_track_common_times_centroids=cur_track(cur_track_common_idx,centroid1Col:centroid2Col);

candidate_common_times_centroids=candidate_track(candidate_track_common_idx,centroid1Col:centroid
2Col);
        dist_between_tracks=hypot(candidate_common_times_centroids(:,1)-
cur_track_common_times_centroids(:,1),...
            candidate_common_times_centroids(:,2)-cur_track_common_times_centroids(:,2));
        if (max(dist_between_tracks)<max_merge_dist)
            %found a track we should merge with
            tracks_to_be_merged=[tracks_to_be_merged; [candidateID curID]];
            %remove the candidateID from the list of tracks to be checked
            untested_ids(untested_ids==candidateID)=[];
            break;
        end
    end
    untested_ids(1)=[];
end

output_args.TracksToBeMerged=tracks_to_be_merged;

%end detectMergeCandidatesUsingDistance
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% detectMitoticEvents.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=detectMitoticEvents(input_args)
% Usage
% This module is used to detect mitotic events in time-lapse movies of cells stained with a
nuclear stain.
% Input Structure Members
% MaxSplitArea - Maximum area a nucleus may have at the time it splits.
% MaxSplitDistance - Nuclei that are further apart than this distance will not be considered as a
potential split pair.
% MaxSplitEccentricity - Any nucleus with an eccentricity above this value will not be considered
a split candidate.
% MinSplitEccentricity - Any nucleus with an eccentricity below this value will not be considered
a split candidate.
```

```
% MinTimeForSplit - A cell needs to have a lifespan above this value to be considered a split
candidate.
% Tracks - Tracks matrix including time stamps and object centroids.
% TracksLayout - Matrix describing the order of the columns in the tracks matrix.
% UntestedIDs - Track IDs to be tested for mitosis.
% Output Structure Members
% SplitCells - List of mitotic cell pairs.


tracks=input_args.Tracks.Value;
tracks_layout=input_args.TracksLayout.Value;
areaCol=tracks_layout.AreaCol;
eccCol=tracks_layout.EccCol;
timeCol=tracks_layout.TimeCol;
trackIDCol=tracks_layout.TrackIDCol;
centroid1Col=tracks_layout.Centroid1Col;
centroid2Col=tracks_layout.Centroid2Col;
max_split_area=input_args.MaxSplitArea.Value;
min_split_ecc=input_args.MinSplitEccentricity.Value;
max_split_ecc=input_args.MaxSplitEccentricity.Value;
max_split_dist=input_args.MaxSplitDistance.Value;
min_time_for_split=input_args.MinTimeForSplit.Value;
untested_ids=input_args.UntestedIDs.Value;
split_cells=[];
%these are the cells that have a known parent therefore a known age
cells_with_known_split_frames=java.util.Hashtable;

while (~isempty(untested_ids))
    curID=untested_ids(1);
    cur_track_idx=(tracks(:,trackIDCol)==curID);
    cur_track=tracks(cur_track_idx,:);
    cur_track_median_area=median(cur_track(:,areaCol));
    cur_track_times=cur_track(:,timeCol);
    track_start_time=cur_track_times(1);
    track_end_time=cur_track_times(end);
    cur_area=cur_track(1,areaCol);
    if (cur_area>1.3*cur_track_median_area)
        %a cell is smaller right after splitting
        untested_ids(1)=[];
        continue;
    end
    if (cur_area>max_split_area)
        %a cell is smaller right after splitting
        untested_ids(1)=[];
        continue;
    end
    cur_ecc=cur_track(1,eccCol);
    if (cur_ecc<min_split_ecc)
        %nuclei are elongated right after splitting
        untested_ids(1)=[];
        continue;
    end
    if (cur_ecc>max_split_ecc)
        %nuclei are not perfect lines
        untested_ids(1)=[];
        continue;
    end
    %get the tracks that exist when this track appears for now just tracks
    %in the current time for the future we might add other times
    existing_tracks_idx=(tracks(:,timeCol)==track_start_time)&(~cur_track_idx);
    existing_tracks=tracks(existing_tracks_idx,:);
    cur_track_centroid=cur_track(1,centroid1Col:centroid2Col);
    existing_tracks_centroids=existing_tracks(:,centroid1Col:centroid2Col);
    %get the tracks that are near our cell
    dist_to_existing_tracks=hypot(existing_tracks_centroids(:,1)-cur_track_centroid(1),...
        existing_tracks_centroids(:,2)-cur_track_centroid(2));
    split_candidates_idx=dist_to_existing_tracks<max_split_dist;
    split_candidates=existing_tracks(split_candidates_idx,:);
    if isempty(split_candidates)
        %no possible candidates to merge with so move on to next track
        untested_ids(1)=[];
        continue;
    end
    %sort the merge candidates by area
    [dummy sort_idx]=sort(split_candidates(:,areaCol));
    split_candidates=split_candidates(sort_idx,trackIDCol);

    %we have some tracks that may need to be merged with this track
    candidates_nr=size(split_candidates,1);
    for j=1:candidates_nr
        candidateID=split_candidates(j);
```

```
        candidate_track=tracks(tracks(:,trackIDCol)==candidateID,:);
        candidate_birth_time=cells_with_known_split_frames.get(candidateID);
        if isempty(candidate_birth_time)
            candidate_start_time=candidate_track(1,timeCol);
        else
            cell_life_span=track_start_time-candidate_birth_time;
            if (cell_life_span<min_time_for_split)
                %this cell has split too recently to be splitting again
                continue;
            end
            candidate_start_time=candidate_birth_time;
        end
        if (candidate_start_time>=track_start_time)
            %this track cannot be a parent of our track
            continue;
        end
        %get the times at which track exists
%         candidate_times=candidate_track(:,timeCol);
%         %get the times when both tracks exist
%         [common_times cur_track_common_idx
candidate_track_common_idx]=intersect(cur_track_times,candidate_times);
%         if (length(common_times)<4)
%             continue;
%         end
        %get the centroids at those times
%
cur_track_common_times_centroids=cur_track(cur_track_common_idx,centroid1Col:centroid2Col);
%
candidate_common_times_centroids=candidate_track(candidate_track_common_idx,centroid1Col:centroid
2Col);
%         %get the distance between tracks
%         dist_between_tracks=hypot(candidate_common_times_centroids(:,1)-
cur_track_common_times_centroids(:,1),...
%             candidate_common_times_centroids(:,2)-cur_track_common_times_centroids(:,2));
        candidate_track_median_area=median(candidate_track(:,areaCol));
        potential_split_idx=candidate_track(:,timeCol)==track_start_time;
        potential_split_params=candidate_track(potential_split_idx,:);
        cur_area=potential_split_params(1,areaCol);
        if (cur_area>1.1*candidate_track_median_area)
            %a cell is smaller right after splitting
            continue;
        end
        if (cur_area>max_split_area)
            %a cell is smaller right after splitting
            continue;
        end
        cur_ecc=potential_split_params(1,eccCol);
        if (cur_ecc<min_split_ecc)
            %nuclei are elongated right after splitting
            continue;
        end
        if (cur_ecc>max_split_ecc)
            %nuclei are not perfect lines
            continue;
        end
        split_cells=[split_cells; [candidateID curID track_start_time track_end_time]];
        cells_with_known_split_frames.put(candidateID,track_start_time);
        cells_with_known_split_frames.put(curID,track_start_time);
        break;
    end
    untested_ids(1)=[];
end

output_args.SplitCells=split_cells;

%end detectMitoticEvents
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% determineValidObjects.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function [valid_segmentation_ids valid_len
new_segmentation_idx]=determineValidObjects(segmentation_idx,...
    min_object_area,blob_1,blob_2)
%helper function. blobs smaller than our min threshold have to be unsegmented by assigning
%them to the nearest blob that is larger than minimum blob area

new_blob_areas=accumarray(segmentation_idx, 1);
segmentation_ids=[1:length(new_blob_areas)];
```

```
valid_new_blobs_idx=(new_blob_areas>min_object_area);
valid_areas=new_blob_areas(valid_new_blobs_idx);
valid_segmentation_ids=segmentation_ids(valid_new_blobs_idx);
new_segmentation_idx=[];
if isempty(valid_segmentation_ids)
    %no valid split
    valid_len=0;
    return;
end
valid_len=length(valid_segmentation_ids);
if (valid_len==1)
    %only one of the blobs will be large enough so we can't split
    return;
end
invalid_segmentation_ids=segmentation_ids(~valid_new_blobs_idx);
if (~isempty(invalid_segmentation_ids))
    invalid_areas=new_blob_areas(~valid_new_blobs_idx);
    %calculate the centroids of the new valid blobs
    valid_centroids=zeros(valid_len,2);
    for i=1:valid_len
        cur_segmentation_id=valid_segmentation_ids(i);
        cur_area=valid_areas(i);
        cur_segmentation_idx=segmentation_idx==cur_segmentation_id;
        segmented_idx_1=blob_1(cur_segmentation_idx);
        segmented_idx_2=blob_2(cur_segmentation_idx);
        valid_centroids(i,:)=[sum(segmented_idx_1./cur_area) sum(segmented_idx_2./cur_area)];
    end
    invalid_len=length(invalid_segmentation_ids);
    %calculate the centroids of the new invalid blobs
    %reassign the invalid segmentations to their nearest valid neighbors
    new_segmentation_idx=segmentation_idx;
    for i=1:invalid_len
        cur_segmentation_id=invalid_segmentation_ids(i);
        cur_area=invalid_areas(i);
        cur_segmentation_idx=segmentation_idx==cur_segmentation_id;
        segmented_idx_1=blob_1(cur_segmentation_idx);
        segmented_idx_2=blob_2(cur_segmentation_idx);
        cur_centroid=[sum(segmented_idx_1./cur_area) sum(segmented_idx_2./cur_area)];
        dist_to_valid_centroids=hypot(valid_centroids(:,1)-cur_centroid(1),...
            valid_centroids(:,2)-cur_centroid(2));
        [dummy closest_valid_centroid_idx]=min(dist_to_valid_centroids);
        nearest_valid_id=valid_segmentation_ids(closest_valid_centroid_idx);
        new_segmentation_idx(segmentation_idx==cur_segmentation_id)=nearest_valid_id;
    end
end

%end determineValidObjects
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% displayAncestryData.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=displayAncestryData(input_args)
% Usage
% This module is used to overlay cell outlines (using different colors to indicate different cell
generations) and cell labels on the original images after tracking and save the resulting image.
% Input Structure Members
% AncestryLayout - Matrix describing the order of the columns in the tracks matrix.
% CellsAncestry - Matrix containing the ancestry records for the cells in the image.
% CellsLabel - The label matrix containing the cell outlines for the current image.
% ColorMap - Color map to be used in drawing the cell outlines for each generation. Each
generation will use the next color in the color map until all colors have been used.  Afterwards,
the colors in the map are recycled.
% CurFrame - Integer containing the current frame number.
% CurrentTracks - The list of the tracks for the current image.
% DS - The directory separator to be used when generating file names (“\” for Windows, “/” for
Unix/Linux).
% Image - The original image which will be used to generate the image with overlayed outlines and
labels.
% ImageFileName - The root of the image file name to be used when generating the image file name
for the current image in combination with the current frame number.
% NumberFormat - A string indicating the number format of the file name to be used when saving
the overlayed image.
% ProlDir - Output directory where the resulting image will be saved.
% TracksLayout - Matrix describing the order of the columns in the tracks matrix.
% Output Structure Members
% None.


cur_img=input_args.Image.Value;
```

```
cur_tracks=input_args.CurrentTracks.Value;
cells_lbl=input_args.CellsLabel.Value;
cells_ancestry=input_args.CellsAncestry.Value;
curframe=input_args.CurFrame.Value;
cmap=input_args.ColorMap.Value;
number_fmt=input_args.NumberFormat.Value;

field_names=fieldnames(input_args);
if (max(strcmp(field_names,'LabelColorRGB')))
    lbl_color=input_args.LabelColorRGB.Value;
else
    lbl_color=[0 255 0];
end

img_sz=size(cur_img);
max_pxl=intmax('uint8');
imnorm_args.IntegerClass.Value='uint8';
imnorm_args.RawImage.Value=cur_img;
imnorm_output=imNorm(imnorm_args);
cur_img=imnorm_output.Image;
red_color=cur_img;
green_color=cur_img;
blue_color=cur_img;

prol_dir=input_args.ProlDir.Value;
img_file_name=input_args.ImageFileName.Value;
ds=input_args.DS.Value;
output1=[prol_dir ds img_file_name];

tracks_layout=input_args.TracksLayout.Value;
centroid1Col=tracks_layout.Centroid1Col;
centroid2Col=tracks_layout.Centroid2Col;
trackIDCol=tracks_layout.TrackIDCol;

ancestry_layout=input_args.AncestryLayout.Value;
ancestryIDCol=ancestry_layout.TrackIDCol;
generationCol=ancestry_layout.GenerationCol;

cur_cell_number=size(cur_tracks,1);
cell_lbl_id=zeros(cur_cell_number,1);
%i need to get the outlines of each individual cell since more than one
%cell might be in a blob
avg_filt=fspecial('average',[3 3]);
lbl_avg=imfilter(cells_lbl,avg_filt,'replicate');
lbl_avg=double(lbl_avg).*double(cells_lbl>0);
img_bounds=abs(double(cells_lbl)-lbl_avg);
img_bounds=img_bounds>0.1;
bounds_lbl=zeros(img_sz);
bounds_lbl(img_bounds)=cells_lbl(img_bounds);

%draw the cell boundaries
for j=1:cur_cell_number
    cur_centroid=cur_tracks(j,centroid1Col:centroid2Col);
    cell_id=cur_tracks(j,trackIDCol);
    cell_lbl_id=getLabelId(cells_lbl,cur_centroid);
    cell_generation=cells_ancestry(cells_ancestry(:,ancestryIDCol)==cell_id,generationCol);
    cell_bounds_idx=(bounds_lbl==cell_lbl_id);
    %draw in the red channel
    red_color(cell_bounds_idx)=max_pxl*cmap(cell_generation,1);
    green_color(cell_bounds_idx)=max_pxl*cmap(cell_generation,2);
    blue_color(cell_bounds_idx)=max_pxl*cmap(cell_generation,3);
end

%draw the cell labels
for j=1:cur_cell_number
    cur_centroid=cur_tracks(j,centroid1Col:centroid2Col);
    cell_id=cur_tracks(j,trackIDCol);

    text_img=text2im(num2str(cell_id));
    text_img=imresize(text_img,0.75,'nearest');
    text_length=size(text_img,2);
    text_height=size(text_img,1);
    rect_coord_1=round(cur_centroid(1)-text_height/2);
    rect_coord_2=round(cur_centroid(1)+text_height/2);
    rect_coord_3=round(cur_centroid(2)-text_length/2);
    rect_coord_4=round(cur_centroid(2)+text_length/2);
    if ((rect_coord_1<1)||(rect_coord_2>img_sz(1))||(rect_coord_3<1)||(rect_coord_4>img_sz(2)))
        continue;
    end
    [text_coord_1 text_coord_2]=find(text_img==0);
    %offset the text coordinates by the image coordinates in the (low,low)
```

```
    %corner of the rectangle
    text_coord_1=text_coord_1+rect_coord_1;
    text_coord_2=text_coord_2+rect_coord_3;
    text_coord_lin=sub2ind(img_sz,text_coord_1,text_coord_2);
    %write the text in blue
    red_color(text_coord_lin)=lbl_color(1);
    green_color(text_coord_lin)=lbl_color(2);
    blue_color(text_coord_lin)=lbl_color(3);

%     plot(cell_bounds{1}(:,2),cell_bounds{1}(:,1),'Color',cmap(cell_generation,:),'LineWidth',1)
%
text(cur_centroid(2),cur_centroid(1),num2str(cell_id),'Color','g','HorizontalAlignment','center',
...
%         'FontSize',5);
end
% toc
% hold off
% drawnow;
%write the combined channels as an rgb image
imwrite(cat(3,red_color,green_color,blue_color),[output1 num2str(curframe,number_fmt)
'.jpg'],'jpg');
% saveas(h1,[output1 num2str(curframe,'%03d') '.jpg'],'jpg');
output_args=[];

% end displayAncestryData
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% displayFrameMSD.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function displayFrameMSD()
%internal function for the manual tracking review GUI. It displays the
%mean square displacement for all the cells in the frame.

global mtr_gui_struct;

frame_msds=mtr_gui_struct.FrameMSDs;
averages_text=[mtr_gui_struct.AveragesText ' Frame MSD '
num2str(frame_msds(mtr_gui_struct.CurFrame),'%1.2f')];
set(mtr_gui_struct.AveragesTextHandle,'String',averages_text);

%end displayFrameMSD
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% displayImage.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=displayImage(input_args)
%displayImage module
%create or select figure FigureNr and display the image provided in Image

showmaxfigure(input_args.FigureNr.Value), imshow(input_args.Image.Value,[]);
output_args=[];

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% displayObjectOutlines.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=displayObjectOutlines(input_args)
% Usage
% This module is used to overlay cell outlines (using different colors to indicate different cell
generations) and cell labels on the original images after tracking and save the resulting image.
% Input Structure Members
% CellsLabel - The label matrix containing the cell outlines for the current image.
% CurrentTracks - The list of the tracks for the current image.
% FileRoot - The root of the image file name to be used when generating the image file name for
the current image in combination with the current frame number.
% Image - The original image which will be used to generate the image with overlayed outlines and
labels.
%  Output Structure Members
% None.


normalize_args.RawImage.Value=input_args.Image.Value;
int_class='uint8';
normalize_args.IntegerClass.Value=int_class;
```

```
normalize_output=imNorm(normalize_args);
cur_img=normalize_output.Image;
objects_lbl=input_args.ObjectsLabel.Value;
img_sz=size(objects_lbl);
max_pxl=intmax(int_class);

red_color=cur_img;
green_color=cur_img;
blue_color=cur_img;

field_names=fieldnames(input_args);
if (max(strcmp(field_names,'ShowIDs')))
    b_show_ids=input_args.ShowIDs.Value;
else
    b_show_ids=true;
end

%i need to get the outlines of each individual cell since more than one
%cell might be in a blob
avg_filt=fspecial('average',[3 3]);
lbl_avg=imfilter(objects_lbl,avg_filt,'replicate');
lbl_avg=double(lbl_avg).*double(objects_lbl>0);
img_bounds=abs(double(objects_lbl)-lbl_avg);
img_bounds=im2bw(img_bounds,graythresh(img_bounds));

obj_bounds_lin=find(img_bounds);
%draw the cell bounds in red
red_color(obj_bounds_lin)=max_pxl;
green_color(obj_bounds_lin)=0;
blue_color(obj_bounds_lin)=0;

%get the centroids
obj_centroids=getApproximateCentroids(objects_lbl);
obj_centroids(isnan(obj_centroids(:,1)),:)=[];
nr_objects=size(obj_centroids,1);

if (b_show_ids)
    for j=1:nr_objects
        cur_centroid=obj_centroids(j,:);
        cell_id=j;
        %add the cell ids
        text_img=text2im(num2str(cell_id));
        text_img=imresize(text_img,0.75,'nearest');
        text_length=size(text_img,2);
        text_height=size(text_img,1);
        rect_coord_1=round(cur_centroid(1)-text_height/2);
        rect_coord_2=round(cur_centroid(1)+text_height/2);
        rect_coord_3=round(cur_centroid(2)-text_length/2);
        rect_coord_4=round(cur_centroid(2)+text_length/2);
        if
((rect_coord_1<1)||(rect_coord_2>img_sz(1))||(rect_coord_3<1)||(rect_coord_4>img_sz(2)))
            continue;
        end
        [text_coord_1 text_coord_2]=find(text_img==0);
        %offset the text coordinates by the image coordinates in the (low,low)
        %corner of the rectangle
        text_coord_1=text_coord_1+rect_coord_1;
        text_coord_2=text_coord_2+rect_coord_3;
        text_coord_lin=sub2ind(img_sz,text_coord_1,text_coord_2);
        %write the text in green
        red_color(text_coord_lin)=max_pxl;
        green_color(text_coord_lin)=max_pxl;
        blue_color(text_coord_lin)=max_pxl;
    end
end

%write the combined channels as an rgb image
imwrite(cat(3,red_color,green_color,blue_color),input_args.FileName.Value);
output_args=[];

%end displayObjectOutlines
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% displayTracksData.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=displayTracksData(input_args)
% Usage
```

```
% This module is used to overlay cell outlines (using different colors to indicate different cell
generations) and cell labels on the original images after tracking and save the resulting image.
% Input Structure Members
% CellsLabel - The label matrix containing the cell outlines for the current image.
% CurFrame - Integer containing the current frame number.
% CurrentTracks - The list of the tracks for the current image.
% FileRoot - The root of the image file name to be used when generating the image file name for
the current image in combination with the current frame number.
% Image - The original image which will be used to generate the image with overlayed outlines and
labels.
% NumberFormat - A string indicating the number format of the file name to be used when saving
the overlayed image.
% TracksLayout - Matrix describing the order of the columns in the tracks matrix.
% Output Structure Members
% None.


normalize_args.RawImage.Value=input_args.Image.Value;
int_class='uint8';
normalize_args.IntegerClass.Value=int_class;
normalize_output=imNorm(normalize_args);
cur_img=normalize_output.Image;
cur_tracks=input_args.CurrentTracks.Value;
img_sz=size(cur_img);

field_names=fieldnames(input_args);
if (max(strcmp(field_names,'CellsLabel')))
    cells_lbl=input_args.CellsLabel.Value;
    show_lbl=true;
else
    show_lbl=false;
end

if (max(strcmp(field_names,'LabelColorRGB')))
    lbl_color=input_args.LabelColorRGB.Value;
else
    lbl_color=[0 255 0];
end

max_pxl=intmax(int_class);
tracks_layout=input_args.TracksLayout.Value;

red_color=cur_img;
green_color=cur_img;
blue_color=cur_img;

if (max(strcmp(field_names,'ShowIDs')))
    b_show_ids=input_args.ShowIDs.Value;
else
    b_show_ids=true;
end
if (max(strcmp(field_names,'IDList')))
    id_list=input_args.IDList.Value;
else
    id_list=[];
end
cur_cell_number=size(cur_tracks,1);

if (show_lbl)
    %i need to get the outlines of each individual cell since more than one
    %cell might be in a blob
    avg_filt=fspecial('average',[3 3]);
    lbl_avg=imfilter(cells_lbl,avg_filt,'replicate');
    lbl_avg=double(lbl_avg).*double(cells_lbl>0);
    img_bounds=abs(double(cells_lbl)-lbl_avg);
    img_bounds=im2bw(img_bounds,graythresh(img_bounds));

    cell_bounds_lin=find(img_bounds);
    %draw the cell bounds in red
    red_color(cell_bounds_lin)=max_pxl;
    green_color(cell_bounds_lin)=0;
    blue_color(cell_bounds_lin)=0;
end

if (b_show_ids)
    centroid1Col=tracks_layout.Centroid1Col;
    centroid2Col=tracks_layout.Centroid2Col;
    trackIDCol=tracks_layout.TrackIDCol;
    for j=1:cur_cell_number
        cur_centroid=cur_tracks(j,centroid1Col:centroid2Col);
        cell_id=cur_tracks(j,trackIDCol);
```

```
        if (~isempty(id_list))
            if (max(id_list==cell_id)==0)
                continue;
            end
        end

        %add the cell ids
        text_img=text2im(num2str(cell_id));
        text_img=imresize(text_img,0.75,'nearest');
        text_length=size(text_img,2);
        text_height=size(text_img,1);
        rect_coord_1=round(cur_centroid(1)-text_height/2);
        rect_coord_2=round(cur_centroid(1)+text_height/2);
        rect_coord_3=round(cur_centroid(2)-text_length/2);
        rect_coord_4=round(cur_centroid(2)+text_length/2);
        if
((rect_coord_1<1)||(rect_coord_2>img_sz(1))||(rect_coord_3<1)||(rect_coord_4>img_sz(2)))
            continue;
        end
        [text_coord_1 text_coord_2]=find(text_img==0);
        %offset the text coordinates by the image coordinates in the (low,low)
        %corner of the rectangle
        text_coord_1=text_coord_1+rect_coord_1;
        text_coord_2=text_coord_2+rect_coord_3;
        text_coord_lin=sub2ind(img_sz,text_coord_1,text_coord_2);
        %write the text in green
        red_color(text_coord_lin)=lbl_color(1);
        green_color(text_coord_lin)=lbl_color(2);
        blue_color(text_coord_lin)=lbl_color(3);
    end
end

%write the combined channels as an rgb image
imwrite(cat(3,red_color,green_color,blue_color),[input_args.FileRoot.Value
num2str(input_args.CurFrame.Value,...
    input_args.NumberFormat.Value) '.jpg'],'jpg');
output_args=[];

%end displayTracksData
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% displayTracksData3D.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=displayTracksData3D(input_args)
% Usage
% This module is used to overlay cell outlines (using different colors to indicate different cell
generations) and cell labels on the original images after tracking and save the resulting image.
% Input Structure Members
% CurFrame - Integer containing the current frame number.
% CurrentTracks - The list of the tracks for the current image.
% FileRoot - The root of the image file name to be used when generating the image file name for
the current image in combination with the current frame number.
% Image - The original image which will be used to generate the image with overlayed outlines and
labels.
% NumberFormat - A string indicating the number format of the file name to be used when saving
the overlayed image.
% ObjectsLabel - The label matrix containing the object outlines for the
% current image.
% ShowIDs - A boolean value indicating whether do display track IDs or not.
% TextSize - Optional value specifying the text size as a percentage of the
% default.
% TracksLayout - Matrix describing the order of the columns in the tracks matrix.
% Output Structure Members
% None.
%module to display and save images showing the object boundaries and object
%of a 3-D stack
img=input_args.Image.Value;
img_class=class(img);
max_pxl=intmax(img_class);
img_sz=size(img);
nr_slices=img_sz(3);
show_ids=input_args.ShowIDs.Value;
cur_tracks=input_args.CurrentTracks.Value;
objects_lbl=input_args.ObjectsLabel.Value;
tracks_layout=input_args.TracksLayout.Value;
centroid1Col=tracks_layout.Centroid1Col;
centroid3Col=tracks_layout.Centroid3Col;
nr_tracks=size(cur_tracks,1);
```

```
label_indices=zeros(nr_tracks,1);
temp_args.LabelMatrix.Value=objects_lbl;
temp_out=getCentroids3D(temp_args);
centroid_list=temp_out.Centroids;
clear temp_args;
clear temp_out;
field_names=fieldnames(input_args);
if (max(strcmp(field_names,'TextSize')))
    text_size=input_args.TextSize.Value;
else
    text_size=0.75;
end
nr_centroids=length(centroid_list);
for i=1:nr_tracks
    %the min distance between the cur_centroid and the centroid list gives the
    %position in the label which in turn provides the id of the object
    centroid_dist=sqrt(sum((centroid_list-
repmat(cur_tracks(i,centroid1Col:centroid3Col),nr_centroids,1)).^2,2));
    [dummy label_indices(i)]=min(centroid_dist);
end
track_ids=cur_tracks(:,tracks_layout.TrackIDCol);

output_tiff_name=[input_args.FileRoot.Value num2str(input_args.CurFrame.Value,...
    input_args.NumberFormat.Value) '.tiff'];
%i need to get the outlines of each individual cell since more than one
%cell might be in a blob
avg_filt=fspecial('average',[3 3]);

for i=1:nr_slices
    img_slice=(img(:,:,i));
    red_color=img_slice;
    green_color=img_slice;
    blue_color=img_slice;
    lbl_slice=objects_lbl(:,:,i);
    slice_ids=unique(lbl_slice(:));
    slice_ids(1)=[];
    slice_centroids=getApproximateCentroids(lbl_slice);
    slice_centroids=slice_centroids(slice_ids,:);
    %draw the object outlines in the slice
    lbl_avg=imfilter(lbl_slice,avg_filt,'replicate');
    lbl_avg=double(lbl_avg).*double(lbl_slice>0);
    obj_bounds=abs(double(lbl_slice)-lbl_avg);
    obj_bounds=im2bw(obj_bounds,graythresh(obj_bounds));
    obj_bounds_lin=find(obj_bounds);
    %draw the cell bounds in red
    red_color(obj_bounds_lin)=max_pxl;
    green_color(obj_bounds_lin)=0;
    blue_color(obj_bounds_lin)=0;
    objects_nr=length(slice_ids);
    if (show_ids)
        %add the label ids for each object
        for j=1:objects_nr
            cur_lbl_id=slice_ids(j);
            cur_centroid=slice_centroids(j,:);
            cur_track_id=track_ids(label_indices==cur_lbl_id);
            %write the label text
            text_img=text2im(num2str(cur_track_id));
            text_img=imresize(text_img,text_size,'nearest');
            text_length=size(text_img,2);
            text_height=size(text_img,1);
            rect_coord_1=round(cur_centroid(1)-text_height/2);
            rect_coord_2=round(cur_centroid(1)+text_height/2);
            rect_coord_3=round(cur_centroid(2)-text_length/2);
            rect_coord_4=round(cur_centroid(2)+text_length/2);
            if
((rect_coord_1<1)||(rect_coord_2>img_sz(1))||(rect_coord_3<1)||(rect_coord_4>img_sz(2)))
                continue;
            end
            [text_coord_1 text_coord_2]=find(text_img==0);
            %offset the text coordinates by the image coordinates in the (low,low)
            %corner of the rectangle
            text_coord_1=text_coord_1+rect_coord_1;
            text_coord_2=text_coord_2+rect_coord_3;
            text_coord_lin=sub2ind(img_sz,text_coord_1,text_coord_2);
            %write the text in green
            red_color(text_coord_lin)=0;
            green_color(text_coord_lin)=max_pxl;
            blue_color(text_coord_lin)=0;
        end
    end
    %save the slice to tiff
```

```
    if (i==1)

imwrite(cat(3,red_color,green_color,blue_color),output_tiff_name,'tif','Compression','none');
    else

imwrite(cat(3,red_color,green_color,blue_color),output_tiff_name,'tif','WriteMode','append','Comp
ression','none');
    end
end

output_args=[];

%end displayTracksData
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% displayVariable.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=displayVariable(input_args)
% Usage
% This module is used to display a variable name and its value.
% Input Structure Members
% Variable - The variable whose value is to be displayed.
% VariableName - The variable name to be displayed along with the variable value.
% Output Structure Members
% None.

output_args=[];
disp(input_args.VariableName.Value);
disp(input_args.Variable.Value);

%end displayVariable
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% distanceWatershed.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=distanceWatershed(input_args)
% Usage
% This module is used to compute the distance watershed of a binary image.
% Input Structure Members
% Image - The binary image for which the distance watershed will be computed.
% MedianFilterNhood - The size of the median filter which will be used to smooth the watershed.
% Output Structure Members
% LabelMatrix - The result of the distance watershed.

img_neg=~input_args.Image.Value;
img_dist=bwdist(img_neg);
img_dist=-img_dist;
img_dist(img_neg)=-Inf;
med_filt_nhood=input_args.MedianFilterNhood.Value;
output_args.LabelMatrix=watershed(medfilt2(img_dist,[med_filt_nhood med_filt_nhood]));

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% divideFunction.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=divideFunction(input_args)
%Usage
%This module divides two variables.
%
%Input Structure Members
%Var1 - The variable to be divided.
%Var2 - The divisor.
%ConvertToDouble - Boolean value. If true the variables are first converted
%to doubles.
%
%Output Structure Members
%Quotient - The result of the division.

var1=input_args.Var1.Value;
var2=input_args.Var2.Value;

if (input_args.ConvertToDouble.Value)
```

```
    output_args.Quotient=double(var1)./double(var2);
else
    output_args.Quotient=var1./var2;
end

%end addFunction
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% dpsimplify.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function [ps,ix] = dpsimplify(p,tol)

% Recursive Douglas-Peucker Polyline Simplification, Simplify
%
% [ps,ix] = dpsimplify(p,tol)
%
% dpsimplify uses the recursive Douglas-Peucker line simplification
% algorithm to reduce the number of vertices in a polyline
% according to specific tolerance. Works also for polylines in
% higher dimensions.
%
% In case of nans (missing vertex coordinates) dpsimplify treats each
% line separately.
%
% For additional information on the algorithm follow this link
% http://en.wikipedia.org/wiki/Ramer-Douglas-Peucker_algorithm
%
% Input arguments
%
%     p     polyline n*d matrix with n vertices in d
%           dimensions.
%     tol   tolerance (maximal euclidean distance allowed
%           between the new line and a vertex)
%
% Output arguments
%
%     ps    simplified line
%     ix    linear index of the vertices retained in p
%
% Examples
%
% 1. Simplify line
%
%     tol    = 1;
%     x      = 1:0.1:8*pi;
%     y      = sin(x) + randn(size(x));
%     p      = [x' y'];
%     ps     = dpsimplify(p,tol);
%
%     plot(p(:,1),p(:,2),'k')
%     hold on
%     plot(ps(:,1),ps(:,2),'r','LineWidth',2);
%     legend('original polyline','simplified')
%
% 2. Reduce polyline so that only knickpoints remain by
%    choosing a very low tolerance
%
%     p = [(1:10)' [1 2 3 2 4 6 7 8 5 2]'];
%     p2 = dpsimplify(p,eps);
%     plot(p(:,1),p(:,2),'k+--')
%     hold on
%     plot(p2(:,1),p2(:,2),'ro','MarkerSize',10);
%     legend('original line','knickpoints')
%
% 3. Simplify a circle
%
%     x = cos(0:0.01:2*pi);
%     y = sin(0:0.01:2*pi);
%     x(end+1) = x(1);
%     y(end+1) = y(1);
%     plot(x,y)
%     p2 = dpsimplify([x' y'],.1);
%     plot(x,y)
%     hold on
%     plot(p2(:,1),p2(:,2),'k','LineWidth',2);
%
%
%
%
```

```
% Author: Wolfgang Schwanghart, 10. Januar, 2009.
% w.schwanghart[at]unibas.ch


if nargin == 0
    help dpsimplify
    return
end

if nargin ~= 2
    error('wrong number of input arguments')
end

% error checking
if ~isscalar(tol)
    error('tol must be a scalar')
end



% nr of dimensions
dims    = size(p,2);

% ______________________________________
% what happens, when there are NaNs?
% NaNs divide polylines.
Inan      = any(isnan(p),2);

% if there is only one vertex
if size(p,1) == 1 || isempty(p);
    ps = p;
    ix = 1;

elseif any(Inan);
    if ~Inan(end)
        Inan = [Inan;true];
        p    = [p;nan(1,dims)];
    end

    if nargout == 2;
        % starting indices of respective polylines
        sIX = [true; diff(Inan)~=0];
        sIX(Inan) = false;
        sIX = find(sIX);
    end


    IX = [0;find(Inan)];
    IX = diff(IX);
    IX(IX == 1) = [];

    m   = IX-1;

    % if lines are diveded by nans single lines are stored in a cell array
    % cellfun is then used to call dpsimplify for each line
    c   = mat2cell(p(~Inan,:),m,dims);
    if nargout == 2;
        [ps,ix]   = cellfun(@(x) dpsimplify(x,tol),c,'uniformoutput',false);
        ix        = cellfun(@(x,six) x+six-1,ix,num2cell(sIX),'uniformoutput',false);
    else
        ps   = cellfun(@(x) dpsimplify(x,tol),c,'uniformoutput',false);
    end
    ps = cellfun(@(x) [x;nan(1,dims)],ps,'uniformoutput',false);

    ps   = cell2mat(ps);
    ps(end,:) = [];

    if nargout == 2;
        ix = cell2mat(ix);
        ixempty = ps(:,1);
        ixempty(~isnan(ixempty)) = ix;
        ix = ixempty;
    end


else

% ______________________________________
% if there are no nans than start the recursive algorithm

ixe     = size(p,1);
```

```
ixs     = 1;

% logical vector for the vertices to be retained
I   = true(ixe,1);

% anonymous function for starting point and end point comparision
compare = @(a,b) (a+eps >= b && a <= b) || ...
                 (a-eps <= b && a >= b);

% call recursive function
p   = simplifyrec(p,tol,ixs,ixe);
ps  = p(I,:);

% if desired return the index of retained vertices
if nargout == 2;
    ix  = find(I);
end

end

% _____________________________________________________________
function p  = simplifyrec(p,tol,ixs,ixe)

    % check if startpoint and endpoint are the same

    c1 = num2cell(p(ixs,:));
    c2 = num2cell(p(ixe,:));

    % same start and endpoint with tolerance (see anonymous function
    % compare)
    sameSE = all(cell2mat(cellfun(compare,c1(:),c2(:),'UniformOutput',false)));

    if sameSE;
        % calculate the shortest distance of all vertices between ixs and
        % ixe to ixs only
        if dims == 2;
            d    = hypot(p(ixs,1)-p(ixs+1:ixe-1,1),p(ixs,2)-p(ixs+1:ixe-1,2));
        else
            d    = sqrt(sum(bsxfun(@minus,p(ixs,:),p(ixs+1:ixe-1,:)).^2,2));
        end
    else
        % calculate shortest distance of all points to the line from ixs to ixe
        % subtract starting point from other locations
        pt = bsxfun(@minus,p(ixs+1:ixe,:),p(ixs,:));

        % end point
        a = pt(end,:)';

        beta = (a' * pt')./(a'*a);
        b    = pt-bsxfun(@times,beta,a)';
        if dims == 2;
            % if line in 2D use the numerical more robust hypot function
            d    = hypot(b(:,1),b(:,2));
        else
            d    = sqrt(sum(b.^2,2));
        end
    end

    % identify maximum distance and get the linear index of its location
    [dmax,ixc] = max(d);
    ixc  = ixs + ixc;

    % if the maximum distance is smaller than the tolerance remove vertices
    % between ixs and ixe
    if dmax <= tol;
        if ixs ~= ixe-1;
            I(ixs+1:ixe-1) = false;
        end
    % if not, call simplifyrec for the segments between ixs and ixc (ixc
    % and ixe)
    else
        % call recursive function
        p   = simplifyrec(p,tol,ixs,ixc);
        p   = simplifyrec(p,tol,ixc,ixe);

    end

end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% eccentricityFilterLabel.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=eccentricityFilterLabel(input_args)
% Usage
% This module is used to remove objects below and/or above a certain eccentricity from a label
matrix.
% Input Structure Members
% MaxEccentricity - Any object with an eccentricity higher than this value will be removed.
% MinEccentricity - Any object with an eccentricity lower than this value will be removed.
% ObjectsLabel - The label matrix from which the objects will be removed.
% Output Structure Members
% LabelMatrix - The filtered label matrix.


cells_lbl=input_args.ObjectsLabel.Value;
cells_props=regionprops(cells_lbl,'Eccentricity');
field_names=fieldnames(input_args);
if (max(strcmp(field_names,'MinEccentricity')))
    b_min=true;
else
    b_min=false;
end
if (max(strcmp(field_names,'MaxEccentricity')))
    b_max=true;
else
    b_max=false;
end
cells_eccentricity=[cells_props.Eccentricity];
cells_nr=length(cells_eccentricity);
valid_eccentricities_idx=false(1,cells_nr);
if (b_min)

valid_eccentricities_idx=valid_eccentricities_idx|(cells_eccentricity>=input_args.MinEccentricity
.Value);
end
if (b_max)

valid_eccentricities_idx=valid_eccentricities_idx|(cells_eccentricity<=input_args.MaxEccentricity
.Value);
end
if (min(valid_eccentricities_idx)==1)
    %no invalid objects return the same label back
    output_args.LabelMatrix=cells_lbl;
else
    valid_object_numbers=find(valid_eccentricities_idx);
    new_object_numbers=1:length(valid_object_numbers);
    %we will replace valid numbers with new and everything else will be set to
    %zero
    object_idx=cells_lbl>0;
    new_object_index=zeros(max(cells_lbl(object_idx)),1);
    new_object_index(valid_object_numbers)=new_object_numbers;
    new_cells_lbl=cells_lbl;
    %replace the old object numbers to prevent skips in numbering
    new_cells_lbl(object_idx)=new_object_index(cells_lbl(object_idx));
    output_args.LabelMatrix=new_cells_lbl;
end

%end eccentricityFilterLabel
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% enableMultipleSelection.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function enableMultipleSelection()
%helper function for manual segmentation review module.
global msr_gui_struct;
msr_gui_struct.SelectMultiple=get(msr_gui_struct.CheckBoxSelectMultipleHandle,'Value');

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% equalFunction.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=equalFunction(input_args)
%module to compare two values. outdated, use compareValues instead.
```

```
arg_1=input_args.Number1.Value;
arg_2=input_args.Number2.Value;
output_args.Boolean=(arg_1==arg_2);

%end addFunction
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% expandText.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function new_list=expandText(listbox_list,selection_idx,modules_list,modules_map,sublevel_depth)
%helper function for assayEditorGUI. used to show the submodules in the
%currently selected submodule list
%get the name of the module which contains this chain
i=1;
while 1
    selection_text=listbox_list{selection_idx-i};
    if
(strcmp(selection_text(1:9),'<html><b>')&&(getSelectionLevel(selection_text)==sublevel_depth))
        break;
    end
    i=i+1;
end
module_id=stripHTMLFromString(selection_text);
module_idx=modules_map.get(module_id);
module_struct=modules_list{module_idx};
selection_text=listbox_list{selection_idx};
chain_name=stripHTMLFromString(selection_text);
chain_idx=strcmp(chain_name,module_struct.ChainVars);
chain_name=module_struct.Chains(chain_idx);
%get the modules that are connected to this chain
submodules_idx=cellfun(@(x) strcmp(x.ChainName,chain_name),modules_list);
submodules_list=modules_list(submodules_idx);
modules_strings=formatModuleStrings(submodules_list);

if (selection_idx>1)
    new_list=[listbox_list(1:selection_idx); modules_strings];
else
    new_list=modules_strings;
end

if (selection_idx<length(listbox_list))
    new_list=[new_list; listbox_list((selection_idx+1):end)];
end

%end expandText
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% exportLabelToImage.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=exportLabelToImage(input_args)
%basic wrapper for the MATLAB imwrite function
%Input Structure Members
%Image - The image to be saved.
%FileName - The path where the image will be saved.
%Format - The image format.
%Output Structure Members
%None

img=uint16(input_args.Image.Value);
file_name=input_args.FileName.Value;
fmt=input_args.Format.Value;
imwrite(img,file_name,fmt,'Compression','none');

output_args=[];

%end exportLabelToImage
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% exportScriptVariables.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function exportScriptVariables(handles)
%helper function for assayEditorGUI. Used to export the current set of
%script variables to a file.
```

```
[file_name path_name]=uiputfile('*.m','Export Script Variables to:');
if (file_name==0)
    return;
end
file_text=addScriptVars(handles);
fid=fopen([path_name '/' file_name],'wt');
fwrite(fid,file_text);
fclose(fid);

%end exportScriptVariables
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% fillHoles.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=fillHoles(input_args)
%simple wrapper for MATLAB imfill function
output_args.Image=imfill(input_args.Image.Value,'holes');

%end fillHoles
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% flattenCellArray.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function new_array=flattenCellArray(cell_array)
%take an array that may contain cells and cell arrays and flatten it so it
%only contains cells
new_array={};
for i=1:length(cell_array)
    if (iscell(cell_array{i}))
        new_array=[new_array;flattenCellArray(cell_array{i})];
    else
        new_array=[new_array;cell_array(i)];
    end
end

%end flattenCellArray
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% forLoop.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=forLoop(function_struct)
% Usage
% This module is used to loop through a series of modules. The modules that will be looped
through are listed in the LoopFunctions input structure member.
% Input Structure Members
% EndLoop - The loop will stop when the loop counter reaches this value.
% IncrementLoop - The value by which the loop counter will be incremented every time one loop
cycle is completed.
% LoopFunctions - The list of module input structures that will be looped through.
% StartLoop - The loop counter will be initialized to this value.
% Any other arguments may be added to the FunctionArgs structure and they will be available to
the modules contained in the LoopFunctions structure member. This allows the loop modules to have
access to values generated by modules outside the loop.
% Output Structure Members
% Any values generated by the modules in the loop can be made available to modules outside the
loop by adding them to the KeepValues structure.

global dependencies_list;
global dependencies_index;

instance_name=function_struct.InstanceName;
cur_idx=dependencies_index.get(instance_name);
input_args=function_struct.FunctionArgs;
start_loop=input_args.StartLoop.Value;
end_loop=input_args.EndLoop.Value;
increment_loop=input_args.IncrementLoop.Value;
loop_functions=function_struct.LoopFunctions;

for i=start_loop:increment_loop:end_loop
    dependency_item=dependencies_list{cur_idx};
    %propagate any updated input args to the loop functions
    updateArgs(instance_name,dependency_item.FunctionArgs,'input');
    %update the value of LoopCounter to dependent functions
    updateArg(instance_name,'output','LoopCounter',i);
```

```
    for j=1:size(loop_functions,1)
        loop_function_instance_name=loop_functions{j}.InstanceName;
        callFunction(loop_function_instance_name,false);
    end
    output_args=makeOutputStruct(function_struct);
    output_args.LoopCounter=i;
    updateArgs(instance_name,output_args,'output');
end


for i=1:size(loop_functions,1)
    loop_function_instance_name=loop_functions{j}.InstanceName;
    clearArgs(loop_function_instance_name);
end

%end forloop
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% formatModuleItem.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function module_text=formatModuleItem(module_struct)
%helper function for assayEditorGUI. format a module item for display in the assay list box

module_level=module_struct.Level;
if (module_struct.IsParent)
    %add the module but also display the chains structure
    ws=repmat('  ',1,module_level-1);
    module_text={['<html><b>' ws module_struct.InstanceName '</b></html>']};
    ws=repmat('  ',1,module_level);
    for j=1:length(module_struct.ChainVars)
        module_text=[module_text; {['<html><i>' ws module_struct.ChainVars{j} '</i></html>']}];
    end
else
    ws=repmat('  ',1,module_level-1);
    module_text={['<html>' ws module_struct.InstanceName '</html>']};
end

%end formatModuleItem
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% formatModuleStrings.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function module_strings=formatModuleStrings(modules_list)
%helper function for assayEditorGUI. format module strings for display in listbox
module_strings={};
for i=1:length(modules_list)
    module_strings=[module_strings;formatModuleItem(modules_list{i})];
end

%end formatModuleString
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% freehandSelection.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=freehandSelection(input_args)
%simple wrapper for MATLAB imfreehand function.
%Input Structure Members
%ParentHandle - Handle to the image where the freehand region will be
%drawn.
%Output Structure Members
%RegionPixels - Array containing the selected pixels.

parent_handle=input_args.ParentHandle.Value;
region_handle=imfreehand(parent_handle);
freehand_api=iptgetapi(region_handle);
output_args.RegionPixels=freehand_api.getPosition();

%end freehandSelection
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% gaussianFilter.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function output_args=gaussianFilter(input_args)
% Usage
% This module is used to apply a gaussian blur to an image. See MATLAB
% documentation on fspecial for additional details.
% Input Structure Members
% KernelSize - The size of the kernel.
% StandardDev - The standard deviation of the gaussian distribution.
% Output Structure Members
% Image - The normalized image.

kernel_size=input_args.KernelSize.Value;
standard_dev=input_args.StandardDev.Value;
output_args.Image=imfilter(input_args.Image.Value, fspecial('gaussian',kernel_size,standard_dev),
'symmetric', 'conv');

%end function
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% gaussianPyramid.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=gaussianPyramid(input_args)
%simple wrapper module for the Matlab impyramid function
%Input Structure Members
%Image - The image to be processed.
%Output Structure Members
%Image - The filtered image.
output_args.Image=impyramid(input_args.Image.Value,'reduce');

%end gaussianPyramid
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% gcaWrapper.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=gcaWrapper(input_args)
%simple wrapper for gca MATLAB function
%Input Structure Members
%None
%Output Structure Members
%CurrentAxesHandle - The handle of the current figure.
output_args.CurrentAxesHandle=gca;

%end gcaWrapper
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% generateBinImgUsingGlobInt.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=generateBinImgUsingGlobInt(input_args)
% Usage
% This module is used to convert a grayscale image to binary using a global intensity threshold.
% Input Structure Members
% ClearBorder - If this value is set to true, objects that are within ClearBorderDist of the
image edges will be erased.
% ClearBorderDist - Objects that are within this distance from the edges of the image will be
erased if ClearBorder is set to true.
% Pixel intensities greater than the threshold intensity are converted to 1 in the binary image.
% Image - Grayscale image to be converted.
% IntensityThresholdPct - This is a percentage of the intensity range of the image. The threshold
intensity value is calculated as IntensityThresholdPct*double(max_pixel-min_pixel)+min_pixel.
%
% Output Structure Members
% Image - Resulting binary image.


max_pixel=max(input_args.Image.Value(:));
min_pixel=min(input_args.Image.Value(:));
brightnessPct=input_args.IntensityThresholdPct.Value;
threshold_intensity=brightnessPct*double(max_pixel-min_pixel)+min_pixel;
img_bw=input_args.Image.Value>threshold_intensity;
% img_bw=im2bw(img_to_proc,brightnessPct*graythresh(img_to_proc));
clear_border_dist=input_args.ClearBorderDist.Value;
if (input_args.ClearBorder.Value)
    if (clear_border_dist>1)
        img_bw(1:clear_border_dist-1,1:end)=1;
        img_bw(end-clear_border_dist+1:end,1:end)=1;
```

```
        img_bw(1:end,1:clear_border_dist-1)=1;
        img_bw(1:end,end-clear_border_dist+1:end)=1;
    end
    output_args.Image=imclearborder(img_bw);
else
    output_args.Image=img_bw;
end

%end generateBinImgUsingGlobInt
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% generateBinImgUsingGlobInt3D.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=generateBinImgUsingGlobInt3D(input_args)
% Usage
% This module is used to convert a 3-D grayscale image to binary using a global intensity
threshold by slices.
% Input Structure Members
% Pixel intensities greater than the threshold intensity are converted to 1 in the binary image.
% Image - Grayscale image to be converted.
% IntensityThresholdPct - This is a percentage of the intensity range of the image. The threshold
intensity value is calculated as IntensityThresholdPct*double(max_pixel-min_pixel)+min_pixel.
%
% Output Structure Members
% Image - Resulting binary image.


img=input_args.Image.Value;
img_sz=size(img);
img_bw=zeros(img_sz);
brightnessPct=input_args.IntensityThresholdPct.Value;
for i=1:img_sz(3)
    slice=img(:,:,i);
    max_pixel=max(slice(:));
    min_pixel=min(slice(:));
    threshold_intensity=brightnessPct*double(max_pixel-min_pixel)+min_pixel;
    img_bw(:,:,i)=slice>threshold_intensity;
end

output_args.Image=img_bw;

%end generateBinImgUsingGlobInt
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% generateBinImgUsingGradient.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=generateBinImgUsingGradient(input_args)
% Usage
% This module is used to convert a grayscale image to a binary image using values of the image
gradient.
% Input Structure Members
% ClearBorder - If this value is set to true, objects that are within ClearBorderDist of the
image edges will be erased.
% ClearBorderDist - Objects that are within this distance from the edges of the image will be
erased if ClearBorder is set to true.
% GradientThreshold - Areas in the gradient image where the gradient is higher than this value
will be set to 1 in the binary image.
% Image - Grayscale image to be converted.
% Output Structure Members
% Image - Resulting binary image.


[grad_x grad_y]=gradient(double(input_args.Image.Value));
grad_mag=sqrt(grad_x.^2+grad_y.^2);
img_bw=grad_mag>input_args.GradientThreshold.Value;
clear_border_dist=input_args.ClearBorderDist.Value;
if (input_args.ClearBorder.Value)
    if (clear_border_dist>1)
        img_bw(1:clear_border_dist-1,1:end)=1;
        img_bw(end-clear_border_dist+1:end,1:end)=1;
        img_bw(1:end,1:clear_border_dist-1)=1;
        img_bw(1:end,end-clear_border_dist+1:end)=1;
    end
    output_args.Image=imclearborder(img_bw);
else
```

```
    output_args.Image=img_bw;
end

end %end generateBinImgUsingGradient

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% generateBinImgUsingGradient3D.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=generateBinImgUsingGradient3D(input_args)
% Usage
% This module is used to convert a 3-D grayscale image to a binary image using values of the
image gradient by slices.
% Input Structure Members
% GradientThreshold - Areas in the gradient image where the gradient is higher than this value
will be set to 1 in the binary image.
% Image - Grayscale image to be converted.
% Output Structure Members
% Image - Resulting binary image.

img=input_args.Image.Value;
img_sz=size(img);
img_bw=zeros(img_sz);
grad_thresh=input_args.GradientThreshold.Value;
for i=1:img_sz(3)
    slice=img(:,:,i);
    [grad_x grad_y]=gradient(double(slice));
    grad_mag=sqrt(grad_x.^2+grad_y.^2);
    img_bw(:,:,i)=grad_mag>grad_thresh;
end

output_args.Image=img_bw;

end %end generateBinImgUsingGradient
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% generateBinImgUsingLocAvg.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=generateBinImgUsingLocAvg(input_args)
% Usage
% This module is used to convert a grayscale image to a binary image using local average values.
% Input Structure Members
% BrightnessThresholdPct - This value indicates what percentage of the local average value will
be used to threshold the image.  If the pixel intensity is higher than this value times the local
average value the  corresponding pixel in the binary image will be set to one.
% ClearBorder - If this value is set to true, objects that are within ClearBorderDist of the
image edges will be erased.
% ClearBorderDist - Objects that are within this distance from the edges of the image will be
erased if ClearBorder is set to true.
% Image - Grayscale image to be converted.
% Strel - Filter type used to generate the local average image. Currently 'circular' is the only
value supported.
% StrelSize - Size of the local neighborhood used to calculate the average for each pixel in the
image.
% Output Structure Members
% Image - Resulting binary image.


avg_filter=fspecial(input_args.Strel.Value,input_args.StrelSize.Value);
img_avg=imfilter(input_args.Image.Value,avg_filter,'replicate');
img_bw=input_args.Image.Value>(input_args.BrightnessThresholdPct.Value*img_avg);
if (input_args.ClearBorder.Value)
    clear_border_dist=input_args.ClearBorderDist.Value;
    if (clear_border_dist>1)
        img_bw(1:clear_border_dist-1,1:end)=1;
        img_bw(end-clear_border_dist+1:end,1:end)=1;
        img_bw(1:end,1:clear_border_dist-1)=1;
        img_bw(1:end,end-clear_border_dist+1:end)=1;
    end
    output_args.Image=imclearborder(img_bw);
else
    output_args.Image=img_bw;
end

%end generateBinImgUsingLocAvg
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% generateBinImgUsingLocAvg3D.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=generateBinImgUsingLocAvg3D(input_args)
%module to convert z-stack grayscale image to a binary image using the local
%average values by slices
avg_filter=fspecial(input_args.Strel.Value,input_args.StrelSize.Value);
img=input_args.Image.Value;
img_sz=size(img);
img_bw=zeros(img_sz);
brightness_pct=input_args.BrightnessThresholdPct.Value;
for i=1:img_sz(3)
    slice=img(:,:,i);
    slice_avg=imfilter(slice,avg_filter,'replicate');
    img_bw(:,:,i)=slice>(brightness_pct*slice_avg);
end

output_args.Image=img_bw;

%end generateBinImgUsingLocAvg
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getApproximateCentroids.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function cell_centroids=getApproximateCentroids(cells_lbl)
%helper function. these centroids are exact for shapes without disconnects but only
%approximate the centroids of shapes with disconnects. it is however close
%enough for matching purposes
lbl_idx=cells_lbl>0;
[cells_1 cells_2]=find(lbl_idx);
cell_coords_1=accumarray(cells_lbl(lbl_idx),cells_1);
cell_coords_2=accumarray(cells_lbl(lbl_idx),cells_2);
cell_areas=accumarray(cells_lbl(lbl_idx),1);
cell_centroids=[cell_coords_1 cell_coords_2]./[cell_areas cell_areas];

%end getApproximateCentroids
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getArrayVal.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=getArrayVal(input_args)
% Usage
% This module returns a value or set of values from an array.
% Input Structure Members
% Array - Array from which the value is to be extracted.
% Index - The index of the values to be returned.
% Output Structure Members
% ArrayVal - Set of values extracted from the array.


array=input_args.Array.Value;
output_args.ArrayVal=array(input_args.Index.Value,:);

%end getArrayVal
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getAssaysList.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function assays_list=getAssaysList()
%helper function for assayEditorGUI. build a list of assays in the current directory
%get alll the .m files

m_files=dir('*.m');
assays_list={};
%check which ones are modules
%modules are recognized by the presence of an input_args and an output_args
%structure
for i=1:length(m_files)
    %these files are not assays
    if strcmp(m_files(i).name,'getAssaysList.m')
        continue;
    end
    if strcmp(m_files(i).name,'addToFunctionChain.m')
        continue;
```

```
    end
    if strcmp(m_files(i).name,'aeMenuOpenAssay.m')
        continue;
    end
    if strcmp(m_files(i).name,'saveAssay.m')
        continue;
    end
    file_text=fileread(m_files(i).name);
    %remove comments since they may contain the strings we're searching for
    module_text= regexprep(file_text, '%[^%\n]*\n', '');
    if isempty(strfind(module_text,'addToFunctionChain'))
        continue;
    end
    assays_list=[assays_list;m_files(i).name];
end

%end getAssayList
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getBetterMatchToTrack.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function
best_match_id=getBetterMatchToTrack(cur_track,cells_shape_params,cells_centroids,cells_ids,prev_t
racks,matching_groups,...
    tracks_layout, cells_lbl,
prev_cells_lbl,relevant_params_idx,param_weights,unknown_param_weights,unknown_ranking_order)
%helper function for CA tracking algorithm. figure out which cell of a pair is a better match for
the track this
%should only be used with cell pairs otherwise is meaningless
assert(size(cells_shape_params,1)==2);
%figure out which cell is a better match for this track

areaCol=tracks_layout.AreaCol;
centroid1Col=tracks_layout.Centroid1Col;
centroid2Col=tracks_layout.Centroid2Col;
trackIDCol=tracks_layout.TrackIDCol;
groupIDCol=tracks_layout.MatchGroupIDCol;
solCol=tracks_layout.SolCol;


track_centroid=cur_track(centroid1Col:centroid2Col);
dist_to_cells=hypot(cells_centroids(:,1)-track_centroid(1), cells_centroids(:,2)-
track_centroid(2));
if (isempty(prev_tracks))
    prev_track_centroid=[];
else

prev_track_centroid=prev_tracks(prev_tracks(:,trackIDCol)==cur_track(:,trackIDCol),centroid1Col:c
entroid2Col);
end
if isempty(prev_track_centroid)
    track_params=[min(dist_to_cells) cur_track(areaCol:solCol)];
    cells_params=[dist_to_cells cells_shape_params(:,1:solCol-areaCol+1)];
    b_use_direction=false;
else
    prev_angle=atan2((track_centroid(2)-prev_track_centroid(2)), (track_centroid(1)-
prev_track_centroid(1)));
    possible_angles=atan2((cells_centroids(:,2)-track_centroid(2)), (cells_centroids(:,1)-
track_centroid(1)));
    track_params=[min(dist_to_cells) prev_angle cur_track(areaCol:solCol)];
    cells_params=[dist_to_cells possible_angles cells_shape_params(:,1:solCol-areaCol+1)];
    b_use_direction=true;
end

group_idx=cur_track(:,groupIDCol);
if (group_idx==0)
    ranking_order=unknown_ranking_order;
else
    ranking_order=matching_groups(group_idx,:);
end
pair_scores=getPairScoresToSingle(cells_params,track_params,b_use_direction,unknown_param_weights
,...
    param_weights,ranking_order,group_idx,relevant_params_idx);
best_match_id=[];
if (pair_scores(1)==pair_scores(2))
    %neither cell is a better match to the track
    return;
end
```

```
track_lbl_id=getLabelId(prev_cells_lbl, track_centroid);
if (pair_scores(1)<pair_scores(2))
    if (~pathGoesThroughACell(cells_lbl, prev_cells_lbl, cells_ids(1), track_lbl_id, 0))
        best_match_id=cells_ids(1);
        return;
    end
    if (~pathGoesThroughACell(cells_lbl, prev_cells_lbl, cells_ids(2), track_lbl_id, 0))
        best_match_id=cells_ids(2);
        return;
    end
else
    if (~pathGoesThroughACell(cells_lbl, prev_cells_lbl, cells_ids(2), track_lbl_id, 0))
        best_match_id=cells_ids(2);
        return;
    end
    if (~pathGoesThroughACell(cells_lbl, prev_cells_lbl, cells_ids(1), track_lbl_id, 0))
        best_match_id=cells_ids(1);
        return;
    end
end

%end getBetterMatchToTrack
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% getBlobClusters.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [blob_clusters linkage_clusters]=getBlobClusters(blob_coord, cluster_dist)
%helper function for cluster segmentation module. figure out what's the best nr of clusters for
splitting
max_clusters=20;
blob_length=length(blob_coord);
if (length(blob_coord)<=max_clusters)
    blob_clusters=1;
    linkage_clusters=ones(blob_length,1);
    return;
end
% RMSSTD=zeros(max_clusters,1);
% nr_points=length(blob_coord);

blob_dist=pdist(blob_coord);
%tested all linkage params - this works best
blob_linkage=linkage(blob_dist,'average');
% linkage_clusters=cluster(blob_linkage,'criterion','distance','cutoff',12); %10x kam cells
linkage_clusters=cluster(blob_linkage,'criterion','distance','cutoff',cluster_dist); %20x ht1080
cells
max_clusters=max(linkage_clusters(:));
blob_clusters=max_clusters;
return;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% getCentroids.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=getCentroids(input_args)
% Usage
% This module returns a list of centroids for the objects in a label matrix.
% Input Structure Members
% LabelMatrix - Label matrix from which the object centroids will be calculated.
% Output Structure Members
% Centroids - The list of centroids extracted from the label matrix.

objects_lbl=input_args.LabelMatrix.Value;
objects_idx=objects_lbl>0;
[objects_1 objects_2]=find(objects_idx);
object_coords_1=accumarray(objects_lbl(objects_idx),objects_1);
object_coords_2=accumarray(objects_lbl(objects_idx),objects_2);
object_areas=accumarray(objects_lbl(objects_idx),1);
objects_centroids=[object_coords_1 object_coords_2]./[object_areas object_areas];
output_args.Centroids=objects_centroids;

%end getCentroids
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% getCentroids3D.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=getCentroids3D(input_args)
```

```
% Usage
% This module returns a list of centroids for the objects in a 3D label matrix.
% Input Structure Members
% LabelMatrix - Label matrix from which the object centroids will be calculated.
% Output Structure Members
% Centroids - The list of centroids extracted from the label matrix.

objects_lbl=input_args.LabelMatrix.Value;
objects_props=regionprops(objects_lbl,'Centroid');
objects_centroids=[objects_props.Centroid]';
centr_len=size(objects_centroids,1);
objects_centroids=[objects_centroids(2:3:centr_len) objects_centroids(1:3:centr_len)
objects_centroids(3:3:centr_len)];
output_args.Centroids=objects_centroids;

%end getCentroids3D
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% getConvexObjects.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=getConvexObjects(input_args)
% Usage
% This module is used to find and return the index of convex objects in a binary image. The
object outlines are simplified using a Douglas-Pecker algorithm to prevent detection of
insignificant convexities.
% Input Structure Members
% ApproximationDistance - This value represents the minimum distance between the approximated
outline and the real one. Increasing this value makes the contours simpler but less like the
original outlines.
% Image - Binary image to be processed.
%
% Output Structure Members
% ConvexObjectsIndex - List containing the index of the convex objects. The index of each object
is based on performing a bwlabeln operation on the binary image.

obj_bounds=bwboundaries(input_args.Image.Value);
obj_nr=length(obj_bounds);
convex_objects_idx=false(obj_nr,1);
pol_simplify=input_args.ApproximationDistance.Value;

for i=1:obj_nr
    cur_bound=obj_bounds{i};
    pol_points_1=dpsimplify([cur_bound(:,1) cur_bound(:,2)],pol_simplify);
    %to get the angles inside the polygon we need to create vectors that
    %have as base each vertex and as end the next and previous vertice and
    %then translate their base to the origin. because we need each vertex
    %as a base twice we traverse the polygon in both directions
    pol_points_2=flipud(pol_points_1);
    %now get the vectors
    pol_vect_1=diff(pol_points_1);
    pol_vect_2=diff(pol_points_2);
    pol_len=length(pol_vect_1);
    if (pol_len<=3)
        %a triangle is a convex polygon
        convex_objects_idx(i)=true;
        continue;
    end
    %index to match the coord of pol_dist_2 to pol_dist_1
    pol_idx_2=pol_len-[1:pol_len-1]+1;
    %element 1 of pol_vect_1 matches with element 1 of pol_vect_2 ie if you
    %have 123451 and 154321 1 matches 1 but the rest have to be matched
    pol_idx_2=[1 pol_idx_2];
    %now match them
    pol_vect_2(:,1)=pol_vect_2(pol_idx_2,1);
    pol_vect_2(:,2)=pol_vect_2(pol_idx_2,2);
    u2=pol_vect_1(:,1);
    u1=pol_vect_2(:,1);
    v2=pol_vect_1(:,2);
    v1=pol_vect_2(:,2);
    % Assuming a = [x1,y1] and b = [x2,y2] are two vectors with their bases at the
    % origin, the non-negative angle between them measured counterclockwise from a to b is given
by
    %pol_angles=mod(atan2(x1*y2-x2*y1,x1*x2+y1*y2),2*pi);
    %dot product gives cos and cross product gives sin
    %the angle measured on the outside is 2*pi-angle measured on the inside
    %of the polygon
    outside_angles=2*pi-mod(atan2(u1.*v2-u2.*v1,u1.*u2+v1.*v2),2*pi);
    %get the convex angles where we might cut
    concave_idx=find(outside_angles<pi,1);
```

```
    if (isempty(concave_idx))
        %convex polygon - no concave angles
        convex_objects_idx(i)=true;
    end
end

output_args.ConvexObjectsIndex=convex_objects_idx;

%end getConvexObjects
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getCurrentTracks.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=getCurrentTracks(input_args)
% Usage
% Module to extract a subset of tracks out of the tracks matrix starting with the current frame.
% Input Structure Members
% CurFrame - The current frame index.
% FrameStep - How many frames to skip when reading the track subset. If every frame is to be read
set this value to 1.
% MaxMissingFrames - This value indicates if tracks not present in the current frame should be
included in the track subset and if so how many frames away from the current frame a track is
allowed to be and still be included in the subset. Setting this value to zero ensures that only
tracks present in the current frame are included.
% OffsetFrame - The number of frames from the current frame the subset should include. This value
can be a positive or negative integer.
% TimeCol - Index of the time column in the tracks matrix.
% TimeFrame - The amount of time elapsed between each frame.
% TrackIDCol - Index of the track ID column in the tracks matrix.
% Tracks - Matrix containing the set of tracks from which the subset is to be extracted.
% Output Structure Members
% Tracks - The subset of tracks extracted from the tracks matrix.

tracks=input_args.Tracks.Value;
frame_step=input_args.FrameStep.Value;
offset_frame=input_args.OffsetFrame.Value;
startframe=input_args.CurFrame.Value+frame_step*offset_frame;
offset_dir=sign(offset_frame);
timeframe=input_args.TimeFrame.Value;
timeCol=input_args.TimeCol.Value;
max_missing_frames=input_args.MaxMissingFrames.Value;
track_id_col=input_args.TrackIDCol.Value;
cur_tracks=tracks(tracks(:,timeCol)==(startframe-1)*timeframe,:);
track_ids=cur_tracks(:,track_id_col);
min_time=min(tracks(:,timeCol));
if (max_missing_frames<1)
    max_missing_frames=1;
end
for i=frame_step:frame_step:(frame_step*max_missing_frames)
    cur_time=(startframe+offset_dir*i-1)*timeframe;
    if (cur_time<min_time)
        break;
    end
    new_tracks_idx=tracks(:,timeCol)==cur_time;
    new_track_ids=tracks(new_tracks_idx,track_id_col);
    [diff_track_ids diff_track_idx]=setdiff(new_track_ids,track_ids);
    if isempty(diff_track_ids)
        continue;
    end
    new_tracks=tracks(new_tracks_idx,:);
    diff_tracks=new_tracks(diff_track_idx,:);
    cur_tracks=[cur_tracks; diff_tracks];
    track_ids=[track_ids; diff_track_ids];
end
output_args.Tracks=cur_tracks;
%end getCurrentTracks
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getCurrentUnassignedCell.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=getCurrentUnassignedCell(input_args)
%module to retrieve the current unassigned cell from the list
output_args.CellID=input_args.UnassignedCells.Value(1);

%end getCurrentUnassignedCell
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getDaughterCells.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [new_cells_ids parent_ids merge_with_parent new_cells_centroids
parent_cells_centroids]=getDaughterCells(all_tracks, cur_tracks, ...
    cur_tracks_ids, cur_time, max_dist, new_cells_ids, new_tracks_idx, min_track_frames,
cell_areas, cells_lbl,...
    med_area)
%helper function. use a series of filters to determine potential daughter
%cells
new_tracks=cur_tracks(new_tracks_idx,:);
new_tracks_sz=size(new_tracks,1);
merge_with_parent=false(new_tracks_sz,1);
parent_ids=zeros(new_tracks_sz,1);
new_cells_centroids=zeros(new_tracks_sz,2);
parent_cells_centroids=zeros(new_tracks_sz,2);
cur_tracks_sz=size(cur_tracks,1);
cur_tracks_cent=cur_tracks(:,1:2);
for i=1:new_tracks_sz
    cur_centroid=new_tracks(i,1:2);
    cur_lbl_id=cells_lbl(round(cur_centroid(1)),round(cur_centroid(2)));
    cur_centroid_mat=repmat(cur_centroid,cur_tracks_sz,1);
    cur_centroid_dist=hypot(cur_centroid_mat(:,1)-cur_tracks_cent(:,1),cur_centroid_mat(:,2)-
cur_tracks_cent(:,2));
    cur_centroid_dist_idx=(cur_centroid_dist<max_dist)&(cur_centroid_dist>0.1);
    cur_centroid_dist=cur_centroid_dist(cur_centroid_dist_idx);
    nearby_centroids=cur_tracks_cent(cur_centroid_dist_idx,:);
    if (isempty(cur_centroid_dist))
        continue;
    end
    nearby_cell_ids=cur_tracks_ids(cur_centroid_dist_idx);
    %remove cells that are new from the list of nearby cells
    keep_cells_idx=~ismember(nearby_cell_ids,new_cells_ids);
    nearby_cell_ids=nearby_cell_ids(keep_cells_idx);
    cur_centroid_dist=cur_centroid_dist(keep_cells_idx);
    nearby_centroids=nearby_centroids(keep_cells_idx,:);
    if (isempty(nearby_cell_ids))
        continue;
    end
%     daughter_track_idx=all_tracks(:,4)==new_cells_ids(i);
%     daughter_track_centroids=all_tracks(daughter_track_idx,1:2);
%     daughter_track_length=size(daughter_track_centroids,1);
    [cur_centroid_dist dist_sort_idx]=sort(cur_centroid_dist);
    nearby_cell_ids=nearby_cell_ids(dist_sort_idx);
    nearby_centroids=nearby_centroids(dist_sort_idx,:);
    nearby_cells_sz=size(nearby_cell_ids,1);
    shrink_ratio=2*ones(nearby_cells_sz,1);
    shrink_ratio2=2*ones(nearby_cells_sz,1);
    daughter_area=cell_areas{new_cells_ids(i)};
    %use only the last area recorded for this id
    daughter_area=daughter_area(end);
    for j=1:size(nearby_cell_ids,1)
        if (j==1)
            nearby_lbl_id=cells_lbl(round(nearby_centroids(1,1)),round(nearby_centroids(1,2)));
            if (nearby_lbl_id==cur_lbl_id)
                parent_ids(i)=nearby_cell_ids(j);
                merge_with_parent(i)=true;
                new_cells_centroids(i,:)=cur_centroid;
                parent_cells_centroids(i,:)=nearby_centroids(1,:);
                break;
            end
        end
        parent_track_idx=(all_tracks(:,4)==nearby_cell_ids(j))&(all_tracks(:,3)<cur_time);
        parent_track_centroids=all_tracks(parent_track_idx,1:2);
        parent_track_length=size(parent_track_centroids,1);
        if (parent_track_length<min_track_frames)
            continue;
        end
        pot_parent_areas=cell_areas{nearby_cell_ids(j)};
        parent_areas_len=length(pot_parent_areas);
        if (parent_areas_len<2)
            continue;
        end
        %the previous parent area should not be smaller than 1.5 times its daughter
        %cells
        if (parent_areas_len>3)
            prev_parent_area=pot_parent_areas(end-2);
        else
```

```
            prev_parent_area=pot_parent_areas(end-1);
        end
        cur_parent_area=pot_parent_areas(end);
        thresh_area=0.6*med_area;
        if ((cur_parent_area>thresh_area)||(daughter_area>thresh_area))
            %if either of the daughter cells is too big it's not a real
            %mitotic event
            continue;
        end
        if
(((2*prev_parent_area)<(cur_parent_area+daughter_area))||(cur_parent_area>3*daughter_area)...
                ||(daughter_area>3*cur_parent_area))
            %not a true mitotic event just a new nucleus next to a
            %preexisting nucleus
            continue;
        end
%         if (parent_areas_len>3)
%             test_areas=pot_parent_areas(end-2:end);
%             shrink_ratio(j)=test_areas(3)/test_areas(2);
%             shrink_ratio2(j)=test_areas(2)/test_areas(1);
%         else
%             shrink_ratio(j)=pot_parent_areas(end)/pot_parent_areas(end-1);
%         end
%         parent_track_idx=(all_tracks(:,4)==nearby_cell_ids(j))&(all_tracks(:,3)>=cur_time);
%         parent_track_centroids=all_tracks(parent_track_idx,1:2);
%         parent_track_length=size(parent_track_centroids,1);
%         if (parent_track_length<min_track_time)
%             continue;
%         end
%         parent_idx=find(cells_ids==nearby_cell_ids(j),1);
%         parent_gen=cells_generations(parent_idx);
%         if (parent_gen>1)
%             %cells higher than first generations need to have been around
%             %at least for a little while before they split
%
prev_track_idx=find((all_tracks(:,4)==nearby_cell_ids(j))&(all_tracks(:,3)<cur_time));
%             if (length(prev_track_idx)<min_track_time)
%                 continue;
%             end
%         end
%
%         min_track_length=min(daughter_track_length,parent_track_length);
%         p_d_centroid_dist=hypot(daughter_track_centroids(1:min_track_length,1)-...
%             parent_track_centroids(1:min_track_length,1),
daughter_track_centroids(1:min_track_length,2)...
%             -parent_track_centroids(1:min_track_length,2));
%         max_centroid_dist=max(p_d_centroid_dist);
%         %cells separate after splitting-it's not just a multinucleated cell
% %         if (max_centroid_dist>1.2*max_dist)
%             parent_ids(i)=nearby_cell_ids(j);
%             break;
% %         end
        parent_ids(i)=nearby_cell_ids(j);
        break;
    end
%     [parent_ratio parent_idx]=min(shrink_ratio);
%     if (parent_ratio>0.9)
%         [parent_ratio parent_idx]=min(shrink_ratio2);
%         if (parent_ratio>0.9)
%             continue;
%         end
%     end
end

%end function
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% getFileInfo.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=getFileInfo(input_args)
% Usage
% This module is used to extract the file name, extension and directory from the absolute path.
% Input Structure Members
% PathName - Absolute path name from which file name, extension and directory will be extracted.
% Output Structure Members
% DirName - The extracted directory name.
% FileName - The extracted file name.
% ExtName - The extracted extension name.
```

```
path_name=input_args.PathName.Value;
dir_idx=strfind(path_name,'\');
dir_idx=[dir_idx strfind(path_name,'/')];
dir_idx=sort(dir_idx);
dir_idx=dir_idx(end);
ext_idx=strfind(path_name,'.');
ext_idx=ext_idx(end);

output_args.DirName=path_name(1:dir_idx);
output_args.FileName=path_name((dir_idx+1):(ext_idx-1));
output_args.ExtName=path_name(ext_idx:end);

%end getFileInfo
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% getInputArgs.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function input_args=getInputArgs(module_path)
%helper function for assayEditorGUI. extract the input args from the specified module

module_name=module_path(1:(end-2));
switch(module_name)
    case 'forLoop'
        input_args={'EndLoop','IncrementLoop','StartLoop'};
    case 'if_statement'
        input_args={'TestVariable'};
    case 'whileLoop'
        input_args={'TestFunction'};
    otherwise
        module_text=fileread(module_path);
        search_pattern='input_args.(\w*).Value';
        input_tokens=regexp(module_text,search_pattern,'tokens');
        input_args=cellfun(@(x) x{1},input_tokens,'UniformOutput',false);
        input_args=unique(input_args);
end

%end getInputArgs
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% getIntegratedIntensities.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=getIntegratedIntensities(input_args)
%module to calculate the integrated intensity of objects in a
%label matrix.
%Input Structure Members
%LabelMatrix - Label matrix from which the object properties will be
%extracted
%Output Structure Members
%IntegratedIntensities - Matrix containing the integrated intensity values
%for each object

objects_lbl=input_args.ObjectsLabel.Value;
intensity_img=input_args.IntensityImage.Value;
objects_idx=(objects_lbl>0);

output_args.IntegratedIntensities=accumarray(objects_lbl(objects_idx),intensity_img(objects_idx))
;

%end getIntegratedIntensities
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% getLabelId.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function lbl_id=getLabelId(cells_lbl, cell_centroid)
%helper function to get the label id of a cell whose centroid is known. main reason for this
function is that the centroid may fall outside the
%cell body
lbl_id=0;
xtend_sz=0;
cell_coord=round(cell_centroid);
img_sz=size(cells_lbl);
while(lbl_id==0)
    min_1=cell_coord(1)-xtend_sz;
    if (min_1<1)
```

```
        min_1=1;
    end
    max_1=cell_coord(1)+xtend_sz;
    if (max_1>img_sz(1))
        max_1=img_sz(1);
    end
    min_2=cell_coord(2)-xtend_sz;
    if (min_2<1)
        min_2=1;
    end
    max_2=cell_coord(2)+xtend_sz;
    if (max_2>img_sz(2))
        max_2=img_sz(2);
    end
    lbl_vals=cells_lbl(min_1:max_1,min_2:max_2);
    lbl_vals=lbl_vals(:);
    lbl_vals(lbl_vals==0)=[];
    if (isempty(lbl_vals))
        xtend_sz=xtend_sz+1;
    else
        lbl_id=mode(lbl_vals);
    end
end

%end getLabelID
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% getMatchingGroupMeans.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=getMatchingGroupMeans(input_args)
%helper function for CA tracking module. get the mean param values for each matching group
tracks=input_args.Tracks.Value;
tracks_layout=input_args.TracksLayout.Value;
group_id_col=tracks_layout.MatchGroupIDCol;
start_params_col=tracks_layout.AreaCol;
end_params_col=tracks_layout.SolCol;

tracks_no_zeroth_group=tracks(tracks(:,group_id_col)~=0,:);
if (isempty(tracks_no_zeroth_group))
    output_args.MatchingGroupStats=[];
    return;
end
group_ids=tracks_no_zeroth_group(:,group_id_col);
nr_groups=max(group_ids);
nr_params=end_params_col-start_params_col+1;
group_stats=zeros(nr_groups,nr_params);
for i=1:nr_params
    group_stats(:,i)=accumarray(group_ids,tracks_no_zeroth_group(:,start_params_col+i-
1),[],@mean);
end
output_args.MatchingGroupStats=group_stats;

%end getMatchingGroupMeans
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% getMaxTrackID.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=getMaxTrackID(input_args)
% Usage
% This module is used to return the current maximum track ID from a track matrix.
% Input Structure Members
% TrackIDCol - Index of the track ID column in the tracks matrix.
% Tracks - Matrix containing the set of tracks.
% Output Structure Members
% MaxTrackID - The maximum track ID.

output_args.MaxTrackID=max(input_args.Tracks.Value(:,input_args.TrackIDCol.Value));

%end getMaxTrackID
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% getMinDistanceBetweenPointSets.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [min_distance min_distance_index interset_edge_indexes
interset_edges_length]=getMinDistanceBetweenPointSets(set_a,set_b)
%helper function. get the smallest distance between two non-overlapping sets of points

points_cloud=[set_a;set_b];
set_a_sz=size(set_a,1);
delaunay_tri=delaunay(points_cloud(:,1),points_cloud(:,2));
delaunay_edge_indexes=[delaunay_tri(:,1:2);delaunay_tri(:,2:3); [delaunay_tri(:,3)
delaunay_tri(:,1)]];
%keep only the edges between the two sets and get rid of those within the two sets
delaunay_edge_indexes((delaunay_edge_indexes(:,1)<=set_a_sz)&(delaunay_edge_indexes(:,2)<=set_a_s
z),:)=[];
delaunay_edge_indexes((delaunay_edge_indexes(:,1)>set_a_sz)&(delaunay_edge_indexes(:,2)>set_a_sz)
,:)=[];
%calculate the inter-set distances
first_edge_points=points_cloud(delaunay_edge_indexes(:,1),:);
second_edge_points=points_cloud(delaunay_edge_indexes(:,2),:);
interset_edges_length=hypot(first_edge_points(:,1)-
second_edge_points(:,1),first_edge_points(:,2)-second_edge_points(:,2));
interset_edge_indexes=delaunay_edge_indexes;
[min_distance min_distance_index]=min(interset_edges_length);

%end getMinDistanceBetweenPointSets
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getModuleDescription.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function module_description=getModuleDescription(module_path)
%helper function for assayEditorGUI. get the description of a module from the module file
module_description='';
fi=fopen(module_path);
module_name=module_path(1:(end-2));
tl=fgetl(fi);
while isempty(strfind(tl,module_name))
    tl=fgetl(fi);
end
tl=fgetl(fi);
while (length(tl)>0)&&(tl(1)=='%')
    module_description=[module_description ' ' tl(2:end)];
    tl=fgetl(fi);
end
fclose(fi);

%end getModuleDescription
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getModuleList.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function modules_list=getModuleList()
%helper function for assayEditorGUI. build a list of modules in the current directory
%get alll the .m files

m_files=dir('*.m');
modules_list={};
%check which ones are modules
%modules are recognized by the presence of an input_args and an output_args
%structure
for i=1:length(m_files)
    file_text=fileread(m_files(i).name);
    if strcmp(m_files(i).name,'getModuleList.m')
        continue;
    end
    if strcmp(m_files(i).name,'aeMenuOpenAssay.m')
        continue;
    end
    if strcmp(m_files(i).name,'wrapFunction.m')
        continue;
    end
    %remove comments since they may contain the strings we're searching for
    module_text= regexprep(file_text, '%[^%\n]*\n', '');
    if isempty(strfind(module_text,'input_args'))
        continue;
    end
    if isempty(strfind(module_text,'output_args'))
        continue;
    end
    modules_list=[modules_list;m_files(i).name];
```

```
end

%sort the module list alphabetically
[dummy sort_idx]=sort(lower(modules_list));
modules_list=modules_list(sort_idx);

%end getModuleList
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getModuleStruct.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function [module_struct is_chain chain_name]=getModuleStruct(list_text,selection_idx,handles)
%helper function for assayEditorGUI. get the module struct corresponding to the current selection
selection_text=list_text{selection_idx};
sublevel_depth=getSelectionLevel(selection_text);
is_control=false;
if strcmp(selection_text(1:9),'<html><i>')
    %get the module text for the current chain
    chain_var=regexp(selection_text,'<html><i>(?: )*(\w*)<','tokens','once');
    is_chain=true;
    i=1;
    while 1
        selection_text=list_text{selection_idx-i};
        if
(strcmp(selection_text(1:9),'<html><b>')&&(getSelectionLevel(selection_text)==sublevel_depth))
            break;
        end
        i=i+1;
    end
elseif strcmp(selection_text(1:9),'<html><b>')
    is_chain=true;
    is_control=true;
else
    is_chain=false;
end
module_id=stripHTMLFromString(selection_text);
modules_list=handles.ModulesList;
modules_map=handles.ModulesMap;
module_idx=modules_map.get(module_id);
module_struct=modules_list{module_idx};
if (~is_chain)
    chain_name=module_struct.ChainName;
else
    if (is_control)
        chain_name=module_struct.Chains{1};
    else
        chain_idx=strcmp(chain_var,module_struct.ChainVars);
        chain_name=module_struct.Chains{chain_idx};
    end
end
%end getModuleStruct
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getNearbyTracksSorted.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function [nearby_tracks_sorted group_idx
matching_groups]=getNearbyTracksSorted(cur_id,cells_centroids,shape_params,tracks_layout...

,cur_tracks,prev_tracks,search_radius_pct,matching_groups,params_coeff_var,relevant_params_idx,ma
tching_group_stats,...

params_for_sure_match,param_weights,unknown_param_weights,distance_ranking_order,direction_rankin
g_order,unknown_ranking_order,...

min_second_distance,max_dist_ratio,max_angle_diff,max_search_dist,min_search_dist,front_params)
%helper function for CA tracking algorithm. get the tracks in the local nhood of this cell sorted
by matching scores
hugeNbr=1e6;
cur_cell_centroid=cells_centroids(cur_id,:);
areaCol=tracks_layout.AreaCol;
solCol=tracks_layout.SolCol;
centroid1Col=tracks_layout.Centroid1Col;
centroid2Col=tracks_layout.Centroid2Col;
trackIDCol=tracks_layout.TrackIDCol;
min_reliable_params=params_for_sure_match;
group_idx=0;
```

```
cur_shape_params=shape_params(cur_id,:);
dist_to_tracks=hypot(cur_tracks(:,centroid1Col)-cur_cell_centroid(1),...
    cur_tracks(:,centroid2Col)-cur_cell_centroid(2));
dist_to_tracks_sorted=sort(dist_to_tracks);
nearest_distance=dist_to_tracks_sorted(1);
search_radius=search_radius_pct*nearest_distance;
if (search_radius>max_search_dist)
    search_radius=max_search_dist;
end
if (search_radius<min_search_dist)
    search_radius=min_search_dist;
end
nearby_tracks_idx=(dist_to_tracks<search_radius);
%keep only tracks in the current search nhood
dist_to_tracks=dist_to_tracks(nearby_tracks_idx);
nearby_tracks=cur_tracks(nearby_tracks_idx,:);
if (isempty(nearby_tracks))
    nearby_tracks_sorted=[];
else
    %rank the tracks by how close their features are to the features of the
    %current cell
    %try to get previous cell travel direction if possible
    if (~isempty(prev_tracks))
        prev_nearby_tracks_idx=ismember(prev_tracks(:,trackIDCol),nearby_tracks(:,trackIDCol));
        prev_nearby_tracks=prev_tracks(prev_nearby_tracks_idx,:);
        if isempty(prev_nearby_tracks)
            prev_tracks_centroids=[];
        else
            nr_cur_tracks=size(nearby_tracks,1);
            prev_tracks_centroids=zeros(nr_cur_tracks,2);

preexisting_tracks_idx=ismember(nearby_tracks(:,trackIDCol),prev_nearby_tracks(:,trackIDCol));
            nr_preexisting_tracks=sum(preexisting_tracks_idx);
            preexisting_tracks=nearby_tracks(preexisting_tracks_idx,:);
            cur_nearby_tracks_centroids=preexisting_tracks(:,centroid1Col:centroid2Col);
            if (nr_preexisting_tracks==nr_cur_tracks)
                %easy situation - all tracks are more than a frame old so
                %we have directional information for all of them
                %match order of tracks
                [dummy sort_cur_tracks_idx]=sort(preexisting_tracks(:,trackIDCol));
                [dummy sort_prev_tracks_idx]=sort(prev_nearby_tracks(:,trackIDCol));
                %key to match the previous tracks with the current tracks
                key(sort_cur_tracks_idx)=[1:length(sort_cur_tracks_idx)];
                match_tracks_idx=sort_prev_tracks_idx(key);
                prev_nearby_tracks=prev_nearby_tracks(match_tracks_idx,:);
                prev_nearby_tracks_centroids=prev_nearby_tracks(:,centroid1Col:centroid2Col);
                prev_tracks_centroids(preexisting_tracks_idx,:)=prev_nearby_tracks_centroids;
            else
                prev_nearby_tracks_centroids=zeros(nr_preexisting_tracks,2);
                for i=1:nr_preexisting_tracks
                    track_id=preexisting_tracks(i,trackIDCol);
                    track_idx=prev_nearby_tracks(:,trackIDCol)==track_id;

prev_nearby_tracks_centroids(i,:)=prev_nearby_tracks(track_idx,centroid1Col:centroid2Col);
                end
            end

        end
    else
        prev_tracks_centroids=[];
    end

    if isempty(prev_tracks_centroids)
        b_use_direction=false;
        %i'm assuming areaCol is the first param column and solCol the last
        cell_ranking_params=[min(dist_to_tracks) cur_shape_params(:,1:(solCol-areaCol+1))];
        tracks_ranking_params=[dist_to_tracks nearby_tracks(:,areaCol:solCol)];
        %sort the tracks by ranking tracks in pairs to the cell instead of
        %all tracks at once. this prevents false best matching tracks.
        tracks_ranks=rankParams(cell_ranking_params,tracks_ranking_params);
        ranking_order=getRankingOrder(cell_ranking_params,tracks_ranking_params,tracks_ranks,...

matching_groups,tracks_layout,b_use_direction,matching_group_stats,min_second_distance,max_dist_r
atio,max_angle_diff,...
            unknown_ranking_order,distance_ranking_order,direction_ranking_order);
        group_idx=0;
        [tracks_params_sorted
sort_idx]=sortManyToOneUsingPairs(cell_ranking_params,tracks_ranking_params,...

b_use_direction,unknown_param_weights,param_weights,ranking_order,group_idx,relevant_params_idx);
        %rank the best and second best matching track
```

```
    else
        b_use_direction=true;
        prev_tracks_directions=atan2((cur_nearby_tracks_centroids(:,2)-
prev_nearby_tracks_centroids(:,2)),...
            (cur_nearby_tracks_centroids(:,1)-prev_nearby_tracks_centroids(:,1)));
        cur_possible_track_directions=atan2((cur_cell_centroid(2)-
cur_nearby_tracks_centroids(:,2)),...
            (cur_cell_centroid(1)-cur_nearby_tracks_centroids(:,1)));
        directions_diff=zeros(nr_cur_tracks,1);
        directions_diff(preexisting_tracks_idx)=abs(cur_possible_track_directions-
prev_tracks_directions);
        directions_diff(~preexisting_tracks_idx)=hugeNbr;
        %i'm assuming areaCol is the first param column and solCol the last
        cell_ranking_params=[min(dist_to_tracks) 0 cur_shape_params(:,1:(solCol-areaCol+1))];
        tracks_ranking_params=[dist_to_tracks directions_diff nearby_tracks(:,areaCol:solCol)];
        tracks_ranks=rankParams(cell_ranking_params,tracks_ranking_params);
        [ranking_order
group_idx]=getRankingOrder(cell_ranking_params,tracks_ranking_params,tracks_ranks...

,matching_groups,tracks_layout,b_use_direction,matching_group_stats,min_second_distance,max_dist_
ratio,max_angle_diff,...
            unknown_ranking_order,distance_ranking_order,direction_ranking_order);
        %sort the tracks by ranking tracks in pairs to the cell instead of
        %all tracks at once. this prevents false best matching tracks.
        [tracks_params_sorted
sort_idx]=sortManyToOneUsingPairs(cell_ranking_params,tracks_ranking_params,...

b_use_direction,unknown_param_weights,param_weights,ranking_order,group_idx,relevant_params_idx);
        %figure out if we have a track that is a sure match to the cell
        if (length(sort_idx)==1)
            b_sure_match=true;
            pair_ranks=rankParams(cell_ranking_params,tracks_params_sorted);
        else
            %rank the best and second best matching track
            pair_ranks=rankParams(cell_ranking_params,tracks_params_sorted(1:2,:));
            nr_best_match_params=sum(pair_ranks(1,:)==1);
            if (nr_best_match_params>=params_for_sure_match)
                b_sure_match=true;
            else
                b_sure_match=false;
            end
        end
        if (b_sure_match)
            %this track is a sure match-we'll use it to figure out which
            %parameters work best to assign other cells
            if (length(sort_idx)==1)
                [dummy matching_groups
group_idx]=addToMatchingGroups(matching_groups,cell_ranking_params,...
                    tracks_params_sorted,params_coeff_var,1,min_reliable_params,pair_ranks,...

relevant_params_idx,max_angle_diff,min_second_distance,max_dist_ratio,front_params);
            else
                [dummy matching_groups
group_idx]=addToMatchingGroups(matching_groups,cell_ranking_params,...

tracks_params_sorted(1:2,:),params_coeff_var,1,min_reliable_params,pair_ranks,...

relevant_params_idx,max_angle_diff,min_second_distance,max_dist_ratio,front_params);
            end
        else
            if (length(sort_idx)==1)
                [dummy
group_idx]=getRankingOrder(cell_ranking_params,tracks_params_sorted,pair_ranks,...

matching_groups,tracks_layout,true,matching_group_stats,min_second_distance,max_dist_ratio,max_an
gle_diff,...
                    unknown_ranking_order,distance_ranking_order,direction_ranking_order);
            else
                    [dummy
group_idx]=getRankingOrder(cell_ranking_params,tracks_params_sorted(1:2,:),pair_ranks,...

matching_groups,tracks_layout,true,matching_group_stats,min_second_distance,max_dist_ratio,max_an
gle_diff,...
                    unknown_ranking_order,distance_ranking_order,direction_ranking_order);
            end
        end
    end
    nearby_tracks_sorted=nearby_tracks(sort_idx,:);
end
```

```
%end getNearbyTracksSorted
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getObjectIntensities.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=getObjectIntensities(input_args)
% Usage
% This module is used to return the object intensities from objects in a label matrix using the
corresponding intensity image.
% Input Structure Members
% LabelMatrix - The label matrix containing the objects for which the mean intensity data will be
extracted.
% IntensityImage - Matrix containing the intensity image.
% Output Structure Members
% MeanIntensities - Array containing the mean intensities for each object in the label matrix.

objects_lbl=input_args.LabelMatrix.Value;
objects_idx=objects_lbl>0;
intensity_img=input_args.IntensityImage.Value;
objects_intensities=accumarray(objects_lbl(objects_idx),intensity_img(objects_idx));
object_areas=accumarray(objects_lbl(objects_idx),1);
output_args.MeanIntensities=objects_intensities./object_areas;

%end getObjectIntensities
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getObjectsMeanDisplacement.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=getObjectsMeanDisplacement(input_args)
% Usage
% This module estimates the average displacement of the cells in the frame using the centroids
from the previous and current frame.
% Input Structure Members
% Centroid1Col - The index of the first coordinate of the object centroids in the tracks matrix.
% Centroid2Col - The index of the second coordinate of the object centroids in the tracks matrix.
% CurrentTracks - Set of tracks belonging to previous frame.
% ObjectCentroids - The list of centroids in the current frame.
% Output Structure Members
% MeanDisplacement - The estimated mean displacement of the objects in the frame.
% SDDisplacement - The estimated standard deviation of the objects in the frame.

prev_frame_centroids=input_args.CurrentTracks.Value(:,input_args.Centroid1Col.Value:input_args.Ce
ntroid2Col.Value);
cur_frame_centroids=input_args.ObjectCentroids.Value;
delaunay_tri=delaunay(prev_frame_centroids(:,1),prev_frame_centroids(:,2));
nearest_neighbors_idx=dsearch(prev_frame_centroids(:,1),prev_frame_centroids(:,2),delaunay_tri,..
.
    cur_frame_centroids(:,1),cur_frame_centroids(:,2));
cell_displacements=hypot(cur_frame_centroids(:,1)-
prev_frame_centroids(nearest_neighbors_idx,1),...
    cur_frame_centroids(:,2)-prev_frame_centroids(nearest_neighbors_idx,2));
output_args.MeanDisplacement=mean(cell_displacements);
output_args.SDDisplacement=std(cell_displacements);
output_args.SearchRadius=output_args.MeanDisplacement+10*output_args.SDDisplacement;

%end getObjectsMeanDisplacement
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getPairScoresToSingle.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function
pair_scores=getPairScoresToSingle(pair_params,single_params,b_use_direction,unknown_param_weights
,...
    param_weights,pair_ranking_order,pair_group_idx,relevant_params_idx)
%helper function for CA tracking algorithm. see which one of two cells is a better match for a
track or which of two tracks is a better match for a cell. this has to
%be done one cell pair at a time otherwise the best cell/track may not be picked
assert(size(pair_params,1)==2);
%keep only the relevant parameters
irrelevant_cols=find(relevant_params_idx==0);
irrelevant_cols_idx=ismember(pair_ranking_order,irrelevant_cols);
pair_ranking_order(irrelevant_cols_idx)=[];
for i=1:length(pair_ranking_order)
```

```
    %need to adjust the index of the columns that were above
    %columns that were removed
    pair_ranking_order(i)=pair_ranking_order(i)-sum(irrelevant_cols<pair_ranking_order(i));
end
if(~b_use_direction)
    relevant_params_idx(2)=[];
    if (pair_group_idx==0)
        unknown_param_weights=unknown_param_weights(1:end-1);
    else
       param_weights=param_weights(1:end-1);
    end
end
pair_params=pair_params(:,relevant_params_idx);
single_params=single_params(:,relevant_params_idx);
if (pair_group_idx==0)
        unknown_param_weights=unknown_param_weights(relevant_params_idx);
    else
       param_weights=param_weights(relevant_params_idx);
end
pair_ranks=rankParams(single_params,pair_params);
if (pair_group_idx==0)
    if (b_use_direction)
        [dummy dummy2
pair_scores]=sortTracks(pair_ranks,pair_params,pair_ranking_order,unknown_param_weights);
    else
        [dummy dummy2 pair_scores]=sortTracks(pair_ranks,pair_params,...

pair_ranking_order(pair_ranking_order<max(pair_ranking_order)),unknown_param_weights);
    end
else
    if (b_use_direction)
        [dummy dummy2
pair_scores]=sortTracks(pair_ranks,pair_params,pair_ranking_order,param_weights);
    else
        [dummy dummy2 pair_scores]=sortTracks(pair_ranks,pair_params,...
            pair_ranking_order(pair_ranking_order<max(pair_ranking_order)),param_weights);
    end
end

%end getPairScoresToSingle
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getParamsCoefficientOfVariation.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=getParamsCoefficientOfVariation(input_args)
%module to get the coefficient of variation for tracking params
nr_params=input_args.SolidityCol.Value-input_args.AreaCol.Value+1;
shape_params=input_args.Params.Value;
params_means=mean(shape_params(:,1:nr_params));
params_sds=std(shape_params(:,1:nr_params));
output_args.CoefficientOfVariation=params_sds./params_means; %coefficient of variation of the
params;

%end getParamsCoefficientOfVariation
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getRankingOrder.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function [ranking_order
group_idx]=getRankingOrder(cur_shape_params,nearby_shape_params,nearby_ranks,...

matching_groups,tracks_layout,bUseDirection,matching_group_stats,min_second_distance,max_dist_rat
io,max_angle_diff,unknown_ranking_order,...
    distance_ranking_order,direction_ranking_order)
%helper function for CA tracking algorithm. we need to figure out which parameters to use first
when trying to match
%this cell to a track - this can be done if the cell can be assigned to a
%matching group. if not we'll assign a default ranking_order. determine if
%we need a distance-biased or direction-biased ranking order
distanceCol=1;
angleCol=2;
start_params_col=tracks_layout.AreaCol;
end_params_col=tracks_layout.SolCol;
group_id_col=tracks_layout.MatchGroupIDCol;
[dummy closest_distance_idx]=min(nearby_ranks(:,distanceCol));
[dummy closest_angle_idx]=min(nearby_ranks(:,angleCol));
```

```
bDirection=false;
bDistance=false;
if (isempty(matching_group_stats))
    b_group_stats=false;
else
    b_group_stats=true;
    nr_groups=size(matching_group_stats,1);
    %can't use the matching groups added in this frame as we don't have the
    %stats for those yet
    matching_groups=matching_groups(1:nr_groups,:);
end

if (closest_angle_idx~=closest_distance_idx)
    %nearest distance and angle point to different cells
    %use the other parameters to pick which one is right
    angle_score=sum(nearby_ranks(closest_angle_idx,:)==1);
    dist_score=sum(nearby_ranks(closest_distance_idx,:)==1);
    if (abs(angle_score-dist_score)>2)
        %we have a clear favorite
        if (angle_score<dist_score)
            if (bUseDirection)
                %use direction matching groups
                if (~isempty(matching_groups))
                    direction_groups_idx=matching_groups(:,1)==2;
                    matching_groups=matching_groups(direction_groups_idx,:);
                    if (b_group_stats)
                        matching_group_stats=matching_group_stats(direction_groups_idx,:);
                    end
                end
                bDirection=true;
            end
        else
            %use distance matching groups
            if (~isempty(matching_groups))
                distance_groups_idx=matching_groups(:,1)==1;
                matching_groups=matching_groups(distance_groups_idx,:);
                if (b_group_stats)
                    matching_group_stats=matching_group_stats(distance_groups_idx,:);
                end
            end
            bDistance=true;
        end
    else
        %no clear favorite - use more details
        %if one of the cells is a lot closer than the others use distance
        distances_sorted=sort(nearby_shape_params(:,distanceCol));
        dist_ratio=distances_sorted(1)/distances_sorted(2);
        if ((distances_sorted(2)>min_second_distance)&&(dist_ratio<max_dist_ratio))
            %use distance
            %use distance matching groups
            if (~isempty(matching_groups))
                distance_groups_idx=matching_groups(:,1)==1;
                matching_groups=matching_groups(distance_groups_idx,:);
                if (b_group_stats)
                    matching_group_stats=matching_group_stats(distance_groups_idx,:);
                end
            end
            bDistance=true;
        end
        %if one angle is within 20 degress of the previous direction and
        %all other angles are further than 20 degrees from our angle use
        %direction
        angle_diffs_sorted=sort(nearby_shape_params(:,2));
        if ((angle_diffs_sorted(1)<max_angle_diff)&&(abs(angle_diffs_sorted(1)-
angle_diffs_sorted(2))>max_angle_diff)&&bUseDirection)
            %use direction matching groups
            if (~isempty(matching_groups))
                direction_groups_idx=matching_groups(:,1)==2;
                matching_groups=matching_groups(direction_groups_idx,:);
                if (b_group_stats)
                    matching_group_stats=matching_group_stats(direction_groups_idx,:);
                end
            end
            bDirection=true;
        end
    end
end

if (isempty(matching_group_stats)||(size(cur_shape_params,2)<(end_params_col-
start_params_col+3)))
    if (bDistance)
```

```
        ranking_order=distance_ranking_order;
    elseif (bDirection)
        if (bDistance)
            ranking_order=unknown_ranking_order;
        else
            ranking_order=direction_ranking_order;
        end
    else
        ranking_order=unknown_ranking_order;
    end
    group_idx=0;
    return;
end

%can compare only on the true shape params first two columns are distance
%and direction
cur_params=cur_shape_params(:,3:end_params_col-start_params_col+3);
nr_groups=size(matching_group_stats,1);
group_diff=abs(matching_group_stats-repmat(cur_params,nr_groups,1));
[dummy sort_idx]=sort(group_diff);
ranks_sum=sum(sort_idx,2);
[dummy group_idx]=min(ranks_sum);
ranking_order=matching_groups(group_idx,:);

%end getRankingOrder
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getRegionProps.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=getRegionProps(input_args)
%module to return the region properties of the objects in a
%label matrix. a particular wrapper to regionprops.
%Input Structure Members
%LabelMatrix - Label matrix from which the object properties will be
%extracted
%Output Structure Members
%RegionProps - Matrix containing the object properties extracted from the
%label matrix.
cells_lbl=input_args.LabelMatrix.Value;
cells_props=regionprops(cells_lbl,'Centroid','Area','Eccentricity','MajorAxisLength','MinorAxisLe
ngth',...
    'Orientation','Perimeter','Solidity');
cells_centroids=[cells_props.Centroid]';
centr_len=size(cells_centroids,1);
cells_centroids=[cells_centroids(2:2:centr_len) cells_centroids(1:2:centr_len)];
%i want the orientation from 0 to 180 instead of -90 to 90
region_props=[(1:centr_len/2)' cells_centroids [cells_props.Area]' [cells_props.Eccentricity]'
[cells_props.MajorAxisLength]' ...
    [cells_props.MinorAxisLength]' [cells_props.Orientation]'+90 [cells_props.Perimeter]'...
    [cells_props.Solidity]'];
output_args.RegionProps=region_props;

%end getRegionProps
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getScriptVariables.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function script_variables=getScriptVariables(selected_assay)
%helper function for assayEditorGUI. read any script variables from the assay text

file_text=fileread(selected_assay);
%extract the script variables section
var_text=regexp(file_text,'%script variables(.*)%end script variables','tokens');
script_variables=regexp(var_text{1},'(?:\s*)([^=]*)=([^;]*);','tokens');
script_variables=script_variables{1};

%end getScriptVariables
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% getSelectionLevel.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function selection_level=getSelectionLevel(selection_text)
%helper function for assayEditorGUI. get the level of the chain this module is attached to in the
current assay
```

```
%get the number of whitespaces
search_pattern=['[^&]*( )*\w*<'];
ws=regexp(selection_text,search_pattern,'tokens');
selection_level=length(ws{1}{1})/12;
if ~strcmp(selection_text(1:9),'<html><i>')
    selection_level=selection_level+1;
end

%end getSelectionLevel
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%% getSelectionNames.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function selection_names=getSelectionNames()
%helper function for manual tracking review. get a list of selection layer
%names
global mtr_gui_struct;

selection_layers=mtr_gui_struct.SelectionLayers;
selection_names={};
for i=1:length(selection_layers)
    selection_names{i}=selection_layers{i}.Name;
end

%end getSelectionNames
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%% getShapeParams.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=getShapeParams(input_args)
% Usage
% This module is used to extract the shape parameters (Area, Eccentricity, etc.) of objects in a
label matrix.  Mostly, a wrapper for the MATLAB regionprops function. Adds a blob id to the
output so that objects that belong to the same blob may be identified at a later time.
% Input Structure Members
% LabelMatrix - The label matrix from which the shape parameters will be extracted.
% Output Structure Members
% Centroids - The centroids of the objects in the label matrix.
% ShapeParameters - The shape parameters of the objects in the label matrix.

cells_lbl=input_args.LabelMatrix.Value;
cells_props=regionprops(cells_lbl,'Centroid','Area','Eccentricity','MajorAxisLength','MinorAxisLe
ngth',...
    'Orientation','Perimeter','Solidity');
cells_centroids=[cells_props.Centroid]';
centr_len=size(cells_centroids,1);
cells_centroids=[cells_centroids(2:2:centr_len) cells_centroids(1:2:centr_len)];
%also get the blob id of each cell-this helps merge oversegmented
%cells later
cells_bw=cells_lbl>0;
blob_lbl=bwlabeln(cells_bw);
nr_cells=size(cells_centroids,1);
blob_ids=zeros(nr_cells,1);
matching_group_ids=blob_ids;
for i=1:nr_cells
    blob_ids(i)=getLabelId(blob_lbl,cells_centroids(i,:));
end
%i want the orientation from 0 to 180 instead of -90 to 90 so i can run
%percentages
shape_params=[[cells_props.Area]' [cells_props.Eccentricity]' [cells_props.MajorAxisLength]' ...
    [cells_props.MinorAxisLength]' [cells_props.Orientation]'+90 [cells_props.Perimeter]'...
    [cells_props.Solidity]' blob_ids matching_group_ids];
output_args.ShapeParameters=shape_params;
output_args.Centroids=cells_centroids;

%end getShapeParams
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%% getShapeParamsWithDisconnects.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=getShapeParamsWithDisconnects(input_args)
% Usage
% This module is used to extract the shape parameters (Area, Eccentricity, etc.) of objects in a
label matrix.  The difference between this module and getShapeParams is that
```

```
getShapeParamsWithDisconnects supports objects which are disjointed (spread across multiple
blobs) at the cost of execution speed.
% Input Structure Members
% LabelMatrix - The label matrix from which the shape parameters will be extracted.
% Output Structure Members
% Centroids - The centroids of the objects in the label matrix.
% ShapeParameters - The shape parameters of the objects in the label matrix.


cells_lbl=input_args.LabelMatrix.Value;
cells_props=regionprops(cells_lbl,'Centroid','Area','Eccentricity','MajorAxisLength','MinorAxisLe
ngth',...
    'Orientation','Perimeter','Solidity');
cells_centroids=[cells_props.Centroid]';
centr_len=size(cells_centroids,1);
cells_centroids=[cells_centroids(2:2:centr_len) cells_centroids(1:2:centr_len)];
%also get the blob id of each cell-this helps merge oversegmented
%cells later
cells_bw=cells_lbl>0;
blob_lbl=bwlabeln(cells_bw);
nr_cells=size(cells_centroids,1);
blob_ids=zeros(nr_cells,1);
matching_group_ids=blob_ids;
%i want the orientation from 0 to 180 instead of -90 to 90 so i can run
%percentages
shape_params=[[cells_props.Area]' [cells_props.Eccentricity]' [cells_props.MajorAxisLength]' ...
    [cells_props.MinorAxisLength]' [cells_props.Orientation]'+90 [cells_props.Perimeter]'...
    [cells_props.Solidity]' blob_ids matching_group_ids];

for i=1:nr_cells
    cur_blob=(cells_lbl==i);
    cur_blob_lbl=bwlabeln(cur_blob);
    if max(cur_blob_lbl(:)>1)
        %a blob that consists of more than one piece
        cur_blob_ids=unique(blob_lbl(cur_blob));
        blob_ids(i)=min(cur_blob_ids);
        cur_blob=joinFragmentedBlob(cur_blob);
        cur_blob_lbl=bwlabeln(cur_blob);
        cur_blob_props=regionprops(cur_blob_lbl,'Centroid','Area','Eccentricity',...
            'MajorAxisLength','MinorAxisLength','Orientation',...
            'Perimeter','Solidity');
        shape_params(i,1:(end-2))=[cur_blob_props.Area cur_blob_props.Eccentricity
cur_blob_props.MajorAxisLength...
            cur_blob_props.MinorAxisLength (cur_blob_props.Orientation+90)
cur_blob_props.Perimeter...
            cur_blob_props.Solidity];
        cells_centroids(i,:)=[cur_blob_props.Centroid(2) cur_blob_props.Centroid(1)];
    else
        blob_ids(i)=getLabelId(blob_lbl,cells_centroids(i,:));
    end
end

shape_params(:,(end-1))=blob_ids;
output_args.ShapeParameters=shape_params;
output_args.Centroids=cells_centroids;

%end getShapeParams
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% getTrackIDs.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=getTrackIDs(input_args)
% Usage
% This module is used to retrieve the list of track IDs from the track matrix.
% Input Structure Members
% TrackIDCol - Index of the track ID column in the tracks matrix.
% Tracks - Matrix containing the set of tracks from which IDs will be extracted.
% Output Structure Members
% TrackIDs - The IDs extracted from the tracks matrix.

output_args.TrackIDs=unique(input_args.Tracks.Value(:,input_args.TrackIDCol.Value));

%end getTrackIDs
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% ginput_Wrapper.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=ginput_Wrapper(input_args)
%basic wrapper for the matlab ginput function
%Input Structure Members
%None
%Output Structure Members
%XYCoords - Array containing the coordinates returning by ginput.

output_args.XYCoords=ginput;

%end imadjust_Wrapper
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% holdValue.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=holdValue(input_args)
% Usage
% The only purpose for this module is to hold a value so that other modules may use it, for
example by having the holdValue module at a higher level in the hierarchy, thereby preventing
modules at lower levels from overwriting the value.
% Input Structure Members
% ValueToHold - The value to hold.
% Output Structure Members
% ValueToHold - The held value.

output_args.ValueToHold=input_args.ValueToHold.Value;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% if_statement.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function [output_args]=if_statement(function_struct)
% Usage
% This module is used to create branching execution in an assay. Depending on the result of a
test function, either the modules in the IfFunctions or the modules in the ElseFunctions
structures will be executed.
% Input Structure Members
% TestVariable - The module which will be used to determine what subset of modules will be
executed. This module needs to return a true/false value.
% IfFunctions - Set of modules which will be executed if the value returned by the test function
is true.
% ElseFunctions - Set of modules which will be executed if the value returned by the test
function is false.

updateArgs(function_struct.InstanceName,function_struct.FunctionArgs,'input');
input_args=function_struct.FunctionArgs;
if(input_args.TestVariable.Value)
    if_functions=function_struct.IfFunctions;
else
    if_functions=function_struct.ElseFunctions;
end

for i=1:size(if_functions,1)
    if_function_instance_name=if_functions{i}.InstanceName;
    callFunction(if_function_instance_name,false);
end

output_args=makeOutputStruct(function_struct);

%end if_statement
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% im2bw_Wrapper.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=im2bw_Wrapper(input_args)
%a very simple wrapper module for the Matlab im2bw function
%Input Structure Members
%Image - The grayscale image to be converted.
%Output Structure Members
%Image - The resulting binary image.

output_args.Image=im2bw(input_args.Image.Value);
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% imadjust_Wrapper.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=imadjust_Wrapper(input_args)
%basic wrapper for the matlab imadjust function
%Input Structure Members
%Image - The grayscale image to be processed.
%Output Structure Members
%Image - The filtered image.

output_args.Image=imadjust(input_args.Image.Value);

%end imadjust_Wrapper
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% imageErode.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=imageErode(input_args)
%simple wrapper for the Matlab imerode(im,se) function
%Input Structure Members
%Image - The grayscale image to be processed.
%Output Structure Members
%Image - The filtered image.
img=input_args.Image.Value;
se=input_args.StructuralElement.Value;
output_args.Image=imerode(img,se);

%end imageErode
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% imageFromRegion.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=imageFromRegion(input_args)
%create a binary image setting the region as foreground
%Input Structure Members
%ImageSize - The size of the image to be created.
%RegionPixels - Array containing the list of pixels in region.
%Output Structure Members
%Image - The mask image.
img_sz=input_args.ImageSize.Value;
fgnd=round(input_args.RegionPixels.Value);
mask=poly2mask(fgnd(:,1),fgnd(:,2),img_sz(1),img_sz(2));
output_args.Image=repmat(mask,[1 1 img_sz(3)]);

%end imageFromRegion
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% imclearborderSlices.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=imclearborderSlices(input_args)
%used to clear the border in a z-stack image. imclearborder can handle 3-D
%images but it clears the top and bottom slices completely.
%Input Structure Members
%Image - The binary image to be processed.
%Output Structure Members
%Image - The resulting binary image.
img=input_args.Image.Value;
img_sz=size(img);
img_output=zeros(img_sz);

for i=1:img_sz(3)
    img_output(:,:,i)=imclearborder(img(:,:,i));
end

output_args.Image=img_output;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% imclearborder_Wrapper.m %%%%%%%%%%%%%%%%%%%%%%%%
function output_args=imclearborder_Wrapper(input_args)
```

```
%a very simple wrapper module for the Matlab imclearborder function
%Input Structure Members
%Image - The binary image to be processed.
%Output Structure Members
%Image - The resulting binary image.

output_args.Image=imclearborder(input_args.Image.Value);

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% imcomplementWrapper.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=imcomplementWrapper(input_args)
%simple module to wrap the Matlab imcomplement function
%Input Structure Members
%Image - The image to be processed.
%Output Structure Members
%Image - The resulting image.

output_args.Image=imcomplement(input_args.Image.Value);

%end saveCellsLabel
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% imNorm.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=imNorm(input_args)
% Usage
% This module is used to normalize an image so that the lowest pixel value is zero and the
highest pixel value is the maximum allowed value for the specified integer class.
% Input Structure Members
% IntegerClass - The integer class of the image.  Has to be an integer class supported by MATLAB
such as 'int8' or 'uint8'.
% RawImage - The image to be processed.
% Output Structure Members
% Image - The normalized image.

int_class=input_args.IntegerClass.Value;
max_val=double(intmax(int_class));
img_raw=input_args.RawImage.Value;
img_dbl=floor(double((img_raw-min(img_raw(:))))*max_val./double(max(img_raw(:))-
min(img_raw(:))));
switch(int_class)
    case 'uint8'
       output_args.Image=uint8(img_dbl);
    case 'uint16'
       output_args.Image=uint16(img_dbl);
    otherwise
        output_args.Image=[];
end

%end function
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% importScriptVariables.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function importScriptVariables(handles)
%helper function for assayEditorGUI. import script variables from a text file

[file_name,path_name] = uigetfile('*.m','Select file containing script variables:');
if ~file_name
    return;
end
handles.ScriptVariables=getScriptVariables([path_name '/' file_name]);
guidata(handles.figure1,handles);

%end importScriptVariables
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% imregionalmax_Wrapper.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=imregionalmax_Wrapper(input_args)
```

```
%wrapper module for the Matlab imregionalmax function
%Input Structure Members
%Image - The grayscale image for which the regional maxima should be calculated.
%Output Structure Members
%Image - The resulting binary image.

output_args.Image=imregionalmax(input_args.Image.Value);

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% imwrite_Wrapper.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=imwrite_Wrapper(input_args)
%basic wrapper for the MATLAB imwrite function
%Input Structure Members
%Image - The image to be saved.
%FileName - The path where the image will be saved.
%Format - The image format.
%Output Structure Members
%None

img=input_args.Image.Value;
file_name=input_args.FileName.Value;
fmt=input_args.Format.Value;
imwrite(img,file_name,fmt,'Compression','none');

output_args=[];

%end saveImage
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% inputArgumentsGUI.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function varargout = inputArgumentsGUI(varargin)
% INPUTARGUMENTSGUI M-file for inputArgumentsGUI.fig
%      INPUTARGUMENTSGUI, by itself, creates a new INPUTARGUMENTSGUI or raises the existing
%      singleton*.
%
%      H = INPUTARGUMENTSGUI returns the handle to a new INPUTARGUMENTSGUI or the handle to
%      the existing singleton*.
%
%      INPUTARGUMENTSGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in INPUTARGUMENTSGUI.M with the given input arguments.
%
%      INPUTARGUMENTSGUI('Property','Value',...) creates a new INPUTARGUMENTSGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before inputArgumentsGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to inputArgumentsGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help inputArgumentsGUI

% Last Modified by GUIDE v2.5 21-Sep-2011 21:01:26

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @inputArgumentsGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @inputArgumentsGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before inputArgumentsGUI is made visible.
function inputArgumentsGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to inputArgumentsGUI (see VARARGIN)

% Choose default command line output for inputArgumentsGUI

handles.output = hObject;
handles.OK=false;
mt_idx=find(strcmp(varargin, 'ModulesList'))+1;
handles.ModulesList=varargin{mt_idx};
mt_idx=find(strcmp(varargin, 'ModulesMap'))+1;
handles.ModulesMap=varargin{mt_idx};
ms_idx=find(strcmp(varargin, 'ModuleStruct'))+1;
handles.ModuleStruct=varargin{ms_idx};
handles.SelectionType='';
handles.SelectionValue='';
handles.Erased=false;
%make sure all the args that are filled refer to modules that still exist
handles.ModuleStruct=validateModuleArgs(handles);
module_struct=handles.ModuleStruct;
%update the window title
win_title=['Edit Module Parameters - ' module_struct.InstanceName ' - '
module_struct.ModuleName];
set(handles.figure1,'Name',win_title);
%show the module description
set(handles.editStatus,'String',getModuleDescription([module_struct.ModuleName '.m']));
%populate the dialog boxes
populateInputStringsListbox(handles);
populateModuleInstancesPopup(handles);
updateOutputArgPopup(handles);
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes inputArgumentsGUI wait for user response (see UIRESUME)
uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = inputArgumentsGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1}=handles.OK;
varargout{2}=handles.ModuleStruct;
delete(hObject);

% --- Executes on selection change in listboxInputArgumens.
function listboxInputArgumens_Callback(hObject, eventdata, handles)
% hObject    handle to listboxInputArgumens (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listboxInputArgumens contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listboxInputArgumens
inputArgumentsSelChange(handles);

% --- Executes during object creation, after setting all properties.
function listboxInputArgumens_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listboxInputArgumens (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on selection change in popupModuleInstance.
function popupModuleInstance_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to popupModuleInstance (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
updateOutputArgPopup(handles);
%get the current selection
assay_list=get(handles.listboxInputArgumens,'String');
selection_idx=get(handles.listboxInputArgumens,'Value');
selection_text=assay_list{selection_idx};
%if an output argument is selected update its value
if (length(selection_text)>9)&&strcmp(selection_text(1:9),'<html><i>')
    %find the current argument
    i=1;
    arg_text=assay_list{selection_idx-i};
    while(~isempty(strfind(arg_text,' ')))
        i=i+1;
        arg_text=assay_list{selection_idx-i};
    end
    module_struct=handles.ModuleStruct;
    arg_nr=regexp(selection_text,'([0-9])','tokens','once');
    arg_nr=str2double(arg_nr{1});
    %get the argument indices
    output_idx=find(cellfun(@(x) strcmp(x{1},arg_text), module_struct.OutputArgs));
    output_idx=output_idx(arg_nr);
    output_arg=module_struct.OutputArgs{output_idx};
    module_idx=get(hObject,'Value');
    popup_list=get(hObject,'String');
    module_instance=popup_list{module_idx};
    output_arg{2}=module_instance;
    arg_idx=get(handles.popupOutputArgument,'Value');
    arg_list=get(handles.popupOutputArgument,'String');
    arg_name=arg_list{arg_idx};
    output_arg{3}=['''' arg_name ''''];
    module_struct.OutputArgs(output_idx)={output_arg};
    handles.ModuleStruct=module_struct;
    guidata(handles.figure1,handles);
end

% Hints: contents = get(hObject,'String') returns popupModuleInstance contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupModuleInstance


% --- Executes during object creation, after setting all properties.
function popupModuleInstance_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupModuleInstance (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on selection change in popupOutputArgument.
function popupOutputArgument_Callback(hObject, eventdata, handles)
% hObject    handle to popupOutputArgument (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupOutputArgument contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupOutputArgument
%get the current selection
assay_list=get(handles.listboxInputArgumens,'String');
selection_idx=get(handles.listboxInputArgumens,'Value');
selection_text=assay_list{selection_idx};
%if an output argument is selected update its value
if (length(selection_text)>9)&&strcmp(selection_text(1:9),'<html><i>')
    %find the current argument
    i=1;
    arg_text=assay_list{selection_idx-i};
    while(~isempty(strfind(arg_text,' ')))
        i=i+1;
        arg_text=assay_list{selection_idx-i};
    end
    module_struct=handles.ModuleStruct;
    arg_nr=regexp(selection_text,'([0-9])','tokens','once');
    arg_nr=str2double(arg_nr{1});
    %get the argument indices
    output_idx=find(cellfun(@(x) strcmp(x{1},arg_text), module_struct.OutputArgs));
```

```
    output_idx=output_idx(arg_nr);
    output_arg=module_struct.OutputArgs{output_idx};
    module_idx=get(hObject,'Value');
    popup_list=get(hObject,'String');
    module_output=popup_list{module_idx};
    output_arg(3)={['''' module_output '''']};
    module_struct.OutputArgs(output_idx)={output_arg};
    handles.ModuleStruct=module_struct;
    guidata(handles.figure1,handles);
end


% --- Executes during object creation, after setting all properties.
function popupOutputArgument_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupOutputArgument (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupInputArgument.
function popupInputArgument_Callback(hObject, eventdata, handles)
% hObject    handle to popupInputArgument (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupInputArgument contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupInputArgument


% --- Executes during object creation, after setting all properties.
function popupInputArgument_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupInputArgument (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editManualValue_Callback(hObject, eventdata, handles)
% hObject    handle to editManualValue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editManualValue as text
%        str2double(get(hObject,'String')) returns contents of editManualValue as a double


% --- Executes during object creation, after setting all properties.
function editManualValue_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editManualValue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbuttonOK.
function pushbuttonOK_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonOK (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.OK=true;
if strcmp(handles.SelectionType,'ArgValue')&&~handles.Erased
    handles.ModuleStruct=updateArgValue(handles);
end
guidata(handles.figure1,handles);
```

```
uiresume(handles.figure1);

% --- Executes on button press in pushbuttonCancel.
function pushbuttonCancel_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonCancel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
uiresume(handles.figure1);


function editStatus_Callback(hObject, eventdata, handles)
% hObject    handle to editStatus (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% Hints: get(hObject,'String') returns contents of editStatus as text
%        str2double(get(hObject,'String')) returns contents of editStatus as a double


% --- Executes during object creation, after setting all properties.
function editStatus_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editStatus (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbuttonAddOptArg.
function pushbuttonAddOptArg_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonAddOptArg (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbuttonRemOptArg.
function pushbuttonRemOptArg_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonRemOptArg (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbuttonAddManualValue.
function pushbuttonAddManualValue_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonAddManualValue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
addManualValue(handles);


% --- Executes on button press in pushbuttonAddOutputArgument.
function pushbuttonAddOutputArgument_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonAddOutputArgument (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
addOutputArgument(handles);


% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: delete(hObject) closes the figure
uiresume(hObject);



% --- Executes on button press in pushbuttonRemove.
function pushbuttonRemove_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonRemove (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
removeProvider(handles);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% inputArgumentsSelChange.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function inputArgumentsSelChange(handles)
%helper function for assayEditorGUI. callback for inputArgumentsGUI

if strcmp(handles.SelectionType,'ArgValue')&&~(handles.Erased)
    handles.ModuleStruct=updateArgValue(handles);
end
handles.Erased=false;
guidata(handles.figure1,handles);
input_strings=get(handles.listboxInputArgumens,'String');
selection_idx=get(handles.listboxInputArgumens,'Value');
selection_text=input_strings{selection_idx};
handles.SelectionIndex=selection_idx;
arg_text=selection_text;
i=0;
%find the current module instance
while(1)
    if (isempty(strfind(arg_text,' ')))
        break;
    else
        i=i+1;
    end
    arg_text=input_strings{selection_idx-i};
end

module_struct=handles.ModuleStruct;
if (i==0)
    %argument is selected show description
    %getArgumentDescription
    selection_type='ArgName';
    updateOutputArgPopup(handles);
else
    selection_type='ArgValue';
    if strcmp(selection_text(1:20),'<html><i> Value')
        handles.SelectionValue=get(handles.editManualValue,'String');
    end
    handles.ArgType=showArgValue(arg_text,selection_text,module_struct,handles);
end
handles.SelectionType=selection_type;
guidata(handles.figure1,handles);


%end inputArgumentsSelChange
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% intensityWatershed.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=intensityWatershed(input_args)
% Usage
% This module is used to compute the intensity watershed of a grayscale image.
% Input Structure Members
% Image - The intensity image for which the watershed will be computed.
% MarkerImage - Optional image which will be used to make sure that local
% minima occur only in the regions indicated in the marker image.
% Output Structure Members
% LabelMatrix - The result of the watershed.

field_names=fieldnames(input_args);
img_intensity=input_args.Image.Value;
if (max(strcmp(field_names,'MarkerImage')))
    img_marker=input_args.MarkerImage.Value;
    img_new=imimposemin(img_intensity,img_marker);
    output_args.LabelMatrix=watershed(img_new);
else
    output_args.LabelMatrix=watershed(img_intensity);
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% invertSelection.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function invertSelection()
%helper function for manual segmentation review GUI. used to select the
%objects not currently selected
global msr_gui_struct;

switch (msr_gui_struct.CurrentAction)
    case 'SelectBlob'
        selected_ids=msr_gui_struct.SelectedBlobID;
        cur_lbl=msr_gui_struct.BlobsLabel;
    case 'SelectObject'
        selected_ids=msr_gui_struct.SelectedObjectIDs;
        cur_lbl=msr_gui_struct.ObjectsLabel;
    otherwise
        return;
end

unique_ids=unique(cur_lbl(:));
unique_ids(1)=[];
inverse_idx=~(ismember(unique_ids,selected_ids));
inverse_ids=unique_ids(inverse_idx);
selection_idx=ismember(cur_lbl,inverse_ids);
selection_mask=repmat(selection_idx,[1 1 3]);
image_data=label2rgb(msr_gui_struct.ObjectsLabel,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,
'shuffle');
image_data(selection_mask)=createCheckerBoardPattern(selection_idx);
set(msr_gui_struct.ImageHandle,'CData',image_data);

switch (msr_gui_struct.CurrentAction)
    case 'SelectBlob'
        msr_gui_struct.SelectedBlobID=inverse_ids;
    case 'SelectObject'
        msr_gui_struct.SelectedObjectIDs=inverse_ids;
end

%end invertSelection
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% isChainExpanded.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [is_expanded selection_level]=isChainExpanded(listbox_list,selection_idx)
%helper function for assayEditorGUI. determine the state of the current chain: expanded or
collapsed
selection_text=listbox_list{selection_idx};
selection_level=getSelectionLevel(selection_text);
if (selection_idx==length(listbox_list))
    %selection is at bottom of the list so expand
    is_expanded=false;
    return;
end

%get the level of the instance below to determine the current state
prev_level=getSelectionLevel(listbox_list{selection_idx+1});

if (selection_level>=prev_level)
    is_expanded=false;
else
    is_expanded=true;
end

%end isChainExpanded
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% isEmptyFunction.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=isEmptyFunction(input_args)
%wrapper module for Matlab isempty function
%Input Structure Members
%TestVariable - The variable to be tested.
%Output Structure Members
%Boolean - The result of the test.
output_args.Boolean=isempty(input_args.TestVariable.Value);

%end is_empty_function
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% isEqualFunction.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=isEqualFunction(input_args)
%test two variables for equality

output_args.Boolean=(input_args.Var1.Value==input_args.Var2.Value);

%end isEqualFunction
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% isNotEmptyFunction.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=isNotEmptyFunction(input_args)
%module wrapper for is not empty Matlab function
%Input Structure Members
%TestVariable - The variable to be tested.
%Output Structure Members
%Boolean - The result of the test.
output_args.Boolean=~isempty(input_args.TestVariable.Value);

%end is_empty_function
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% joinFragmentedBlob.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function joined_blob=joinFragmentedBlob(fragmented_blob)
%helper function to join the parts of a fragmented blob
joined_blob=fragmented_blob;
blob_boundaries=bwboundaries(fragmented_blob,8,'noholes');
nr_blobs=length(blob_boundaries);
%calculate the minimum sampling distance
%we want the crust algorithm to close edges around the disjointed blobs
%so the minimum sampling distance will be the max(min_distance) between the
%point sets
%determine the nearest blob to each individual blob
min_distances=ones(nr_blobs)*Inf;
intersets_edge_indexes=cell(nr_blobs);
min_distance_indexes=zeros(nr_blobs);
for i=1:(nr_blobs-1)
    for j=(i+1):nr_blobs
        if (i==j)
            continue;
        end
        [min_distance min_distance_index
interset_edge_indexes]=getMinDistanceBetweenPointSets(blob_boundaries{i},blob_boundaries{j});
        min_distances(i,j)=min_distance;
        min_distance_indexes(i,j)=min_distance_index;
        intersets_edge_indexes{i,j}=interset_edge_indexes;
    end
end

[dummy nearest_blob_idx]=min(min_distances,[],2);

blob_sz=size(fragmented_blob);
for i=1:nr_blobs-1
    j=nearest_blob_idx(i);
    join_polygon=getJoinPolygon(blob_boundaries{i},...
        blob_boundaries{j}, intersets_edge_indexes{i,j}, min_distance_indexes(i,j));
    %create the image of the join
    img_join=poly2mask(join_polygon(2,:),join_polygon(1,:),blob_sz(1),blob_sz(2));
    %merge it with the blob image
    joined_blob=joined_blob|img_join;
end



%end joinFragmentedBlob
end

function join_polygon=getJoinPolygon(set_a, set_b, interset_edge_indexes, min_distance_index)
%get the coordinates of the polygon points from set a and set b that will form the join
%between the two boundaries
%put all the indexes from set a in one vector and all vectors from set b in
%another. the indexes for set b are offset by the size of set a.
set_a_sz=size(set_a,1);
temp_indexes=interset_edge_indexes;
set_b_indexes_idx=interset_edge_indexes>set_a_sz;
```

```
temp_indexes(set_b_indexes_idx)=0;
set_a_indexes=temp_indexes(:,1)+temp_indexes(:,2);
temp_indexes=interset_edge_indexes;
temp_indexes(~set_b_indexes_idx)=0;
set_b_indexes=temp_indexes(:,1)+temp_indexes(:,2);
%remove the set_b indexes offset
set_b_indexes=set_b_indexes-set_a_sz;
%get the min distance points
point_a=set_a(set_a_indexes(min_distance_index),:);
point_b=set_b(set_b_indexes(min_distance_index),:);
%caculate the orientation of the minimum distance line
min_distance_angle=atan2(point_a(:,2)-point_b(:,2),point_a(:,1)-point_b(:,1));
%calculate the orientation of all the inter-set lines
edge_angles=atan2(set_a(set_a_indexes,2)-set_b(set_b_indexes,2),set_a(set_a_indexes,1)-
set_b(set_b_indexes,1));
%get all the edges that have the same orientation
same_orientation_idx=(edge_angles==min_distance_angle);
new_min_distance_index=find(same_orientation_idx)==min_distance_index;
%select edges of the same orientation with the min
%distance line as this is where the join will be made
set_a_select_edges=set_a_indexes(same_orientation_idx);
set_b_select_edges=set_b_indexes(same_orientation_idx);
set_a_selected_points=set_a(set_a_select_edges,:);
set_b_selected_points=set_b(set_b_select_edges,:);
if (size(set_a_selected_points,1)==1)
    point_a=set_a_selected_points';
    point_b=set_b_selected_points';
    line_slope=(point_a(1)-point_b(1))/(point_a(2)-point_b(2));
    if (line_slope>0)
        line_offset=[2;-2];
    else
        line_offset=[2;2];
    end
    join_polygon=[point_a+line_offset point_b+line_offset point_b-line_offset point_a-line_offset
point_a+line_offset];
    return;
end
a_clusters=getPointsClusters(set_a_selected_points);
b_clusters=getPointsClusters(set_b_selected_points);
%find out which clusters contains the min distance points
min_distance_a_cluster_idx=a_clusters(new_min_distance_index);
min_distance_b_cluster_idx=b_clusters(new_min_distance_index);
edges_to_keep_idx=(a_clusters==min_distance_a_cluster_idx)&(b_clusters==min_distance_b_cluster_id
x);
set_a_select_edges=set_a_select_edges(edges_to_keep_idx);
set_b_select_edges=set_b_select_edges(edges_to_keep_idx);
%get the min and max points
%xmin
[set_a_x_min set_a_x_min_idx]=min(set_a(set_a_select_edges,1));
[set_b_x_min set_b_x_min_idx]=min(set_b(set_b_select_edges,1));
if (set_a_x_min<set_b_x_min)
    x_min_idx=set_a_x_min_idx;
else
    x_min_idx=set_b_x_min_idx;
end
x_min_edge_indexes=[set_a_select_edges(x_min_idx) set_b_select_edges(x_min_idx)];
edge_1_points=[set_a(x_min_edge_indexes(1),:)' set_b(x_min_edge_indexes(2),:)'];
b_found_edges=false;
%xmax
[set_a_x_max set_a_x_max_idx]=max(set_a(set_a_select_edges,1));
[set_b_x_max set_b_x_max_idx]=max(set_b(set_b_select_edges,1));
if (set_a_x_max>set_b_x_max)
    x_max_idx=set_a_x_max_idx;
else
    x_max_idx=set_b_x_max_idx;
end
x_max_edge_indexes=[set_a_select_edges(x_max_idx) set_b_select_edges(x_max_idx)];
edge_points=[set_a(x_max_edge_indexes(1),:)' set_b(x_max_edge_indexes(2),:)'];
if (sum(edge_1_points-edge_points))
    b_found_edges=true;
    edge_2_points=edge_points;
end

%ymin
if (~b_found_edges)
    [set_a_y_min set_a_y_min_idx]=min(set_a(set_a_select_edges,2));
    [set_b_y_min set_b_y_min_idx]=min(set_b(set_b_select_edges,2));
    if (set_a_y_min<set_b_y_min)
        y_min_idx=set_a_y_min_idx;
    else
        y_min_idx=set_b_y_min_idx;
```

```
    end
    y_min_edge_indexes=[set_a_select_edges(y_min_idx) set_b_select_edges(y_min_idx)];
    edge_points=[set_a(y_min_edge_indexes(1),:)' set_b(y_min_edge_indexes(2),:)'];
    if (sum(edge_1_points-edge_points))
        b_found_edges=true;
        edge_2_points=edge_points;
    end
end

%ymax
if (~b_found_edges)
    [set_a_y_max set_a_y_max_idx]=max(set_a(set_a_select_edges,2));
    [set_b_y_max set_b_y_max_idx]=max(set_b(set_b_select_edges,2));
    if (set_a_y_max<set_b_y_max)
        y_max_idx=set_a_y_max_idx;
    else
        y_max_idx=set_b_y_max_idx;
    end
    if (sum(edge_1_points-edge_points))
        b_found_edges=true;
        y_max_edge_indexes=[set_a_select_edges(y_max_idx) set_b_select_edges(y_max_idx)];
        edge_points=[set_a(y_max_edge_indexes(1),:)' set_b(y_max_edge_indexes(2),:)'];
        edge_2_points=edge_points;
    else
        %connection is only a line
        edge_2_points=edge_1_points;
    end

end
if (b_found_edges)
    join_polygon=[edge_1_points edge_2_points(:,2) edge_2_points(:,1) edge_1_points(:,1)];
    %need to subtract and add half a pixel or polymask will not included the
    %edge of the polygon
    x_lower_half_idx=join_polygon(1,:)<median(join_polygon(1,1:(end-1)));
    join_polygon(1,x_lower_half_idx)=join_polygon(1,x_lower_half_idx)-1;
    join_polygon(1,~x_lower_half_idx)=join_polygon(1,~x_lower_half_idx)+1;
    y_lower_half_idx=join_polygon(2,:)<median(join_polygon(2,1:(end-1)));
    join_polygon(2,y_lower_half_idx)=join_polygon(2,y_lower_half_idx)-1;
    join_polygon(2,~y_lower_half_idx)=join_polygon(2,~y_lower_half_idx)+1;
else
    %connection is only a line so we'll make a very narrow polygon
    %offset two pixels on each side of the line
    point_a=edge_1_points(:,1);
    point_b=edge_1_points(:,2);
    line_slope=(point_a(1)-point_b(1))/(point_a(2)-point_b(2));
    if (line_slope>0)
        line_offset=[2;-2];
    else
        line_offset=[2;2];
    end
    join_polygon=[point_a+line_offset point_b+line_offset point_b-line_offset point_a-line_offset
point_a+line_offset];
end

%end preparePointSetPairsForJoin
end

function linkage_clusters=getPointsClusters(point_set)
%test and make sure the point sets are not equal
set_sz=size(point_set);
test_set=repmat(point_set(1,:),set_sz(1),1);
if isequal(point_set,test_set)
    linkage_clusters=ones(set_sz(1),1);
    return;
end
blob_dist=pdist(point_set);
%tested all linkage params - this works best
blob_linkage=linkage(blob_dist,'average');
linkage_clusters=cluster(blob_linkage,'criterion','distance','cutoff',5);

%end getPointsClusters
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% joinObjects.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function joinObjects()
%helper function for manual segmentation review module. join objects in a
%label matrix to form a single possibly fragmented object.
global msr_gui_struct;
```

```
objects_lbl=msr_gui_struct.ObjectsLabel;
join_ids=sort(msr_gui_struct.SelectedObjectID);
join_ids_len=length(join_ids);
if (length(join_ids)<2)
    warnDlg('Multiple objects need to be selected!');
    return;
end

%set all the ids to the min id
for i=2:join_ids_len
    objects_lbl(objects_lbl==join_ids(i))=join_ids(1);
end
image_handle=msr_gui_struct.ImageHandle;
image_data=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
set(image_handle,'CData',image_data);
msr_gui_struct.ObjectsLabel=objects_lbl;
updateReviewSegGUIStatus('SelectObject');

%end completeJoinObjects
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% keyPressInManualSegmentationGUI.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function keyPressInManualSegmentationGUI()
%helper function in manual segmentation review GUI. detect key presses and
%take various actions
global msr_gui_struct;
gui_handle=msr_gui_struct.GuiHandle;
char_pressed=get(gui_handle,'CurrentCharacter');

switch (msr_gui_struct.CurrentAction)
    case 'ResegmentBlob'
        if (char_pressed=='d')
            resegmentBlob('complete');
        elseif (char_pressed=='q')
            msr_gui_struct.CurrentAction='SelectBlob';
        elseif (char_pressed=='n')

msr_gui_struct.CurrentResegmentationIndex=msr_gui_struct.CurrentResegmentationIndex+1;
        end
    otherwise
        return;
end

%end keyPressInManualSegmentationGUI
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% labelObjects.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=labelObjects(input_args)
%simple wrapper module for bwlabeln function
%Input Structure Members
%Image - The image to be processed.
%Output Structure Members
%LabelMatrix - The resulting label matrix.
output_args.LabelMatrix=bwlabeln(input_args.Image.Value);

%end labelObjects
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% loadAncestry.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=loadAncestry(input_args)
%load the ancestry data
%Input Structure Members
%FileName - The path to the ancestry .mat file.
%Output Structure Members
%Ancestry - Array containing the ancestry data.
load(input_args.FileName.Value);
output_args.Ancestry=cells_ancestry;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% loadAncestryLayout.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=loadAncestryLayout(input_args)
% Usage
% This module is used to load a structure containing the order of each column in the ancestry
matrix from a MATLAB .mat file.
% Input Structure Members
% FileName - The name of the .mat file containing the ancestry layout structure.
% Output Structure Members
% AncestryLayout - Structure containing the order of each column in the ancestry matrix.

load(input_args.FileName.Value);
output_args.AncestryLayout=ancestry_layout;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% loadCellProfilerTracks.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=loadCellProfilerTracks(input_args)

cp_data=load(input_args.CPDataFile.Value);
tracks_data=cp_data.handles.Measurements.Nuclei;
obj_ids=tracks_data.TrackObjects_Label;
centr_1=tracks_data.Location_Center_X;
centr_2=tracks_data.Location_Center_Y;
tracks=[];
for i=1:length(obj_ids)
    tracks=[tracks; [obj_ids{i} (i-1)*ones(length(obj_ids{i}),1) centr_2{i} centr_1{i}]];
end
output_args.Tracks=tracks;
tracks_layout.TrackIDCol=1;
tracks_layout.TimeCol=2;
tracks_layout.Centroid1Col=3;
tracks_layout.Centroid2Col=4;
output_args.TracksLayout=tracks_layout;


%end loadCellProfilerTracks
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% loadCellsLabel.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=loadCellsLabel(input_args)
% Usage
% This module is used to load cells/nuclei labels. It looks for a variable named cells_lbl in the
.mat data file.
% Input Structure Members
% MatFileName - The name of the data file.
%
% Output Structure Members
% LabelMatrix - The label matrix loaded from the file.


mat_file_name=input_args.FileName.Value;
try
    load_struct=load(mat_file_name);
catch
    warning(['Failed to load ' mat_file_name '! loadCellsLabel will return an empty image.']);
    output_args.LabelMatrix=[];
    return;
end
output_args.LabelMatrix=load_struct.cells_lbl;

%end loadCellsLabel
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% loadColormap.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=loadColormap(input_args)
%load the colormap used to generate the overlayed cells outlines
% Input Structure Members
% FileName - The name of the data file containing the colormap data.
%
```

```
% Output Structure Members
% Colormap - The  colormap data loaded from the file.
load(input_args.FileName.Value);
output_args.Colormap=cmap;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% loadOffsets.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=loadOffsets(input_args)
%load the offset data from file
% Input Structure Members
% FileName - The name of the data file containing the offset data.
%
% Output Structure Members
% Offsets - The  offset data loaded from the file.

load(input_args.FileName.Value);
output_args.Offsets=xy_offsets;

%end loadOffsets
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% loadTracks.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=loadTracks(input_args)
%load the tracks file
% Input Structure Members
% FileName - The name of the data file containing the track data.
%
% Output Structure Members
% Tracks - Matrix containing the  track data loaded from the file.
load(input_args.FileName.Value);
output_args.Tracks=tracks;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% loadTracksLayout.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=loadTracksLayout(input_args)
% Usage
% This module is used to load a structure containing the order of each column in the tracks
matrix from a MATLAB .mat file.
% Input Structure Members
% FileName - The name of the .mat file containing the tracks layout structure.
% Output Structure Members
% TracksLayout - Structure containing the order of each column in the tracks matrix.

load(input_args.FileName.Value);
output_args.TracksLayout=tracks_layout;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% makeAncestryForCellsEnteringFrames.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=makeAncestryForCellsEnteringFrames(input_args)
% Usage
% This module is used to add ancestry records for cells entering the field of view after the
first frame (not the result of a mitotic event in the field of view).
% Input Structure Members
% CellsAncestry - Current cell ancestry records.
% FirstFrameIDs - The IDs of tracks starting in the first frame.
% SplitCells - The IDs of tracks that are the result of mitosis.
% TimeCol - The index of the time column in the tracks matrix.
% TrackIDCol - The index of the track ID column in the tracks matrix.
% TrackIDs - The IDs of all the tracks.
% Tracks - The matrix containing all the tracks.
% Output Structure Members
% CellsAncestry - The current cell ancestry record with ancestry of cells entering the field of
view after the first frame appended to the end.


split_cells=input_args.SplitCells.Value;
```

```
track_ids=input_args.TrackIDs.Value;
first_frame_ids=input_args.FirstFrameIDs.Value;
trackIDCol=input_args.TrackIDCol.Value;
timeCol=input_args.TimeCol.Value;
tracks=input_args.Tracks.Value;
cells_ancestry=input_args.CellsAncestry.Value;

if (isempty(split_cells))
    cells_entering_frame_ids=setdiff(track_ids,first_frame_ids);
else
    cells_entering_frame_ids=setdiff(track_ids,[first_frame_ids; split_cells(:,2)]);
end
cells_entering_frame_len=length(cells_entering_frame_ids);
start_times=zeros(cells_entering_frame_len,1);
stop_times=zeros(cells_entering_frame_len,1);
for i=1:cells_entering_frame_len
    track_times=tracks(tracks(:,trackIDCol)==cells_entering_frame_ids(i),timeCol);
    start_times(i)=track_times(1);
    stop_times(i)=track_times(end);
end
output_args.CellsAncestry=[cells_ancestry; [cells_entering_frame_ids
zeros(cells_entering_frame_len,1)...
    ones(cells_entering_frame_len,1) start_times stop_times]];

%end makeAncestryForCellsEnteringFrames
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% makeAncestryForFirstFrameCells.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=makeAncestryForFirstFrameCells(input_args)
% Usage
% This module is used to create ancestry records for cells present in the first frame of a time-
lapse movie.  These cells are generation zero.
% Input Structure Members
% TimeCol - The index of the time column in the tracks matrix.
% TrackIDCol - The index of the track ID column in the tracks matrix.
% TrackIDs - The IDs of all the tracks.
% Tracks - The matrix containing all the tracks.
% Output Structure Members
% CellsAncestry - The ancestry records for the cells in the first frame.
% FirstFrameIDs - IDs of cells in the first frame.
% UntestedIDs - IDs of cells in the movie that were not present in the first frame.

timeCol=input_args.TimeCol.Value;
trackIDCol=input_args.TrackIDCol.Value;
tracks=input_args.Tracks.Value;
track_ids=input_args.TrackIDs.Value;

first_frame_ids=tracks(tracks(:,timeCol)==0,trackIDCol);
first_frame_ids_len=length(first_frame_ids);
stop_times=zeros(first_frame_ids_len,1);
for i=1:first_frame_ids_len
    track_times=tracks(tracks(:,trackIDCol)==first_frame_ids(i),timeCol);
    stop_times(i)=track_times(end);
end
output_args.CellsAncestry=[first_frame_ids zeros(first_frame_ids_len,1)
ones(first_frame_ids_len,1)...
    zeros(first_frame_ids_len,1) stop_times];
output_args.UntestedIDs=setdiff(track_ids,first_frame_ids);
output_args.FirstFrameIDs=first_frame_ids;

%end makeAncestryForFirstFrameCells
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% makeContinuousLabelMatrix.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function continuous_lbl=makeContinuousLabelMatrix(objects_lbl)
%make the label matrix ids sequential as a number of functions expect it
%that way

continuous_lbl=objects_lbl;
label_ids=unique(objects_lbl);
%remove 0 the background id
label_ids(1)=[];
ids_nr=length(label_ids);
```

```
if (ids_nr==label_ids(end))
    %object label matrix is continuous
    return;
end
skipped_ids_exist=false;
for i=1:ids_nr
    if (label_ids(i)~=i)
        skipped_ids_exist=true;
    end
    if (skipped_ids_exist)
        continuous_lbl(continuous_lbl==label_ids(i))=i;
    else
        continue;
    end
end

%end makeContinuousLabel
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% makeDependencies.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function makeDependencies(parent_function_struct)
%core CellAnimation function. used to make a dependency tree from the module list.
global functions_list;
global dependencies_list;
global dependencies_index;
nr_functions=size(functions_list,1);
prev_functions_nr=size(dependencies_list,1);
dependencies_list=[dependencies_list; cell(nr_functions,1)];
for i=1:nr_functions
    function_struct=functions_list{i};
    dependency_struct=function_struct;
    dependency_struct.DependentFunctions={};
    %deal with special control functions differently
    switch(char(function_struct.FunctionHandle))
        case 'forLoop'
            temp=functions_list;
            functions_list=function_struct.LoopFunctions;
            dependency_struct=getDependencies(dependency_struct,0,parent_function_struct);
            makeDependencies(function_struct);
            functions_list=temp;
        case 'whileLoop'
            temp=functions_list;
            functions_list=function_struct.LoopFunctions;
            dependency_struct=getDependencies(dependency_struct,0,parent_function_struct);
            makeDependencies(function_struct);
            functions_list=temp;
        case 'if_statement'
            temp=functions_list;
            functions_list=[function_struct.IfFunctions; function_struct.ElseFunctions];
            dependency_struct=getDependencies(dependency_struct,0,parent_function_struct);
            makeDependencies(function_struct);
            functions_list=temp;
    end

dependencies_list{prev_functions_nr+i}=getDependencies(dependency_struct,i,parent_function_struct
);
    dependencies_index.put(dependency_struct.InstanceName,prev_functions_nr+i);
end

%end makeDependencies
end

function dependency_struct=getDependencies(dependency_struct,this_idx,parent_function_struct)
global functions_list;
instance_name=dependency_struct.InstanceName;
dependent_functions=dependency_struct.DependentFunctions;
if (~isempty(parent_function_struct))
    [b_dependent dependency_item]=makeDependencyRecord(instance_name,parent_function_struct);
    [b_dependent
dependency_item]=addValuesToKeep(instance_name,parent_function_struct,dependency_item,
b_dependent);
    if (b_dependent)
        dependent_functions=[dependent_functions; {dependency_item}];
    end
end
for i=1:size(functions_list,1)
    [b_dependent dependency_item]=makeDependencyRecord(instance_name,functions_list{i});
    if (b_dependent)
```

```
        dependent_functions=[dependent_functions; {dependency_item}];
    end
end
dependency_struct.DependentFunctions=dependent_functions;

%end getDependencies
end

function [b_dependent dependency_item]=makeDependencyRecord(instance_name,function_struct)
%if function_struct has any arguments that depend on the function for which
%we're building the dependency chain create a dependency record

field_names=fieldnames(function_struct);
if max(strcmp(field_names,'FunctionArgs'))
    function_args=function_struct.FunctionArgs;
else
    function_args={};
end
function_instance=function_struct.InstanceName;
b_first_dep=true;
b_dependent=false;
dependency_item=[];
if isempty(function_args)
    return;
else
    field_names=fieldnames(function_args);
end
dependent_args={};
for i=1:size(field_names,1)
    arg_struct=function_args.(char(field_names{i}));
    struct_field_names=fieldnames(arg_struct);
    function_instances_idx=strncmp('FunctionInstance',struct_field_names,16);
    dependency_field_names=struct_field_names(function_instances_idx);
    arg_field_names=struct_field_names(circshift(function_instances_idx,1));
    for j=1:size(dependency_field_names,1)
        if strcmp(instance_name,arg_struct.(char(dependency_field_names{j})))
            %this argument is a dependency - add it to the list
            if (b_first_dep)
                %create a dependency record
                dependency_item.InstanceName=function_instance;
                b_first_dep=false;
            end
            dependent_arg.ArgumentName=field_names{i};
            if (strncmp('OutputArg',arg_field_names{j},9))
                dependent_arg.OutputArg=arg_struct.(arg_field_names{j});
                dependent_arg.Type=1; %output
            else
                dependent_arg.InputArg=arg_struct.(arg_field_names{j});
                dependent_arg.Type=2; %input
            end
            dependent_arg.DependencyType=1; %functionargs
            dependent_args=[dependent_args; {dependent_arg}];
            break;
        end
    end
end
if (~b_first_dep)
    b_dependent=true;
    dependency_item.DependentArgs=dependent_args;
end

end

function [b_dependent
dependency_item]=addValuesToKeep(instance_name,parent_function_struct,dependency_item,
b_dependent)

parent_fields=fieldnames(parent_function_struct);
if (max(strcmp('KeepValues',parent_fields))==0)
    %parent doesn't want to save any values
    return;
end
parent_instance=parent_function_struct.InstanceName;
if (b_dependent)
    b_initially_dependent=true;
    b_first_dep=false;
else
    b_initially_dependent=false;
    b_first_dep=true;
end
values_to_keep=parent_function_struct.KeepValues;
```

```
values_names=fieldnames(values_to_keep);
b_found_dependency=false;
dependent_args={};
for i=1:size(values_names,1)
    value_struct=values_to_keep.(char(values_names{i}));
    value_struct_field_names=fieldnames(value_struct);
    function_instances_idx=strncmp('FunctionInstance',value_struct_field_names,16);
    dependency_field_names=value_struct_field_names(function_instances_idx);
    arg_field_names=value_struct_field_names(circshift(function_instances_idx,1));
    for j=1:size(dependency_field_names,1)
        if strcmp(instance_name,value_struct.(char(dependency_field_names{j})))
            %this argument is a dependency - add it to the list
            b_found_dependency=true;
            if (b_first_dep)
                %create a dependency record
                dependency_item.InstanceName=parent_instance;
                b_first_dep=false;
            end
            dependent_arg.ArgumentName=values_names{i};
            if (strncmp('OutputArg',arg_field_names{j},9))
                dependent_arg.OutputArg=value_struct.(arg_field_names{j});
                dependent_arg.Type=1; %output
            else
                dependent_arg.InputArg=value_struct.(arg_field_names{j});
                dependent_arg.Type=2; %input
            end
            dependent_arg.DependencyType=2; %keepvalue
            dependent_args=[dependent_args; {dependent_arg}];
            break;
        end
    end
end

if (b_found_dependency)
    if (b_initially_dependent)
        dependency_item.DependentArgs=[dependency_item.DependentArgs; dependent_args];
    else
        b_dependent=true;
        dependency_item.DependentArgs=dependent_args;
    end
end

%end addValuesToKeep
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% makeExcludedTracksList.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=makeExcludedTracksList(input_args)
% Usage
% This module wraps its input, a list of cell IDs, in a MATLAB cell array.
% Input Structure Members
% UnassignedCellsIDs - The list of cell IDs.
% Output Structure Members
% ExcludedTracks - The MATLAB cell array.

output_args.ExcludedTracks=cell(size(input_args.UnassignedCellsIDs.Value,1),1);

%end makeExcludedTracksList
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% makeImgFileName.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=makeImgFileName(input_args)
% Usage
% This module is used to build the filename of a frame from a time-lapse movie using the root
file name, current frame number and a specified number format and file extension.
% Input Structure Members
% CurFrame - The index of the current frame.
% FileBase - The root of the image file name.
% FileExt - The file extension.
% NumberFmt - The number format to be used. See MATLAB sprintf documentation for format
documentation.
% Output Structure Members
% FileName - String containing the resulting file name.
```

```
file_base=input_args.FileBase.Value;
cur_frame=input_args.CurFrame.Value;
number_fmt=input_args.NumberFmt.Value;
file_ext=input_args.FileExt.Value;
output_args.FileName=[file_base num2str(cur_frame,number_fmt) file_ext];

%end makeImgFileName
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% makeOutputStruct.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=makeOutputStruct(function_struct)
%core CellAnimation function. create a CellAnimation output structure
global dependencies_list;
global dependencies_index;
output_args=[];

function_idx=dependencies_index.get(function_struct.InstanceName);
dependency_struct=dependencies_list{function_idx};
function_field_names=fieldnames(dependency_struct);
if (max(strcmp('KeepValues',function_field_names))==0)
    %parent doesn't want to save any values
    return;
end

field_names=fieldnames(dependency_struct.KeepValues);
for i=1:size(field_names,1)
    arg_name=field_names{i};
    keep_value_fields=fieldnames(dependency_struct.KeepValues.(arg_name));
    if max(strcmp(keep_value_fields,'Value'))
        %a value has been filled in so pass it along
        output_args.(arg_name)=dependency_struct.KeepValues.(arg_name).Value;
    end
end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% makeUnassignedCellsList.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=makeUnassignedCellsList(input_args)
% Usage
% This module is used to create a list of IDs for each cell centroid in a list.
% Input Structure Members
% CellsCentroids - List of cell centroids.
% Output Structure Members
% UnassignedCellsIDs - List of IDs.

output_args.UnassignedCellsIDs=[1:size(input_args.CellsCentroids.Value,1)]';

%end makeUnassignedCellsList
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% manageChainText.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function new_list=manageChainText(listbox_list,selection_idx,modules_list,modules_map)
%helper function for assayEditorGUI. collapse/expand the submodule list for the current module
[is_expanded selection_level]=isChainExpanded(listbox_list,selection_idx);
if (is_expanded)
    new_list=collapseText(listbox_list,selection_idx,selection_level);
else
    new_list=expandText(listbox_list,selection_idx,modules_list,modules_map,selection_level);
end

%end manageChainText
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% manageSelectionLayers.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function manageSelectionLayers()
%helper function for manual tracking review. used to start the selection layer GUI and update the
%selection layers
```

```
global sl_gui_struct;
global mtr_gui_struct;
sl_gui_struct=[];

selection_names=getSelectionNames();
sl_gui_struct.SelectionNames=selection_names;
gui_handle=selectionLayersGUI();
children_handles=get(gui_handle,'children');
sl_gui_struct.SelectionLayers=mtr_gui_struct.SelectionLayers;
sl_gui_struct.ListboxSelectionLayersHandle=findobj(children_handles,'tag','listboxSelectionLayers
');
set(sl_gui_struct.ListboxSelectionLayersHandle,'String',selection_names);
waitfor(gui_handle);

mtr_gui_struct.SelectionLayers=sl_gui_struct.SelectionLayers;
clear sl_gui_struct;
updateTrackImage(mtr_gui_struct.CurFrame,mtr_gui_struct.ShowLabels,mtr_gui_struct.ShowOutlines);
addSelectionLayers();

%end manageSelectionLayers
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% manualSegmentationReview.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=manualSegmentationReview(input_args)
%Usage
%This module is used to manually correct errors in automatic segmentation. The module loads
%a GUI for each frame with all the available segmentation corrections. To start the user has to
%
%select between blob correction and object correction by clicking on the "Select Blob" or "Select
%Object" button. A blob is a contiguous region as defined by the background pixels and it may
%contain one or more objects. An object is the set of pixels with the same non-zero ID in the
label
%matrix. An object may span multiple blobs.
%
%In general (with the exception of the "Restore Blob" operation), an object or blob must be
%selected before an operation can be performed on it. Selection is performed by clicking on the
%object or blob of interest. A selected item is indicated by a checkerboard pattern. If the
"Select
%Multiple" box is checked clicking on an unselected item adds it to the selection.
%
%The types of operations that may be performed on a blob are resegmentation, deletion and
%restoration. To resegment a blob one needs to indicate how many objects the new blob will
%contain and their approximate boundaries. Clicking on the selected blob after the "Resegment
%Blob" button has been pressed indicates that the pixels at those locations belong to the first
%object in the blob. To move to the next object press the letter "n" on the keyboard and click
%within the blob to indicate rough boundaries. The boundaries do not have to be specified
%precisely and a blob may be separated into two objects in as few as two clicks. Once all
%the objects have been defined press the letter "d" on the keyboard and the blob will be
%resegmented. During resegmentaion all the pixels in the blob are assigned to objects using a
%nearest-neighbor classifier based on the pixels selected by the user. A blob may be deleted by
%selecting it and then clicking the "Remove Blob" button. To restore a blob removed by mistake
%click on the "Restore Blob" button. The GUI will display the "Raw Label" image which shows all
%the blobs present after thresholding before any removal by filters or manual deletions. Choose
%the blob to restore by clicking on it.
%
%Two types of operations can be performed on objects: joining and deletion. To join a number of
%objects into a single object click on the "Join Objects" button then select the objects you want
to
%join by clicking on them. When you are done with object selection and want to join the objects
%press the "d" letter on the keyboard. To delete an object select it then click on the "Remove
%Object" button.
%
%Once all the corrections have been performed click on the "Save Changes & Continue" button to
%save your changes and move on to the next frame.
%
%Input Structure Members
%Image - The microscopy image for the current ObjectsLabel.
%ObjectsLabel - The label matrix containing the objects for which the automatic segmentation will
%be evaluated or corrected.
%PreviousLabel - The label matrix containing objects from the previous time step.
%RawLabel - The label matrix containing the objects before filtering.
%
%Output Structure Members
%LabelMatrix - The label matrix containing manual corrections if any.


global msr_gui_struct;
```

```
objects_lbl=input_args.ObjectsLabel.Value;
raw_lbl=input_args.RawLabel.Value;
previous_lbl=input_args.PreviousLabel.Value;
msr_gui_struct.ColorMap='colorcube';
msr_gui_struct.BkgColor=[0.7 0.7 0];
msr_gui_struct.ErrorTypes=[];
msr_gui_struct.ErrorBlobIDs=[];
msr_gui_struct.TotalErrors=0;
msr_gui_struct.CurrentAction='';
msr_gui_struct.ObjectsLabel=objects_lbl;
msr_gui_struct.BlobsLabel=bwlabeln(objects_lbl>0);
msr_gui_struct.OriginalObjectsLabel=raw_lbl;
msr_gui_struct.OriginalBlobsLabel=bwlabeln(raw_lbl>0);
msr_gui_struct.PreviousLabel=previous_lbl;
msr_gui_struct.Image=input_args.Image.Value;
msr_gui_struct.SelectMultiple=false;
msr_gui_struct.SelectedBlobID=[];
msr_gui_struct.SelectedObjectID=[];
msr_gui_struct.SnapToNearest=true;
%initialize the gui
field_names=fieldnames(msr_gui_struct);
gui_handle=findall(0,'Tag','ManualResegmentation');
if (~isempty(gui_handle))
    close(gui_handle);
end
if (max(strcmp('FigurePosition',field_names)))

msr_gui_struct.GuiHandle=manualSegmentationReviewGUI('Position',msr_gui_struct.FigurePosition);
else
    msr_gui_struct.GuiHandle=manualSegmentationReviewGUI;
end
gui_handle=msr_gui_struct.GuiHandle;
children_handles=get(gui_handle,'children');
objects_rgb=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
objects_axes_handle=findobj(children_handles,'tag','objectsAxes');
msr_gui_struct.AxesHandle=objects_axes_handle;
msr_gui_struct.SelectBlobButtonHandle=findobj(children_handles,'tag','selectBlobButton');
msr_gui_struct.SelectObjectButtonHandle=findobj(children_handles,'tag','selectObjectButton');
msr_gui_struct.ResegmentBlobButtonHandle=findobj(children_handles,'tag','resegmentBlobButton');
msr_gui_struct.RemoveBlobButtonHandle=findobj(children_handles,'tag','removeBlobButton');
msr_gui_struct.RestoreBlobButtonHandle=findobj(children_handles,'tag','restoreBlobButton');
msr_gui_struct.JoinObjectsButtonHandle=findobj(children_handles,'tag','joinObjectsButton');
msr_gui_struct.RemoveObjectButtonHandle=findobj(children_handles,'tag','removeObjectButton');
msr_gui_struct.StatusTextHandle=findobj(children_handles,'tag','statusText');
msr_gui_struct.CheckBoxPrevLabelHandle=findobj(children_handles,'tag','checkboxPrevLabel');
msr_gui_struct.CheckBoxOverlayPrevLabelHandle=findobj(children_handles,'tag','checkboxOverlayPrev
Label');
msr_gui_struct.CheckBoxSelectMultipleHandle=findobj(children_handles,'tag','checkboxSelectMultipl
e');
msr_gui_struct.CheckBoxSnapToNearestHandle=findobj(children_handles,'tag','checkboxSnapNearest');
if (isempty(previous_lbl))
    %disable show previous label checkbox
    set(msr_gui_struct.CheckBoxPrevLabelHandle,'Enable','off');
    msr_gui_struct.CheckBoxPrevLabelHandle=[];
    %disable overlay previous label checkbox
    set(msr_gui_struct.CheckBoxOverlayPrevLabelHandle,'Enable','off');
    msr_gui_struct.CheckBoxOverlayPrevLabelHandle=[];
end
msr_gui_struct.CheckBoxRawLabelHandle=findobj(children_handles,'tag','checkboxRawLabel');
msr_gui_struct.CheckBoxImageHandle=findobj(children_handles,'tag','checkboxImage');

%display objects image in the objectAxes
msr_gui_struct.ImageHandle=image(objects_rgb,'Parent',objects_axes_handle);
%set the function handle for a mouse click in the objects image
set(msr_gui_struct.ImageHandle,'buttondownfcn','mouseClickInLabel');
set(msr_gui_struct.GuiHandle,'KeyPressFcn','keyPressInManualSegmentationGUI');
%block execution until gui is closed
waitfor(gui_handle);
output_args.LabelMatrix=msr_gui_struct.ObjectsLabel;

%end manualSegmentationReview
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%% manualSegmentationReviewGUI.m %%%%%%%%%%%%%%%%%%%%%%%%%%%
function varargout = manualSegmentationReviewGUI(varargin)
% MANUALSEGMENTATIONREVIEWGUI M-file for manualSegmentationReviewGUI.fig
```

```
%      MANUALSEGMENTATIONREVIEWGUI, by itself, creates a new MANUALSEGMENTATIONREVIEWGUI or
raises the existing
%      singleton*.
%
%      H = MANUALSEGMENTATIONREVIEWGUI returns the handle to a new MANUALSEGMENTATIONREVIEWGUI or
the handle to
%      the existing singleton*.
%
%      MANUALSEGMENTATIONREVIEWGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MANUALSEGMENTATIONREVIEWGUI.M with the given input arguments.
%
%      MANUALSEGMENTATIONREVIEWGUI('Property','Value',...) creates a new
MANUALSEGMENTATIONREVIEWGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before manualSegmentationReviewGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to manualSegmentationReviewGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help manualSegmentationReviewGUI

% Last Modified by GUIDE v2.5 22-Mar-2011 00:08:22

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @manualSegmentationReviewGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @manualSegmentationReviewGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before manualSegmentationReviewGUI is made visible.
function manualSegmentationReviewGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to manualSegmentationReviewGUI (see VARARGIN)

% Choose default command line output for manualSegmentationReviewGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes manualSegmentationReviewGUI wait for user response (see UIRESUME)
% uiwait(handles.ManualResegmentation);


% --- Outputs from this function are returned to the command line.
function varargout = manualSegmentationReviewGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in resegmentBlobButton.
function resegmentBlobButton_Callback(hObject, eventdata, handles)
% hObject    handle to resegmentBlobButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
resegmentBlob('initialize');


% --- Executes on button press in removeBlobButton.
function removeBlobButton_Callback(hObject, eventdata, handles)
% hObject    handle to removeBlobButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
removeBlob(hObject, eventdata, handles);


% --- Executes on button press in selectBlobButton.
function selectBlobButton_Callback(hObject, eventdata, handles)
% hObject    handle to selectBlobButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of selectBlobButton
selectBlobButtonPressed(hObject,eventdata,handles);

% --- Executes on button press in selectObjectButton.
function selectObjectButton_Callback(hObject, eventdata, handles)
% hObject    handle to selectObjectButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of selectObjectButton
selectObjectButtonPressed(hObject,eventdata,handles);


% --- Executes on button press in restoreBlobButton.
function restoreBlobButton_Callback(hObject, eventdata, handles)
% hObject    handle to restoreBlobButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
restoreBlob(hObject, eventdata, handles);


% --- Executes on button press in removeObjectButton.
function removeObjectButton_Callback(hObject, eventdata, handles)
% hObject    handle to removeObjectButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
removeObject(hObject, eventdata, handles);


% --- Executes on button press in undoAllChangesButton.
function undoAllChangesButton_Callback(hObject, eventdata, handles)
% hObject    handle to undoAllChangesButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
undoAllChanges(hObject, eventdata, handles);


% --- Executes on button press in saveChangesButton.
function saveChangesButton_Callback(hObject, eventdata, handles)
% hObject    handle to saveChangesButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
saveChanges(hObject, eventdata, handles);


% --- Executes on button press in joinObjectsButton.
function joinObjectsButton_Callback(hObject, eventdata, handles)
% hObject    handle to joinObjectsButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
joinObjects();


% --- Executes on button press in checkboxPrevLabel.
function checkboxPrevLabel_Callback(hObject, eventdata, handles)
% hObject    handle to checkboxPrevLabel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
showPreviousLabel();


% --- Executes on button press in checkboxOverlayPrevLabel.
function checkboxOverlayPrevLabel_Callback(hObject, eventdata, handles)
% hObject    handle to checkboxOverlayPrevLabel (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
overlayPreviousLabel();

% Hint: get(hObject,'Value') returns toggle state of checkboxOverlayPrevLabel


% --- Executes on button press in checkboxRawLabel.
function checkboxRawLabel_Callback(hObject, eventdata, handles)
% hObject    handle to checkboxRawLabel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
showRawLabel();

% Hint: get(hObject,'Value') returns toggle state of checkboxRawLabel


% --- Executes on button press in checkboxImage.
function checkboxImage_Callback(hObject, eventdata, handles)
% hObject    handle to checkboxImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
showImage();

% Hint: get(hObject,'Value') returns toggle state of checkboxImage


% --- Executes on button press in checkboxSelectMultiple.
function checkboxSelectMultiple_Callback(hObject, eventdata, handles)
% hObject    handle to checkboxSelectMultiple (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
enableMultipleSelection();

% Hint: get(hObject,'Value') returns toggle state of checkboxSelectMultiple




% --- Executes on button press in InvertSelectionButton.
function InvertSelectionButton_Callback(hObject, eventdata, handles)
% hObject    handle to InvertSelectionButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
invertSelection();




% --- Executes on button press in checkboxSnapNearest.
function checkboxSnapNearest_Callback(hObject, eventdata, handles)
% hObject    handle to checkboxSnapNearest (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkboxSnapNearest
snapToNearest();

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% manualTrackingReview.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=manualTrackingReview(input_args)
%Usage
%The manual tracking review module can be used to correct errors in automatic tracking and
%as a visualization module to select subsets of tracks based on speed, motility and ancestry
%parameters. For track correction there are six actions that can be performed: continue a track,
%switch tracks, delete a track, break a track, add a split and remove a split. The GUI displays
%population statistics at the top under the menu bar and individual cell statistics (if a cell is
%selected) in a text box on the right side. Detected cell outlines and cell IDs may be displayed
by
%checking the "Outlines" and/or "Labels" checkboxes.
%
%Track continuation can be used when automatic tracking loses a cell it is tracking and starts a
%new track for that same cell. To continue the pre-existing track with the new track, select the
%pre-existing track. Selecting a track is done by clicking the on the cell body. After selecting
the
%pre-existing track, click on the "Continue Track" button, navigate to a frame where the new
track
%exists and click on the cell body again to select it. The new track is then appended to the pre-
```

```
%existing track. To see the updated track navigate to another frame.
%
%Switching tracks can be used to correct errors resulting from tracks being switched from one
%cell to the other during automatic tracking. To switch the tracks navigate to the frame where
%the error occurs click on the first cell, click on the "Switch tracks" button and finally click
on the
%second cell. To see the updated tracks navigate to another frame.
%
%To erase a track select it then click on the "Delete Track" button.
%
%Breaking a track splits the track in two at the current frame. The newly created track starts at
%the current frame and the old track ends at the frame before that. This can be used to correct
%complex errors by separating a track into smaller segments than using the other actions to fix
%the automatic tracking errors.
%
%Errors in mitotic event detection can be corrected using the "Add Split"
%and "Remove Split"
%buttons. To correct a missed mitotic event select click on the parent cell (the cell with the
older
%track), click on "Add Split" button then click on the second cell that is part of the mitotic
event.
%The older track is broken at the current frame, a new track is created and the ancestry records
%are updated for all three tracks. To remove a spurious mitotic event select the track you want
to
%use to continue the parent track, and then click on the "Remove Split" button. The new track is
%merged with the parent track and the ancestry records are updated for both the parent track and
%the other remaining track.
%
%The visualization part of the GUI is controlled using the "Manage Selection Layers" button.
%A selection layer is a transparency overlaid on the original image that highlights certain cells
%based on a user-defined criterion. The criterion for comparison may be an exact value (such as
%all cells with an area larger than 500 square pixels) or a percentage (cells with an area in the
%top 20%). Any combination of shape, motility and ancestry parameters may be used alone or in
%combination to define a layer. This allows the user to define layers that are either very broad
in
%scope, such as all cells that are larger than average in a movie, or extremely tailored, such as
%selection for small, rounded, fast cells with a specific parent ID. Multiple layers may be
present
%at one time and, due to the use of transparencies and a broad selection of layer colors, cells
%that are part of multiple layers can be detected. The layers are automatically updated as the
%user moves backward or forward through the timelapse sequence, and the resulting images
%themselves may be saved.
%
%To define a selection layer click on the "Manage Selection Layers" button then click the "Add
%Layer" button. Type in the name of the layer, select a layer color from the dropdown box,
%then add a number of conditions. Conditions may be combined using "AND" and "OR" logical
%connectors. A condition consists of a property such as "Area" or "Cell ID" an operation
%("=","<",">" are supported) and a value. The value can be either an absolute number or a
%percentage. Once all the conditions have been set click the "Save Layer" button, then close the
%selection layers GUI to apply the layer. To delete a layer click on
%"Manage Selection Layers"
%then select the layer to be removed and click on the "Remove Layer" button.
%
%Input Structure Members
%AncestryLayout - Matrix describing the order of the columns in the tracks matrix.
%CellsAncestry - Matrix containing cell ancestry records.
%ColorMap - Color map to be used in drawing the cell outlines for each generation. Each
%generation will use the next color in the color map until all colors have been used. Afterwards,
%the colors in the map are recycled.
%FirstFrameIDs - The IDs of tracks starting in the first frame.
%FrameCount - The number of frames to track.
%FrameStep - Read one out of every x frames when reading the image set. Default value is one
%meaning every frame will be read.
%ImageFileBase - The root file name of the images in the sequence. For example, if the image
%names in the time-lapse sequence are "Experiment-0002_Position(8)_t001.jpg","Experiment-
%0002_Position(8)_t002.jpg", etc., the root image file name is "Experiment-0002_Position(8)_t".
%ImgExt - String indicating the image file extension. Usually, ".jpg" or ".tif".
%MaxMissingFrames - This value indicates if tracks not present in the current frame should be
%included in the track subset and if so how many frames away from the current frame a track is
%allowed to be and still be included in the subset.
%NumberFormat - A string indicating the number format of the file name to be used when saving
%the overlayed image.
%SegFileRoot - The root of the data file name containing the segmented objects.
%StartFrame - Integer indicating at which frame the tracking should start.
%SplitCells - The IDs of tracks that are the result of mitosis.
%TimeCol - The index of the time column in the tracks matrix.
%TimeFrame - Time interval between consecutive frames.
%TrackIDCol - The index of the track ID column in the tracks matrix.
%TrackIDs - The IDs of all the tracks.
%Tracks - The matrix containing all the tracks (track IDs and shape parameters for every cell at
%every time point).
```

```
%TracksLayout - Matrix describing the order of the columns in the tracks matrix.
%
%Output Structure Members
%CellsAncestry - Matrix containing corrected cell ancestry records.
%Tracks - The matrix containing all the corrected tracks (track IDs and shape parameters for
%every cell at every time point).

global mtr_gui_struct;
mtr_gui_struct=[];
mtr_gui_struct.TotalErrors=0;

%initialize the gui
field_names=fieldnames(mtr_gui_struct);
gui_handle=findall(0,'Tag','TrackingReview');
if (~isempty(gui_handle))
    close(gui_handle);
end
if (max(strcmp('FigurePosition',field_names)))
    mtr_gui_struct.GuiHandle=manualTrackingReviewGUI('Position',mtr_gui_struct.FigurePosition);
else
    mtr_gui_struct.GuiHandle=manualTrackingReviewGUI;
end
gui_handle=mtr_gui_struct.GuiHandle;
children_handles=get(gui_handle,'children');
mtr_gui_struct.TracksHandle=findobj(children_handles,'tag','tracksAxes');
mtr_gui_struct.SliderHandle=findobj(children_handles,'tag','sliderTracks');
mtr_gui_struct.Tracks=input_args.Tracks.Value;
mtr_gui_struct.CellsAncestry=input_args.CellsAncestry.Value;
mtr_gui_struct.TimeFrame=input_args.TimeFrame.Value;
mtr_gui_struct.TracksLayout=input_args.TracksLayout.Value;
tracks_layout=mtr_gui_struct.TracksLayout;
mtr_gui_struct.TimeCol=tracks_layout.TimeCol;
mtr_gui_struct.TrackIDCol=tracks_layout.TrackIDCol;
mtr_gui_struct.MaxMissingFrames=input_args.MaxMissingFrames.Value;
mtr_gui_struct.FrameStep=input_args.FrameStep.Value;
mtr_gui_struct.ColorMap=input_args.ColorMap.Value;
mtr_gui_struct.AncestryLayout=input_args.AncestryLayout.Value;
mtr_gui_struct.FrameCount=input_args.FrameCount.Value;
mtr_gui_struct.StartFrame=input_args.StartFrame.Value;
mtr_gui_struct.ImageFileBase=input_args.ImageFileBase.Value;
mtr_gui_struct.NumberFormat=input_args.NumberFormat.Value;
mtr_gui_struct.ImgExt=input_args.ImgExt.Value;
mtr_gui_struct.SegFileRoot=input_args.SegFileRoot.Value;
[cell_speeds
cell_sq_disps]=getMotilityParams(mtr_gui_struct.Tracks,mtr_gui_struct.TracksLayout,...
    mtr_gui_struct.CellsAncestry,mtr_gui_struct.AncestryLayout,mtr_gui_struct.TimeFrame);
mtr_gui_struct.CellSpeeds=cell_speeds;
mtr_gui_struct.CellSquareDisplacements=cell_sq_disps;
mtr_gui_struct.FrameMSDs=getFrameMSDs(cell_sq_disps,mtr_gui_struct.TimeFrame);
mtr_gui_struct.StatusTextHandle=findobj(children_handles,'tag','statusText');
mtr_gui_struct.EditStatus1Handle=findobj(children_handles,'tag','editStatus1');
mtr_gui_struct.EditStatus2Handle=findobj(children_handles,'tag','editStatus2');
mtr_gui_struct.EditStatus3Handle=findobj(children_handles,'tag','editStatus3');
mtr_gui_struct.EditCellStatusHandle=findobj(children_handles,'tag','editStatusCell');
mtr_gui_struct.CheckBoxLabelsHandle=findobj(children_handles,'tag','checkboxLabels');
mtr_gui_struct.CheckBoxOutlinesHandle=findobj(children_handles,'tag','checkboxShowOutlines');
mtr_gui_struct.ButtonContinueTrackHandle=findobj(children_handles,'tag','buttonContinueTrack');
mtr_gui_struct.ButtonRemoveSplitHandle=findobj(children_handles,'tag','buttonRemoveSplit');
mtr_gui_struct.ButtonAddSplitHandle=findobj(children_handles,'tag','buttonAddSplit');
mtr_gui_struct.AveragesTextHandle=findobj(children_handles,'tag','textAverages');
cur_frame=mtr_gui_struct.StartFrame;
mtr_gui_struct.CurFrame=cur_frame;
mtr_gui_struct.SelectedCellID=0;
mtr_gui_struct.SelectedCellLabelID=0;
mtr_gui_struct.ShowLabels=get(mtr_gui_struct.CheckBoxLabelsHandle,'Value');
mtr_gui_struct.ShowOutlines=get(mtr_gui_struct.CheckBoxOutlinesHandle,'Value');
mtr_gui_struct.ContinueTrack=false;
mtr_gui_struct.SplitTrack=false;
mtr_gui_struct.SwitchTrack=false;
mtr_gui_struct.SelectionLayers={};

set(mtr_gui_struct.SliderHandle,'Min',cur_frame);
set(mtr_gui_struct.SliderHandle,'Max',mtr_gui_struct.FrameCount+cur_frame);
set(mtr_gui_struct.SliderHandle,'Value',cur_frame);
slider_step_size=1.0/double(mtr_gui_struct.FrameCount);
set(mtr_gui_struct.SliderHandle,'SliderStep',[slider_step_size min([10*slider_step_size 1])]);
%turn off the axes
set(mtr_gui_struct.GuiHandle,'DefaultAxesVisible','off');
set(mtr_gui_struct.StatusTextHandle,'String', ' Frame: 1 Mitotic Events Detected: 0');
set(mtr_gui_struct.EditStatus1Handle,'String', 'New Cells From Split:');
set(mtr_gui_struct.EditStatus2Handle,'String', 'Other New Cells In Frame:');
```

```
mtr_gui_struct.AveragesText=displayCellAverages();
displayFrameMSD();
updateTrackImage(cur_frame,mtr_gui_struct.ShowLabels,mtr_gui_struct.ShowOutlines);
mtr_gui_struct.CurCentroids=getApproximateCentroids(mtr_gui_struct.CellsLabel);
%block execution until gui is closed
waitfor(gui_handle);
output_args.Tracks=mtr_gui_struct.Tracks;
output_args.CellsAncestry=mtr_gui_struct.CellsAncestry;

%end manualTrackingReview
end

function frame_msds=getFrameMSDs(square_disps,time_frame)

max_time_frame=max(square_disps(:,3));
nr_frames=max_time_frame/time_frame+1;
frame_msds=zeros(nr_frames,1);
for i=2:nr_frames
    cur_time=(i-1)*time_frame;
    frame_msds(i)=mean(square_disps(square_disps(:,3)==cur_time,2));
end

%getFrameMSDs
end

function [cell_speeds
square_disps]=getMotilityParams(tracks,tracks_layout,ancestry_records,ancestry_layout,time_frame)

tracks_nr=max(ancestry_records(:,ancestry_layout.TrackIDCol));
cell_speeds=zeros(size(tracks,1),3);
square_disps=cell_speeds;
for i=1:tracks_nr
    cur_track_idx=tracks(:,tracks_layout.TrackIDCol)==i;
    track_centroids=tracks(cur_track_idx,tracks_layout.Centroid1Col:tracks_layout.Centroid2Col);
    if isempty(track_centroids)
        continue;
    end
    cur_dist=hypot(track_centroids(1:(end-1),1)-track_centroids(2:end,1),track_centroids(1:(end-
1),2)-track_centroids(2:end,2));
    total_dist=cumsum(cur_dist);
    cell_speeds(cur_track_idx,2)=[0; cur_dist]./time_frame;
    cell_speeds(cur_track_idx,1)=i;
    cell_speeds(cur_track_idx,3)=tracks(cur_track_idx,tracks_layout.TimeCol);
    square_disps(cur_track_idx,2)=(track_centroids(:,1)-
track_centroids(1,1)).^2+(track_centroids(:,2)-track_centroids(1,2)).^2;
    square_disps(cur_track_idx,1)=i;
    square_disps(cur_track_idx,3)=tracks(cur_track_idx,tracks_layout.TimeCol);
end

%end getMotilityParams
end

function averages_text=displayCellAverages()
global mtr_gui_struct;
str_fmt='%1.2f';

tracks=mtr_gui_struct.Tracks;
tracks_layout=mtr_gui_struct.TracksLayout;
cell_speeds=mtr_gui_struct.CellSpeeds;
cell_areas=tracks(:,tracks_layout.AreaCol);
mean_area=mean(cell_areas);
averages_text=['Cell Averages: Area ' num2str(mean_area,str_fmt)];
cell_ecc=tracks(:,tracks_layout.EccCol);
mean_ecc=mean(cell_ecc);
averages_text=[averages_text ' Eccentricity ' num2str(mean_ecc,str_fmt)];
cell_per=tracks(:,tracks_layout.PerCol);
mean_per=mean(cell_per);
averages_text=[averages_text ' Perimeter ' num2str(mean_per,str_fmt)];
cell_sol=tracks(:,tracks_layout.SolCol);
mean_sol=mean(cell_sol);
averages_text=[averages_text ' Solidity ' num2str(mean_sol,str_fmt)];
mean_speed=mean(cell_speeds(cell_speeds(:,2)>0,2));
averages_text=[averages_text ' Speed ' num2str(mean_speed,str_fmt)];
set(mtr_gui_struct.AveragesTextHandle,'String',averages_text);

%end displayCellAverages
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% manualTrackingReviewGUI.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function varargout = manualTrackingReviewGUI(varargin)
% MANUALTRACKINGREVIEWGUI M-file for manualTrackingReviewGUI.fig
%      MANUALTRACKINGREVIEWGUI, by itself, creates a new MANUALTRACKINGREVIEWGUI or raises the
existing
%      singleton*.
%
%      H = MANUALTRACKINGREVIEWGUI returns the handle to a new MANUALTRACKINGREVIEWGUI or the
handle to
%      the existing singleton*.
%
%      MANUALTRACKINGREVIEWGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MANUALTRACKINGREVIEWGUI.M with the given input arguments.
%
%      MANUALTRACKINGREVIEWGUI('Property','Value',...) creates a new MANUALTRACKINGREVIEWGUI or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before manualTrackingReviewGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to manualTrackingReviewGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help manualTrackingReviewGUI

% Last Modified by GUIDE v2.5 12-Aug-2010 14:51:50

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @manualTrackingReviewGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @manualTrackingReviewGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before manualTrackingReviewGUI is made visible.
function manualTrackingReviewGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to manualTrackingReviewGUI (see VARARGIN)

% Choose default command line output for manualTrackingReviewGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes manualTrackingReviewGUI wait for user response (see UIRESUME)
% uiwait(handles.TrackingReview);


% --- Outputs from this function are returned to the command line.
function varargout = manualTrackingReviewGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on slider movement.
```

```
function sliderTracks_Callback(hObject, eventdata, handles)
% hObject    handle to sliderTracks (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
sliderTracksEvent();


% --- Executes during object creation, after setting all properties.
function sliderTracks_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sliderTracks (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on button press in buttonContinueTrack.
function buttonContinueTrack_Callback(hObject, eventdata, handles)
% hObject    handle to buttonContinueTrack (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
continueTrack();



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editStatus1_Callback(hObject, eventdata, handles)
% hObject    handle to editStatus1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editStatus1 as text
%        str2double(get(hObject,'String')) returns contents of editStatus1 as a double


% --- Executes during object creation, after setting all properties.
function editStatus1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editStatus1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editStatus2_Callback(hObject, eventdata, handles)
% hObject    handle to editStatus2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of editStatus2 as text
%        str2double(get(hObject,'String')) returns contents of editStatus2 as a double


% --- Executes during object creation, after setting all properties.
function editStatus2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editStatus2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkboxLabels.
function checkboxLabels_Callback(hObject, eventdata, handles)
% hObject    handle to checkboxLabels (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkboxLabels
checkBoxLabelsEvent();



function editStatus3_Callback(hObject, eventdata, handles)
% hObject    handle to editStatus3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editStatus3 as text
%        str2double(get(hObject,'String')) returns contents of editStatus3 as a double


% --- Executes during object creation, after setting all properties.
function editStatus3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editStatus3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editStatusCell_Callback(hObject, eventdata, handles)
% hObject    handle to editStatusCell (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editStatusCell as text
%        str2double(get(hObject,'String')) returns contents of editStatusCell as a double


% --- Executes during object creation, after setting all properties.
function editStatusCell_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editStatusCell (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in buttonAddSplit.
function buttonAddSplit_Callback(hObject, eventdata, handles)
% hObject    handle to buttonAddSplit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
addSplit();
```

```
% --- Executes on button press in buttonRemoveSplit.
function buttonRemoveSplit_Callback(hObject, eventdata, handles)
% hObject    handle to buttonRemoveSplit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
removeSplit();


% --- Executes on button press in buttonManageSelectionLayers.
function buttonManageSelectionLayers_Callback(hObject, eventdata, handles)
% hObject    handle to buttonManageSelectionLayers (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
manageSelectionLayers();

% --- Executes on button press in buttonRemoveSelectionLayer.
function buttonRemoveSelectionLayer_Callback(hObject, eventdata, handles)
% hObject    handle to buttonRemoveSelectionLayer (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in checkboxShowOutlines.
function checkboxShowOutlines_Callback(hObject, eventdata, handles)
% hObject    handle to checkboxShowOutlines (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkboxShowOutlines
checkBoxOutlinesEvent();



% --- Executes on button press in buttonSwitchTracks.
function buttonSwitchTracks_Callback(hObject, eventdata, handles)
% hObject    handle to buttonSwitchTracks (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
switchTracks();


% --------------------------------------------------------------------
function fileMenu_Callback(hObject, eventdata, handles)
% hObject    handle to fileMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function saveImage_Callback(hObject, eventdata, handles)
% hObject    handle to saveImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
saveTrackingImage();


% --- Executes on button press in buttonDeleteTrack.
function buttonDeleteTrack_Callback(hObject, eventdata, handles)
% hObject    handle to buttonDeleteTrack (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
deleteTrack();


% --- Executes on button press in buttonBreakTrack.
function buttonBreakTrack_Callback(hObject, eventdata, handles)
% hObject    handle to buttonBreakTrack (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
breakTrack();
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% mergeTracks.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=mergeTracks(input_args)
% Usage
% This module is used to merge a list of track pairs. This module is not used to determine
whether a set of tracks should be merged. Other modules are provided for that purpose, such as
detectMergeCandidatesUsingDistance.
```

```
% Input Structure Members
% FrameCount - The number of frames that are being processed.
% FrameStep - How many frames to skip when reading frames. Setting this value to one will cause
all the frames to be read.
% NumberFormat - A string indicating the number format of the file name to be used when saving
the overlayed image.
% SegFileRoot - The root of the data file name containing the segmented objects.
% StartFrame - The first frame in the processed set.
% TimeFrame - Time interval between consecutive frames.
% Tracks - The current set of tracks.
% TracksLayout - Matrix describing the order of the columns in the tracks matrix.
% TracksToBeMerged - The matrix of track IDs to be merged. Primary IDs (the IDs which will remain
after the merge) are listed in the first column and secondary IDs (the IDs which will be removed
after the merge) are listed in the second column of the matrix.
% Output Structure Members
% Tracks - The new track set with the results from the merge incorporated.

tracks=input_args.Tracks.Value;
tracks_to_be_merged=input_args.TracksToBeMerged.Value;
if isempty(tracks_to_be_merged)
    output_args.Tracks=tracks;
    return;
end

framecount=input_args.FrameCount.Value;
startframe=input_args.StartFrame.Value;
timeframe=input_args.TimeFrame.Value;
seg_file_root=input_args.SegFileRoot.Value;
frame_step=input_args.FrameStep.Value;
number_fmt=input_args.NumberFormat.Value;
tracks_layout=input_args.TracksLayout.Value;
timeCol=tracks_layout.TimeCol;
trackIDCol=tracks_layout.TrackIDCol;
centroid1Col=tracks_layout.Centroid1Col;
centroid2Col=tracks_layout.Centroid2Col;
areaCol=tracks_layout.AreaCol;
blobIDCol=tracks_layout.BlobIDCol;

%the primary ids will remain
primary_ids=tracks_to_be_merged(:,1);
%secondary ids will be gone after merging
secondary_ids=tracks_to_be_merged(:,2);

%replace those ids with their primaries in the primary list. this relies on
%the fact that those ids will be replaced by their primaries in the tracks
%list before we get to the ids that they are primary too. this is true if
%we detect tracks to be merged in chronological order. can't do it using
%ismember as a secondary can be a primary to another secondary. have to do
%it one-by-one in serial fashion
for i=1:length(primary_ids)
    cur_primary_id=primary_ids(i);
    %is this primary id secondary to some other primary id
    primary_to_this_primary_idx=find(secondary_ids==cur_primary_id,1);
    if (isempty(primary_to_this_primary_idx))
        continue;
    end
    %replace all records of this primary id in the primary_ids list
    primary_ids(primary_ids==cur_primary_id)=primary_ids(primary_to_this_primary_idx);
end

for i=1:frame_step:frame_step*framecount
    curframe=startframe+i-1
    cur_time=(curframe-1)*timeframe;
    cur_tracks_idx=tracks(:,timeCol)==cur_time;
    cur_tracks=tracks(cur_tracks_idx,:);
    cur_ids=cur_tracks(:,trackIDCol);
    primary_idx=ismember(primary_ids,cur_ids);
    cur_primary_ids=primary_ids(primary_idx);
    secondary_idx=ismember(secondary_ids,cur_ids);
    cur_secondary_ids=secondary_ids(secondary_idx);
    bEmptySecondary=isempty(cur_secondary_ids);
    if (bEmptySecondary)
        %nothing to merge in this frame
        continue;
    end
    %something to merge or replace so load the cells_lbl
    mat_filename=[seg_file_root num2str(curframe,number_fmt)];
    file_struct=load(mat_filename);
    cells_lbl=file_struct.cells_lbl;
    clear('file_struct');
```

```
    while (~isempty(cur_secondary_ids))
        secondary_id=cur_secondary_ids(1);
        %find a primary_id for this secondary_id
        primary_id_candidates=primary_ids(secondary_ids==secondary_id);
        for k=1:length(primary_id_candidates)
            primary_id=primary_id_candidates(k);
            primary_id_test=cur_primary_ids(cur_primary_ids==primary_id);
            if (~isempty(primary_id_test))
                break;
            end
        end
        if isempty(primary_id_test)
            %assign the properties of the secondary id to the primary id
            track_idx=(tracks(:,trackIDCol)==secondary_id&tracks(:,timeCol)==cur_time);
            tracks(track_idx,trackIDCol)=primary_id;
            %add the primary id to the list of current primary ids
            cur_primary_ids=[cur_primary_ids; primary_id];
        else
            %combine the properties of the secondary id with the primary id
            %get both blobs
            %update the cur_tracks list since the tracks have been modified
            cur_tracks_idx=tracks(:,timeCol)==cur_time;
            cur_tracks=tracks(cur_tracks_idx,:);
            primary_idx=cur_tracks(:,trackIDCol)==primary_id;
            primary_centroid=cur_tracks(primary_idx,centroid1Col:centroid2Col);
            %the merged tracks will keep the blob id of the primary object
            primary_blob_id=cur_tracks(primary_idx,blobIDCol);
            primary_lbl_id=getLabelId(cells_lbl, primary_centroid);

secondary_centroid=cur_tracks(cur_tracks(:,trackIDCol)==secondary_id,centroid1Col:centroid2Col);
            secondary_lbl_id=getLabelId(cells_lbl, secondary_centroid);
            cur_blobs=(cells_lbl==primary_lbl_id|cells_lbl==secondary_lbl_id);
            [cur_blobs_1 cur_blobs_2] =find(cur_blobs);
            %get the bounding box for the blobs
            max_1=max(cur_blobs_1);
            min_1=min(cur_blobs_1);
            max_2=max(cur_blobs_2);
            min_2=min(cur_blobs_2);
            % crop to the extent of the box so we run fast on a small
            % image
            cur_blobs=cur_blobs(min_1:max_1,min_2:max_2);
            cur_blobs_lbl=bwlabeln(cur_blobs);
            nr_blobs=max(cur_blobs_lbl(:));
            %determine if the two blobs are touching
            if (nr_blobs>1)
                %blobs are not touching - i need to connect them
                cur_blobs=connectBlobs(cur_blobs, nr_blobs);
                cur_blobs_lbl=bwlabeln(cur_blobs);
                %get the new coordinates
                [cur_blobs_1 cur_blobs_2]=find(cur_blobs);
                %update them to the main box
                cur_blobs_1=cur_blobs_1+min_1;
                cur_blobs_2=cur_blobs_2+min_2;

            end
            %calculate the new region props
            get_shape_params_args.LabelMatrix.Value=cur_blobs_lbl;
            shape_params_output=getShapeParams(get_shape_params_args);
            new_shape_params=shape_params_output.ShapeParameters;
            new_centroid=shape_params_output.Centroids;
            %update the new centroids with respect to the uncropped image
            new_centroid=new_centroid+[min_1 min_2];
            %update the tracks matrix
            %remove the secondary id
            track_idx=(tracks(:,trackIDCol)==secondary_id&tracks(:,timeCol)==cur_time);
            tracks(track_idx,:)=[];
            %update the primary id with the new params
            track_idx=(tracks(:,trackIDCol)==primary_id&tracks(:,timeCol)==cur_time);
            tracks(track_idx,centroid1Col:centroid2Col)=new_centroid;
            %this assumes the shape params start with the area column
            assert(areaCol==5);
            tracks(track_idx,areaCol:end)=new_shape_params;
            tracks(track_idx,blobIDCol)=primary_blob_id;

            %update the cells_lbl
            blob_lin_idx=sub2ind(size(cells_lbl), cur_blobs_1, cur_blobs_2);
            cells_lbl(blob_lin_idx)=primary_lbl_id;
        end
        cur_secondary_ids(cur_secondary_ids==secondary_id)=[];
    end
    %save the new cells_lbl
```

```
    cells_lbl=makeContinuousLabelMatrix(cells_lbl);
    save([seg_file_root num2str(curframe,number_fmt)],'cells_lbl');
end

output_args.Tracks=tracks;

%end mergeTracks
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% mkdir_Wrapper.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=mkdir_Wrapper(input_args)
%simple wrapper for the MATLAB mkdir function
%Input Structure Members
%DirectoryName - The path where the directory will be created.
%Output Structure Members
%None

mkdir(input_args.DirectoryName.Value);
output_args=[];

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% moduleParentGUI.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function varargout = moduleParentGUI(varargin)
% MODULEPARENTGUI M-file for moduleParentGUI.fig
%      MODULEPARENTGUI, by itself, creates a new MODULEPARENTGUI or raises the existing
%      singleton*.
%
%      H = MODULEPARENTGUI returns the handle to a new MODULEPARENTGUI or the handle to
%      the existing singleton*.
%
%      MODULEPARENTGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MODULEPARENTGUI.M with the given input arguments.
%
%      MODULEPARENTGUI('Property','Value',...) creates a new MODULEPARENTGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before moduleParentGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to moduleParentGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help moduleParentGUI

% Last Modified by GUIDE v2.5 13-Aug-2011 16:15:16

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @moduleParentGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @moduleParentGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before moduleParentGUI is made visible.
function moduleParentGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to moduleParentGUI (see VARARGIN)

% Choose default command line output for moduleParentGUI
handles.output = hObject;
mt_idx=find(strcmp(varargin, 'ParentsList'))+1;
handles.ParentsList=varargin{mt_idx};
set(handles.popupmenuParent,'String',handles.ParentsList);
handles.OK=true;
handles.ParentIdx=1;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes moduleParentGUI wait for user response (see UIRESUME)
uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = moduleParentGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1}=handles.OK;
varargout{2}=handles.ParentIdx;
varargout{3}=get(handles.editInstance,'String');

delete(hObject);



function editInstance_Callback(hObject, eventdata, handles)
% hObject    handle to editInstance (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editInstance as text
%        str2double(get(hObject,'String')) returns contents of editInstance as a double


% --- Executes during object creation, after setting all properties.
function editInstance_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editInstance (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbuttonOK.
function pushbuttonOK_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonOK (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
uiresume(handles.figure1);


% --- Executes on button press in pushbuttonCancel.
function pushbuttonCancel_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonCancel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.OK=false;
guidata(handles.figure1,handles);
uiresume(handles.figure1);


% --- Executes on selection change in popupmenuParent.
function popupmenuParent_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenuParent (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenuParent contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenuParent
```

```
handles.ParentIdx=get(hObject,'Value');
guidata(handles.figure1,handles);


% --- Executes during object creation, after setting all properties.
function popupmenuParent_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenuParent (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: delete(hObject) closes the figure
uiresume(hObject);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% mouseClickInLabel.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function []=mouseClickInLabel()
%helper function for manual segmentation review. detect mouse clicks and
%perform various actions
global msr_gui_struct;

axes_handle=msr_gui_struct.AxesHandle;
original_axes_units=get(axes_handle,'Units');
set(axes_handle,'Units','Pixels');
click_point = get(axes_handle,'CurrentPoint');
set(axes_handle,'Units',original_axes_units);

switch (msr_gui_struct.CurrentAction)
    case 'SelectBlob'
        selectBlob(click_point);
    case 'SelectObject'
        selectObject(click_point);
    case 'ResegmentBlob'
        addBlobCentroid(click_point);
    case 'RestoreBlob'
        selectBlobAndRestore(click_point);
    otherwise
        return;
end

%end mouseClickInLabel
end



function selectBlob(click_point)
global msr_gui_struct;

blobs_lbl=msr_gui_struct.BlobsLabel;
blob_id=blobs_lbl(round(click_point(1,2)),round(click_point(1,1)));
if ((blob_id==0) && msr_gui_struct.SnapToNearest)
   lbl_sz=size(blobs_lbl);
   nhood_sz=20;
   point11=max(1,(round(click_point(1,1)-nhood_sz)));
   point12=min(lbl_sz(2),(round(click_point(1,1)+nhood_sz)));
   point21=max(1,(round(click_point(1,2)-nhood_sz)));
   point22=min(lbl_sz(1),(round(click_point(1,2)+nhood_sz)));
   nearby_frame=blobs_lbl(point21:point22,point11:point12);
   nearby_centroids=getApproximateCentroids(nearby_frame);
   nearby_ids=unique(nearby_frame);
   nearby_ids(1)=[];
   dist_1=nearby_centroids(nearby_ids,1)-click_point(1,1);
   dist_2=nearby_centroids(nearby_ids,2)-click_point(1,2);
   nearby_distances=sqrt(dist_1.^2+dist_2.^2);
   [dummy min_idx]=min(nearby_distances);
   click_point=nearby_centroids(nearby_ids(min_idx),:);
```

```
   blob_id=nearby_frame(round(click_point(1)),round(click_point(2)));
end
selectBlobByID(blob_id);
original_blobs_lbl=msr_gui_struct.OriginalBlobsLabel;
msr_gui_struct.OriginalBlobID=original_blobs_lbl(round(click_point(1,2)),round(click_point(1,1)))
;

%end selectBlob
end

function selectObject(click_point)
global msr_gui_struct;

objects_lbl=msr_gui_struct.ObjectsLabel;
obj_id=objects_lbl(round(click_point(1,2)),round(click_point(1,1)));

if ((obj_id==0) && msr_gui_struct.SnapToNearest)
   lbl_sz=size(objects_lbl);
   nhood_sz=20;
   point11=max(1,(round(click_point(1,1)-nhood_sz)));
   point12=min(lbl_sz(2),(round(click_point(1,1)+nhood_sz)));
   point21=max(1,(round(click_point(1,2)-nhood_sz)));
   point22=min(lbl_sz(1),(round(click_point(1,2)+nhood_sz)));
   nearby_frame=objects_lbl(point21:point22,point11:point12);
   nearby_centroids=getApproximateCentroids(nearby_frame);
   nearby_ids=unique(nearby_frame);
   nearby_ids(1)=[];
   dist_1=nearby_centroids(nearby_ids,1)-click_point(1,1);
   dist_2=nearby_centroids(nearby_ids,2)-click_point(1,2);
   nearby_distances=sqrt(dist_1.^2+dist_2.^2);
   [dummy min_idx]=min(nearby_distances);
   click_point=nearby_centroids(nearby_ids(min_idx),:);
   obj_id=nearby_frame(round(click_point(1)),round(click_point(2)));
end
selectObjectByID(obj_id);

%end selectObject
end

function addBlobCentroid(click_point)
global msr_gui_struct;

blobs_lbl=msr_gui_struct.BlobsLabel;
blob_id=blobs_lbl(round(click_point(1,2)),round(click_point(1,1)));
if (blob_id~=msr_gui_struct.SelectedBlobID)
    warnDlg('You have not clicked on the selected blob!');
    return;
end
msr_gui_struct.SegmentationTrainingPoints=[msr_gui_struct.SegmentationTrainingPoints;[click_point
(1,2),click_point(1,1)]];
msr_gui_struct.SegmentationGroups=[msr_gui_struct.SegmentationGroups;
msr_gui_struct.CurrentResegmentationIndex];

%end addBlobCentroid
end

function selectBlobAndRestore(click_point)
global msr_gui_struct;

original_blobs_lbl=msr_gui_struct.OriginalBlobsLabel;
original_objects_lbl=msr_gui_struct.OriginalObjectsLabel;
blobs_lbl=msr_gui_struct.BlobsLabel;
original_blob_id=original_blobs_lbl(round(click_point(1,2)),round(click_point(1,1)));
blob_id=blobs_lbl(round(click_point(1,2)),round(click_point(1,1)));
image_handle=msr_gui_struct.ImageHandle;
objects_lbl=msr_gui_struct.ObjectsLabel;
if (original_blob_id==0)
    warnDlg('You clicked on the background!');
    image_data=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
    set(image_handle,'CData',image_data);
    msr_gui_struct.SelectedBlobID=[];
    msr_gui_struct.ObjectsLabel=objects_lbl;
    msr_gui_struct.CurrentAction='SelectBlob';
    return;
end
if (blob_id~=0)
    warnDlg('The blob you clicked on exists. You need to delete a blob before you can restore
it!');
    image_data=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
    set(image_handle,'CData',image_data);
    msr_gui_struct.SelectedBlobID=[];
```

```
    msr_gui_struct.ObjectsLabel=objects_lbl;
    msr_gui_struct.CurrentAction='SelectBlob';
    return;
end
%remove the blob errors
error_blob_ids=msr_gui_struct.ErrorBlobIDs;
blob_errors_idx=(error_blob_ids==original_blob_id);
other_errors_nr=sum(blob_errors_idx);
if (other_errors_nr)
    msr_gui_struct.TotalErrors=msr_gui_struct.TotalErrors-other_errors_nr;
    error_types=msr_gui_struct.ErrorTypes;
    error_types(blob_errors_idx)=[];
    msr_gui_struct.ErrorTypes=error_types;
    error_blob_ids(blob_errors_idx)=[];
    msr_gui_struct.ErrorBlobIDs=error_blob_ids;
end

%restore the blob
original_blob=(original_blobs_lbl==original_blob_id);
objects_in_blob=original_objects_lbl(original_blob);
original_ids=unique(objects_in_blob);
max_id=max(objects_lbl(:));
new_ids=max_id+(1:length(original_ids));
%create an array to substitute the original ids with the new ids - make it
%sparse since we're not going to use most values
subs_array=sparse(max(original_ids),1);
subs_array(original_ids)=new_ids;
objects_lbl(original_blob)=subs_array(objects_in_blob);
image_data=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
set(image_handle,'CData',image_data);
msr_gui_struct.SelectedBlobID=[];
msr_gui_struct.ObjectsLabel=objects_lbl;
msr_gui_struct.BlobsLabel=bwlabeln(objects_lbl);
msr_gui_struct.CurrentAction='SelectBlob';

%end selectBlobAndRestore
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% mouseClickInTrackingFrame.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function mouseClickInTrackingFrame()
%helper function for manual tracking review. detect mouse clicks and
%perform various functions
global mtr_gui_struct;

axes_handle=mtr_gui_struct.TracksHandle;
original_axes_units=get(axes_handle,'Units');
set(axes_handle,'Units','Pixels');
click_point = get(axes_handle,'CurrentPoint');
set(axes_handle,'Units',original_axes_units);
cells_lbl=mtr_gui_struct.CellsLabel;
cur_cell_lbl_id=cells_lbl(round(click_point(1,2)),round(click_point(1,1)));
if ((cur_cell_lbl_id==0)||(mtr_gui_struct.SelectedCellLabelID==cur_cell_lbl_id))
    hold off;
    mtr_gui_struct.SelectedCellID=0;
    mtr_gui_struct.SelectedCellLabelID=0;
%
mtr_gui_struct.ImageHandle=image(mtr_gui_struct.ImageData,'Parent',mtr_gui_struct.TracksHandle);
    %set the function handle for a mouse click in the objects image
    set(mtr_gui_struct.ImageHandle,'buttondownfcn','mouseClickInTrackingFrame');
    set(mtr_gui_struct.ButtonContinueTrackHandle,'Enable','off');
    set(mtr_gui_struct.ButtonRemoveSplitHandle,'Enable','off');
    set(mtr_gui_struct.ButtonAddSplitHandle,'Enable','off');
else
    selectCell(cur_cell_lbl_id);
    mtr_gui_struct.SelectedCellLabelID=cur_cell_lbl_id;
    cell_id=getCellIDFromLabelID();
    mtr_gui_struct.SelectedCellID=cell_id;
    cell_ancestries=mtr_gui_struct.CellsAncestry;
    ancestry_layout=mtr_gui_struct.AncestryLayout;
    ancestry_record=cell_ancestries(cell_ancestries(:,ancestry_layout.TrackIDCol)==cell_id,:);

mtr_gui_struct.SelectedCellStart=(ancestry_record(ancestry_layout.StartTimeCol)./mtr_gui_struct.T
imeFrame)+1;

mtr_gui_struct.SelectedCellEnd=(ancestry_record(ancestry_layout.StopTimeCol)./mtr_gui_struct.Time
Frame)+1;
    mtr_gui_struct.CurrentAncestryRecord=ancestry_record;
    updateCellStatus();
```

```
    if (mtr_gui_struct.ContinueTrack)
        continue_id=mtr_gui_struct.TrackToContinueID;
        if (cell_id==continue_id)
            mtr_gui_struct.TrackToContinueID=[];
            mtr_gui_struct.TrackToContinueRecord=[];
            mtr_gui_struct.TrackToContinueAncestry=[];
            mtr_gui_struct.ContinueTrack=false;
            warndlg('You cannot continue the track using this track!');
        end
        completeContinueTrack();
    elseif (mtr_gui_struct.SplitTrack)
        split_id=mtr_gui_struct.TrackToSplitID;
        if (cell_id==split_id)
            mtr_gui_struct.TrackToSplitID=[];
            mtr_gui_struct.TrackToSplitRecord=[];
            mtr_gui_struct.TrackToSplitAncestry=[];
            mtr_gui_struct.SplitTrack=false;
            warndlg('You cannot complete the split with this track!');
        end
        completeSplitTrack();
    elseif (mtr_gui_struct.SwitchTrack)
        switch_id=mtr_gui_struct.SwitchTrackID;
        if (cell_id==switch_id)
            mtr_gui_struct.SwitchTrackID=[];
            mtr_gui_struct.SwitchTrackRecord=[];
            mtr_gui_struct.SwitchTrackAncestry=[];
            mtr_gui_struct.SwitchTrack=false;
            warndlg('You cannot complete the switch with this track!');
        end
        completeSwitchTrack();
    else
        set(mtr_gui_struct.ButtonContinueTrackHandle,'Enable','on');
        set(mtr_gui_struct.ButtonRemoveSplitHandle,'Enable','on');
        set(mtr_gui_struct.ButtonAddSplitHandle,'Enable','on');
    end
end


%end mouseClickInTrackingFrame
end

function completeSwitchTrack()
global mtr_gui_struct;

track1_id=mtr_gui_struct.SelectedCellID;
track2_id=mtr_gui_struct.SwitchTrackID;
cur_frame=mtr_gui_struct.CurFrame;
cur_time=(cur_frame-1).*mtr_gui_struct.TimeFrame;
tracks=mtr_gui_struct.Tracks;
tracks_layout=mtr_gui_struct.TracksLayout;
track_1_idx=(tracks(:,tracks_layout.TrackIDCol)==track1_id)&(tracks(:,tracks_layout.TimeCol)>=cur
_time);
track_2_idx=(tracks(:,tracks_layout.TrackIDCol)==track2_id)&(tracks(:,tracks_layout.TimeCol)>=cur
_time);
tracks(track_1_idx,tracks_layout.TrackIDCol)=track2_id;
tracks(track_2_idx,tracks_layout.TrackIDCol)=track1_id;
mtr_gui_struct.Tracks=tracks;
track1_ancestry_record=mtr_gui_struct.CurrentAncestryRecord;
track2_ancestry_record=mtr_gui_struct.SwitchTrackAncestry;
ancestry_layout=mtr_gui_struct.AncestryLayout;
ancestry_records=mtr_gui_struct.CellsAncestry;
track_1_idx=ancestry_records(:,ancestry_layout.TrackIDCol)==track1_id;
ancestry_records(track_1_idx,ancestry_layout.StopTimeCol)=...
    track2_ancestry_record(ancestry_layout.StopTimeCol);
track_2_idx=ancestry_records(:,ancestry_layout.TrackIDCol)==track2_id;
ancestry_records(track_2_idx,ancestry_layout.StopTimeCol)=...
    track1_ancestry_record(ancestry_layout.StopTimeCol);
affected_times_idx=(ancestry_records(:,ancestry_layout.StartTimeCol)>=cur_time);
track_1_children_idx=(ancestry_records(:,ancestry_layout.ParentIDCol)==track1_id)&affected_times_
idx;
track_2_children_idx=(ancestry_records(:,ancestry_layout.ParentIDCol)==track2_id)&affected_times_
idx;
ancestry_records(track_1_children_idx,ancestry_layout.ParentIDCol)=track2_id;
ancestry_records(track_2_children_idx,ancestry_layout.ParentIDCol)=track1_id;
mtr_gui_struct.CellsAncestry=ancestry_records;
mtr_gui_struct.SelectedCellID=track2_id;
mtr_gui_struct.SwitchTrackID=[];
mtr_gui_struct.SwitchTrackRecord=[];
mtr_gui_struct.SwitchTrackAncestry=[];
mtr_gui_struct.SwitchTrack=false;
updateCellStatus();
```

```
%end completeSwitchTrack
end

function completeSplitTrack()
global mtr_gui_struct;

ancestry_layout=mtr_gui_struct.AncestryLayout;
new_cell_ancestry=mtr_gui_struct.CurrentAncestryRecord;
new_cell_start_frame=(new_cell_ancestry(ancestry_layout.StartTimeCol)./mtr_gui_struct.TimeFrame)+
1;
if (new_cell_start_frame~=mtr_gui_struct.CurFrame)
    mtr_gui_struct.TrackToSplitID=[];
    mtr_gui_struct.TrackToSplitRecord=[];
    mtr_gui_struct.TrackToSplitAncestry=[];
    mtr_gui_struct.SplitTrack=false;
    warndlg('This is not a new track in this frame!');
end
parent_id=new_cell_ancestry(ancestry_layout.ParentIDCol);
if (parent_id~=0)
    mtr_gui_struct.TrackToSplitID=[];
    mtr_gui_struct.TrackToSplitRecord=[];
    mtr_gui_struct.TrackToSplitAncestry=[];
    mtr_gui_struct.SplitTrack=false;
    warndlg('This track is the result of a split. You need to remove that split first!');
end
parent_ancestry=mtr_gui_struct.TrackToSplitAncestry;
parent_id=parent_ancestry(ancestry_layout.TrackIDCol);
new_cell_ancestry(ancestry_layout.ParentIDCol)=parent_id;
second_new_cell_ancestry=parent_ancestry;
second_new_cell_ancestry(ancestry_layout.ParentIDCol)=parent_id;
cur_time=new_cell_ancestry(ancestry_layout.StartTimeCol);
second_new_cell_ancestry(ancestry_layout.StartTimeCol)=cur_time;
ancestry_records=mtr_gui_struct.CellsAncestry;
max_cell_id=max(ancestry_records(:,ancestry_layout.TrackIDCol));
second_new_cell_ancestry(ancestry_layout.TrackIDCol)=(max_cell_id+1);
parent_ancestry(ancestry_layout.StopTimeCol)=...
    cur_time-mtr_gui_struct.TimeFrame;
ancestry_records=mtr_gui_struct.CellsAncestry;
parent_ancestry_idx=ancestry_records(:,ancestry_layout.TrackIDCol)==parent_id;
ancestry_records(parent_ancestry_idx,:)=parent_ancestry;
new_cell_ancestry_idx=ancestry_records(:,ancestry_layout.TrackIDCol)...
    ==mtr_gui_struct.SelectedCellID;
ancestry_records(new_cell_ancestry_idx,:)=new_cell_ancestry;
ancestry_records=[ancestry_records; second_new_cell_ancestry];
mtr_gui_struct.CellsAncestry=ancestry_records;
tracks_layout=mtr_gui_struct.TracksLayout;
tracks=mtr_gui_struct.Tracks;
second_new_track_idx=(tracks(:,tracks_layout.TrackIDCol)==parent_id)&...
    (tracks(:,tracks_layout.TimeCol)>=cur_time);
tracks(second_new_track_idx,tracks_layout.TrackIDCol)=(max_cell_id+1);
mtr_gui_struct.Tracks=tracks;
%increase the generation number of the daughter cells and all their
%proginy
offsetGenerationNumber(parent_id,1);

%end completeSplitTrack
end

function completeContinueTrack()
global mtr_gui_struct;

track_to_remove_ancestry=mtr_gui_struct.CurrentAncestryRecord;
track_to_continue_ancestry=mtr_gui_struct.TrackToContinueAncestry;
ancestry_layout=mtr_gui_struct.AncestryLayout;
track_to_remove_start_time=track_to_remove_ancestry(ancestry_layout.StartTimeCol);
track_to_continue_stop_time=track_to_continue_ancestry(ancestry_layout.StopTimeCol);
if (track_to_continue_stop_time~=(track_to_remove_start_time-mtr_gui_struct.TimeFrame))
    mtr_gui_struct.TrackToContinueID=[];
    mtr_gui_struct.TrackToContinueRecord=[];
    mtr_gui_struct.TrackToContinueAncestry=[];
    mtr_gui_struct.ContinueTrack=false;
    warndlg('You cannot continue the track using this track.  The overlap between the tracks is
incorrect!');
    return;
end

tracks_layout=mtr_gui_struct.TracksLayout;
tracks=mtr_gui_struct.Tracks;
track_to_continue_id=mtr_gui_struct.TrackToContinueID;
track_to_remove_id=mtr_gui_struct.SelectedCellID;
```

```
track_to_remove_idx=track_to_remove_id==tracks(:,tracks_layout.TrackIDCol);
%continue the track
tracks(track_to_remove_idx,tracks_layout.TrackIDCol)=track_to_continue_id;
mtr_gui_struct.Tracks=tracks;
track_to_continue_ancestry(ancestry_layout.StopTimeCol)=track_to_remove_ancestry(ancestry_layout.
StopTimeCol);
mtr_gui_struct.CurrentAncestryRecord=track_to_continue_ancestry;
cells_ancestries=mtr_gui_struct.CellsAncestry;
cells_ancestries(cells_ancestries(:,ancestry_layout.TrackIDCol)==track_to_remove_id,:)=[];
cells_ancestries(cells_ancestries(:,ancestry_layout.TrackIDCol)...
    ==track_to_continue_id,:)=track_to_continue_ancestry;
mtr_gui_struct.CellsAncestry=cells_ancestries;
mtr_gui_struct.TrackToContinueID=[];
mtr_gui_struct.TrackToContinueRecord=[];
mtr_gui_struct.TrackToContinueAncestry=[];
mtr_gui_struct.ContinueTrack=false;
mtr_gui_struct.SelectedCellStart=(track_to_continue_ancestry(ancestry_layout.StartTimeCol)...
    ./mtr_gui_struct.TimeFrame)+1;
mtr_gui_struct.SelectedCellID=track_to_continue_id;
updateCellStatus();

%end completeContinueTrack
end

function cell_id=getCellIDFromLabelID()
global mtr_gui_struct;

cur_cell_blob=mtr_gui_struct.CurCellBlob;
[cur_cell_1 cur_cell_2]=find(cur_cell_blob);
cur_cell_centroid=sum([cur_cell_1 cur_cell_2])./sum(cur_cell_blob(:));
tracks_layout=mtr_gui_struct.TracksLayout;
tracks=mtr_gui_struct.Tracks;
cur_time=(mtr_gui_struct.CurFrame-1)*mtr_gui_struct.TimeFrame;
cur_tracks_idx=(tracks(:,tracks_layout.TimeCol)==cur_time);
cur_tracks=tracks(cur_tracks_idx,:);
cell_speeds=mtr_gui_struct.CellSpeeds;
cell_sq_disps=mtr_gui_struct.CellSquareDisplacements;
cur_speeds=cell_speeds(cur_tracks_idx,2);
cur_sq_disps=cell_sq_disps(cur_tracks_idx,2);
cur_centroids=tracks(cur_tracks_idx,tracks_layout.Centroid1Col:tracks_layout.Centroid2Col);
cur_dist=hypot(cur_cell_centroid(1)-cur_centroids(:,1),cur_cell_centroid(2)-cur_centroids(:,2));
[dummy cell_idx]=min(cur_dist);
cur_track_record=cur_tracks(cell_idx,:);
mtr_gui_struct.CurrentSpeed=cur_speeds(cell_idx);
mtr_gui_struct.CurrentSquareDisplacement=cur_sq_disps(cell_idx);
mtr_gui_struct.CurrentTrackRecord=cur_track_record;
cell_id=cur_track_record(tracks_layout.TrackIDCol);

%end getCellIDFromLabelID
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% moveModule.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function moveModule(handles,move_dir)
%helper function for assayEditorGUI. move a module up or down in the assay list
assay_list=get(handles.listboxCurrentAssay,'String');
selection_idx=get(handles.listboxCurrentAssay,'Value');
if (selection_idx==1)&&(move_dir==-1)
    return;
end
list_len=length(assay_list);
if (selection_idx==list_len)&&(move_dir==1)
    return;
end
if (selection_idx==(list_len-1))&&(move_dir==1)
    at_end=true;
else
    at_end=false;
end

selection_text=assay_list{selection_idx};
if strcmp(selection_text(1:9),'<html><i>')
    warndlg('This is not a module. You cannot move parts of a module!');
    return;
end
%get the module struct
module_instance=stripHTMLFromString(selection_text);
modules_list=handles.ModulesList;
modules_map=handles.ModulesMap;
```

```
module_idx=modules_map.get(module_instance);
module_struct=modules_list{module_idx};
new_idx=selection_idx+move_dir;
%this is the module above/below which the current module is being moved
selection_text=assay_list{new_idx};
module_found=false;
chain_vars=false;
while strcmp(selection_text(1:9),'<html><i>')
    new_idx=new_idx+move_dir;
    if (new_idx==list_len)
        fixed_struct=getModuleStruct(assay_list,selection_idx,handles);
        fixed_idx=module_map.get(fixed_struct.ModuleInstance);
        module_found=true;
        break;
    end
    selection_text=assay_list{new_idx};
    chain_vars=true;
end
if (~module_found)
    fixed_instance=stripHTMLFromString(selection_text);
    fixed_idx=modules_map.get(fixed_instance);
    fixed_struct=modules_list{fixed_idx};
end
modules_list(module_idx)=[];
if (module_idx<fixed_idx)
    fixed_idx=fixed_idx-1;
end
assay_list(selection_idx)=[];
if (chain_vars&&fixed_struct.IsParent)
    selection_idx=new_idx;
else
    selection_idx=selection_idx-1;
end

module_struct.Parent=fixed_struct.Parent;
module_struct.Level=fixed_struct.Level;
module_struct.ChainName=fixed_struct.ChainName;

if (move_dir==1)
    if chain_vars
        %skipped chains so insert before rather than after
        %fixed_struct;
        fixed_idx=fixed_idx-1;
    end
    %moving down
    if (at_end)
        %at the end of list
        modules_list=[modules_list; {module_struct}];
        assay_list=[assay_list; formatModuleItem(module_struct)];
    else
        if (fixed_struct.IsParent)
            selection_idx=selection_idx+2;
            list_len=length(assay_list);
            while 1
                if ((selection_idx+1)>list_len)
                    break;
                end
                selection_text=assay_list{selection_idx+1};
                if ~strcmp(selection_text(1:9),'<html><i>')
                    break;
                end
                selection_idx=selection_idx+1;
            end
            %figure out if the insertion point is inside one of the fixed
            %module's chains
            if isChainExpanded(assay_list,selection_idx)
                module_struct.Parent=fixed_struct.InstanceName;
                module_struct.Level=fixed_struct.Level+1;
                %get the chain name
                chain_var=stripHTMLFromString(assay_list{selection_idx});
                chain_idx=strcmp(fixed_struct.ChainVars,chain_var);
                chain_name=fixed_struct.Chains(chain_idx);
                module_struct.ChainName=chain_name{1};
                submodule_instance=stripHTMLFromString(assay_list{selection_idx+1});
                %adjust the fixed_idx so this is the first submodule
                fixed_idx=modules_map.get(submodule_instance)-2;
            end
            selection_idx=selection_idx-1;
        end
        modules_list=[modules_list(1:fixed_idx); module_struct; modules_list((fixed_idx+1):end)];
```

```
        assay_list=[assay_list(1:(selection_idx+1)); formatModuleItem(module_struct);
assay_list((selection_idx+2):end)];
    end
    set(handles.listboxCurrentAssay,'Value',selection_idx+2);
else
    %moving up
    if chain_vars&&(~fixed_struct.IsParent)
        %skipped chains so insert after rather than before
        %fixed_struct;
        fixed_idx=fixed_idx+1;
    end
    if (selection_idx==1)
        %at the beginning of the list
        modules_list=[{module_struct}; modules_list];
        assay_list=[formatModuleItem(module_struct); assay_list];
    else
        modules_list=[modules_list(1:fixed_idx-1); module_struct; modules_list(fixed_idx:end)];
        assay_list=[assay_list(1:selection_idx-1); formatModuleItem(module_struct);
assay_list(selection_idx:end)];
    end
    set(handles.listboxCurrentAssay,'Value',selection_idx);
end

modules_map=java.util.HashMap;
%rebuild the hashmap
for i=1:length(modules_list)
    modules_map.put(modules_list{i}.InstanceName,i);
end
handles.ModulesList=modules_list;
handles.ModulesMap=modules_map;
guidata(handles.figure1,handles);
%update the listbox
set(handles.listboxCurrentAssay,'String',assay_list);

%end moveModule
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% negativeImage.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=negativeImage(input_args)
% Usage
% This module returns the negative of the image provided as an argument.
% Input Structure Members
% Image - Image to be processed.
% Output Structure Members
% Image - Negative image.

output_args.Image=~input_args.Image.Value;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% offsetGenerationNumber.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function offsetGenerationNumber(parent_id,offset_val)
%helper function for manual tracking review. adjust the generation number
%of daughter cells by the offset_val
global mtr_gui_struct;

ancestry_records=mtr_gui_struct.CellsAncestry;
ancestry_layout=mtr_gui_struct.AncestryLayout;
daughters_idx=ancestry_records(:,ancestry_layout.ParentIDCol)==parent_id;
daughter_ids=ancestry_records(daughters_idx,ancestry_layout.TrackIDCol);
if (isempty(daughter_ids))
    return;
end
ancestry_records(daughters_idx,ancestry_layout.GenerationCol)=...
    ancestry_records(daughters_idx,ancestry_layout.GenerationCol)+offset_val;
mtr_gui_struct.CellsAncestry=ancestry_records;
%recurse for all the daughters
for i=1:length(daughter_ids)
    offsetGenerationNumber(daughter_ids(i));
end

%end offsetGenerationNumber
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% openAssayGUI.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function varargout = openAssayGUI(varargin)
% OPENASSAYGUI M-file for openAssayGUI.fig
%      OPENASSAYGUI, by itself, creates a new OPENASSAYGUI or raises the existing
%      singleton*.
%
%      H = OPENASSAYGUI returns the handle to a new OPENASSAYGUI or the handle to
%      the existing singleton*.
%
%      OPENASSAYGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in OPENASSAYGUI.M with the given input arguments.
%
%      OPENASSAYGUI('Property','Value',...) creates a new OPENASSAYGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before openAssayGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to openAssayGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help openAssayGUI

% Last Modified by GUIDE v2.5 26-Jul-2011 12:18:22

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @openAssayGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @openAssayGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before openAssayGUI is made visible.
function openAssayGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to openAssayGUI (see VARARGIN)

% Choose default command line output for openAssayGUI
handles.OK = false;
handles.SelectedAssay='';

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes openAssayGUI wait for user response (see UIRESUME)
uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
function varargout = openAssayGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.OK;
varargout{2}=handles.SelectedAssay;
delete(hObject);
```

```
% --- Executes on selection change in listboxModules.
function listboxModules_Callback(hObject, eventdata, handles)
% hObject    handle to listboxModules (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listboxModules contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listboxModules
selection_type=get(handles.figure1,'SelectionType');
assay_list=get(hObject,'String');
selected_idx=get(hObject,'Value');
selected_assay=assay_list{selected_idx};
handles.SelectedAssay=selected_assay;
switch (selection_type)
    case 'normal'
        set(handles.editModuleDescription,'String',getModuleDescription(selected_assay));
    case 'open'
        handles.OK=true;
        guidata(handles.figure1,handles);
        uiresume(handles.figure1);
end

% --- Executes during object creation, after setting all properties.
function listboxModules_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listboxModules (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
assay_list=getAssaysList();
set(hObject,'String',assay_list);



function editModuleDescription_Callback(hObject, eventdata, handles)
% hObject    handle to editModuleDescription (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editModuleDescription as text
%        str2double(get(hObject,'String')) returns contents of editModuleDescription as a double


% --- Executes during object creation, after setting all properties.
function editModuleDescription_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editModuleDescription (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
assay_list=getAssaysList();
set(hObject,'String',getModuleDescription(assay_list{1}));

% --- Executes on button press in pushbuttonOK.
function pushbuttonOK_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonOK (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
assay_list=get(handles.listboxModules,'String');
selected_idx=get(handles.listboxModules,'Value');
selected_assay=assay_list{selected_idx};
handles.SelectedAssay=selected_assay;
handles.OK=true;
guidata(handles.figure1,handles);
uiresume(handles.figure1);



% --- Executes on button press in pushbuttonCancel.
function pushbuttonCancel_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonCancel (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
uiresume(handles.figure1);



% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: delete(hObject) closes the figure
uiresume(hObject);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% overlayAncestry.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=overlayAncestry(input_args)
% Usage
% This module is used to overlay the cell outlines (color-coded according to generation number)
and the track ids on top of the original cell image.
% Input Structure Members
% AncestryLayout - Matrix describing the order of the columns in the ancestry matrix.
% CurrentTracks - The set of tracks for the current image.
% CellsLabel - The label matrix containing the detected cell shapes for the current image.
% CellsAncestry - Matrix containing the ancestry records for the cells in the time-lapse movie.
% ColorMap - Color map to be used in drawing the cell outlines for each generation. Each
generation will use the next color in the color map until all colors have been used.  Afterwards,
the colors in the map are recycled.
% Image - This is the original cell image.
% ShowLabels - Boolean value. If set to false the cell IDs will not be overlayed.
% ShowOutlines - Boolean value. If set to false the cell outlines will not be overlayed.
% TracksLayout - Matrix describing the order of the columns in the tracks matrix.
% Output Structure Members
% Image - The overlayed image.

cur_img=input_args.Image.Value;
cur_tracks=input_args.CurrentTracks.Value;
cells_lbl=input_args.CellsLabel.Value;
cells_ancestry=input_args.CellsAncestry.Value;
cmap=input_args.ColorMap.Value;

img_sz=size(cur_img);
max_pxl=intmax('uint8');
imnorm_args.IntegerClass.Value='uint8';
imnorm_args.RawImage.Value=cur_img;
imnorm_output=imNorm(imnorm_args);
cur_img=imnorm_output.Image;
red_color=cur_img;
green_color=cur_img;
blue_color=cur_img;

tracks_layout=input_args.TracksLayout.Value;
centroid1Col=tracks_layout.Centroid1Col;
centroid2Col=tracks_layout.Centroid2Col;
trackIDCol=tracks_layout.TrackIDCol;

ancestry_layout=input_args.AncestryLayout.Value;
ancestryIDCol=ancestry_layout.TrackIDCol;
generationCol=ancestry_layout.GenerationCol;
b_show_labels=input_args.ShowLabels.Value;
b_show_outlines=input_args.ShowOutlines.Value;

cur_cell_number=size(cur_tracks,1);

if (b_show_outlines)
    %i need to get the outlines of each individual cell since more than one
    %cell might be in a blob
    avg_filt=fspecial('average',[3 3]);
    lbl_avg=imfilter(cells_lbl,avg_filt,'replicate');
    lbl_avg=double(lbl_avg).*double(cells_lbl>0);
    img_bounds=abs(double(cells_lbl)-lbl_avg);
    img_bounds=img_bounds>0.1;
    bounds_lbl=zeros(img_sz);
    bounds_lbl(img_bounds)=cells_lbl(img_bounds);
```

```
    %draw the cell boundaries
    for j=1:cur_cell_number
        cur_centroid=cur_tracks(j,centroid1Col:centroid2Col);
        cell_id=cur_tracks(j,trackIDCol);
        cell_lbl_id=getLabelId(cells_lbl,cur_centroid);
        cell_generation=cells_ancestry(cells_ancestry(:,ancestryIDCol)==cell_id,generationCol);
        cell_bounds_idx=(bounds_lbl==cell_lbl_id);
        %draw in the red channel
        red_color(cell_bounds_idx)=max_pxl*cmap(cell_generation,1);
        green_color(cell_bounds_idx)=max_pxl*cmap(cell_generation,2);
        blue_color(cell_bounds_idx)=max_pxl*cmap(cell_generation,3);
    end
end

if (b_show_labels)
    %draw the cell labels
    for j=1:cur_cell_number
        cur_centroid=cur_tracks(j,centroid1Col:centroid2Col);
        cell_id=cur_tracks(j,trackIDCol);

        text_img=text2im(num2str(cell_id));
        text_img=imresize(text_img,0.75,'nearest');
        text_length=size(text_img,2);
        text_height=size(text_img,1);
        rect_coord_1=round(cur_centroid(1)-text_height/2);
        rect_coord_2=round(cur_centroid(1)+text_height/2);
        rect_coord_3=round(cur_centroid(2)-text_length/2);
        rect_coord_4=round(cur_centroid(2)+text_length/2);
        if
((rect_coord_1<1)||(rect_coord_2>img_sz(1))||(rect_coord_3<1)||(rect_coord_4>img_sz(2)))
            continue;
        end
        [text_coord_1 text_coord_2]=find(text_img==0);
        %offset the text coordinates by the image coordinates in the (low,low)
        %corner of the rectangle
        text_coord_1=text_coord_1+rect_coord_1;
        text_coord_2=text_coord_2+rect_coord_3;
        text_coord_lin=sub2ind(img_sz,text_coord_1,text_coord_2);
        %write the text in green
        red_color(text_coord_lin)=0;
        green_color(text_coord_lin)=max_pxl;
        blue_color(text_coord_lin)=0;
    end
end

output_args.Image=cat(3,red_color,green_color,blue_color);

% end displayAncestryData
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% overlayPreviousLabel.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function overlayPreviousLabel()
%helper function for manual segmentation review. overlay the previous label
%as a transparency on the image of the current label.
global msr_gui_struct;
overlay_prev_label=get(msr_gui_struct.CheckBoxOverlayPrevLabelHandle,'Value');
if (overlay_prev_label)
    prev_label=msr_gui_struct.PreviousLabel;
    prev_label_layer=prev_label>0;
    image_overlay=repmat(intmax('uint8')*uint8(prev_label_layer),[1 1 3]);
    image_data=get(msr_gui_struct.ImageHandle,'CData');
    mtr_gui_struct.ImageHandle=imagesc(image_data,'Parent',msr_gui_struct.AxesHandle);
    hold on;
    msr_gui_struct.ImageHandle=image(image_overlay,'Parent',msr_gui_struct.AxesHandle);
    set(msr_gui_struct.ImageHandle,'AlphaData',0.3);
    hold off;
    %set the function handle for a mouse click in the objects image
    set(msr_gui_struct.ImageHandle,'buttondownfcn','mouseClickInLabel');
    %toggle select blob button on
    set(msr_gui_struct.SelectBlobButtonHandle,'Enable','off');
    %toggle select object button on
    set(msr_gui_struct.SelectObjectButtonHandle,'Enable','off');
    %disable blob action buttons
    set(msr_gui_struct.ResegmentBlobButtonHandle,'Enable','off');
    set(msr_gui_struct.RemoveBlobButtonHandle,'Enable','off');
    set(msr_gui_struct.RestoreBlobButtonHandle,'Enable','off');
```

```
    %disable object action buttons
    set(msr_gui_struct.JoinObjectsButtonHandle,'Enable','off');
    set(msr_gui_struct.RemoveObjectButtonHandle,'Enable','off');
else
    %toggle select blob button off
    set(msr_gui_struct.SelectBlobButtonHandle,'Enable','on');
    %toggle select object button off
    set(msr_gui_struct.SelectObjectButtonHandle,'Enable','on');
    switch msr_gui_struct.CurrentAction
        case 'ShowImage'
            objects_rgb=msr_gui_struct.Image;
        case 'ShowPreviousLabel'
            prev_label=msr_gui_struct.PreviousLabel;

objects_rgb=label2rgb(prev_label,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
        case 'ShowRawLabel'
            raw_label=msr_gui_struct.OriginalObjectsLabel;

objects_rgb=label2rgb(raw_label,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
        otherwise
            cur_label=msr_gui_struct.ObjectsLabel;

objects_rgb=label2rgb(cur_label,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
    end
    msr_gui_struct.ImageHandle=imagesc(objects_rgb,'Parent',msr_gui_struct.AxesHandle);
    %set the function handle for a mouse click in the objects image
    set(msr_gui_struct.ImageHandle,'buttondownfcn','mouseClickInLabel');
end

%end overlayPreviousLabel
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% pathGoesThroughACell.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function b_path_goes_through_a_cell=pathGoesThroughACell(cells_lbl, prev_cells_lbl, cur_id,
prev_id, bkg_id)
%helper function for CA tracking module. determine if the desired track
%will have to go through another cell
cur_pxls=cells_lbl==cur_id;
prev_pxls=prev_cells_lbl==prev_id;
and_pxls=cur_pxls&prev_pxls;
if (max(and_pxls(:)==1))
    %the current and previous positions overlap to some extent
    b_path_goes_through_a_cell=false;
    return;
end
%the positions do not overlap get the perimeter pixels
%crop the boxes otherwise bwboundaries will be really slow
[cur_pxls_1 cur_pxls_2]=find(cur_pxls);
[prev_pxls_1 prev_pxls_2]=find(prev_pxls);
min_1=min([cur_pxls_1; prev_pxls_1]);
max_1=max([cur_pxls_1; prev_pxls_1]);
min_2=min([cur_pxls_2; prev_pxls_2]);
max_2=max([cur_pxls_2; prev_pxls_2]);
cur_pxls=cur_pxls(min_1:max_1,min_2:max_2);
prev_pxls=prev_pxls(min_1:max_1,min_2:max_2);
cur_perim_pixels=bwboundaries(cur_pxls,'noholes');
cur_perim_pixels=cur_perim_pixels{1};
prev_perim_pixels=bwboundaries(prev_pxls,'noholes');
prev_perim_pixels=prev_perim_pixels{1};
cur_points_nr=size(cur_perim_pixels,1);
prev_points_nr=size(prev_perim_pixels,1);
%compute the pairwise distance matrix between the two sets of points
distance_matrix=zeros(cur_points_nr,prev_points_nr);
for i=1:cur_points_nr
    point_mat=repmat(cur_perim_pixels(i,:),prev_points_nr,1);
    distance_matrix(i,:)=hypot(prev_perim_pixels(:,1)-point_mat(:,1),prev_perim_pixels(:,2)-
point_mat(:,2));
end
min_val=min(distance_matrix(:));
[cur_point_idx prev_point_idx]=find(distance_matrix==min_val,1);
closest_cur_point=cur_perim_pixels(cur_point_idx,:)+[min_1 min_2];
closest_prev_point=prev_perim_pixels(prev_point_idx,:)+[min_1 min_2];
coord_1_len=abs(closest_cur_point(1)-closest_prev_point(1));
coord_2_len=abs(closest_cur_point(2)-closest_prev_point(2));
if (coord_1_len>coord_2_len)
    if (closest_cur_point(1)>closest_prev_point(1))
        coord_1=round([closest_prev_point(1) closest_cur_point(1)]);
        coord_2=round([closest_prev_point(2) closest_cur_point(2)]);
```

```
    else
        coord_1=round([closest_cur_point(1) closest_prev_point(1)]);
        coord_2=round([closest_cur_point(2) closest_prev_point(2)]);
    end
    coord_1_interp=coord_1(1):coord_1(2);
    coord_2_interp=round(interp1q(coord_1',coord_2',coord_1_interp')');
else
    if (closest_cur_point(2)>closest_prev_point(2))
        coord_1=round([closest_prev_point(1) closest_cur_point(1)]);
        coord_2=round([closest_prev_point(2) closest_cur_point(2)]);
    else
        coord_1=round([closest_cur_point(1) closest_prev_point(1)]);
        coord_2=round([closest_cur_point(2) closest_prev_point(2)]);
    end
    coord_2_interp=coord_2(1):coord_2(2);
    coord_1_interp=round(interp1q(coord_2',coord_1',coord_2_interp')');
end
img_sz=size(cells_lbl);
coord_lin=sub2ind(img_sz,coord_1_interp,coord_2_interp);
lbl_ids=unique(prev_cells_lbl(coord_lin));
lbl_ids(lbl_ids==prev_id)=[];
lbl_ids(lbl_ids==bkg_id)=[];
b_path_goes_through_a_cell=~isempty(lbl_ids);

%end pathGoesThroughACell
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% percentageForeground.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=percentageForeground(input_args)
% Usage
% This module calculates the percentage of foreground pixels in a binary image.
% Input Structure Members
% Image - Binary image for which the percentage of foreground pixels is to be calculated.
% Output Structure Members
% PercentageForeground - The percentage of foreground pixels.

img_bw=input_args.Image.Value;
img_sz=size(img_bw);
pct_fgd=sum(img_bw(:))/(img_sz(1)*img_sz(2));
output_args.PercentageForeground=pct_fgd;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% polygonalAssistedWatershed.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=polygonalAssistedWatershed(input_args)
% Usage
% This module is used to prevent a watershed segmentation module (can be another type of
segmentation module) from splitting convex objects. The assumption is that convex objects are
atomic and should not be split. This assumption works well for nuclear stains.
% Input Structure Members
% ConvexObjectsIndex - List containing the index of convex objects. This list may be generated
using the getConvexObjects module.
% ImageLabel - Label matrix containing the original objects before segmentation.
% MinBlobArea - Objects resulting from segmentation that have an area smaller than this value
will be unsegmented.
% WatershedLabel - Label matrix containing the objects after segmentation.
% Output Structure Members
% LabelMatrix - A label matrix containing the objects segmented according to the segmentation in
WatershedLabel with the exception of convex objects, which are left unaltered.

%determine bkg_ids - have to use area because ws_lbl splits the background
%in two or more pieces
nuclei_lbl=input_args.ImageLabel.Value;
nuclei_props=regionprops(nuclei_lbl,'Area');
nuclei_area=[nuclei_props.Area];
area_max=max(nuclei_area);
img_sz=size(nuclei_lbl);
ws_lbl=input_args.WatershedLabel.Value;
ws_props=regionprops(ws_lbl,'Area');
ws_area=[ws_props.Area];
bkg_mask=ismember(ws_lbl,find(ws_area>area_max));
ws_lbl(bkg_mask)=0;
%optional argument MaxObjectSize
field_names=fieldnames(input_args);
```

```
if (max(strcmp(field_names,'MaxObjectSize')))
    maxObjectSize=input_args.MaxObjectSize.Value;
else
    maxObjectSize=Inf;
end
%
%
nuclei_nr=max(nuclei_lbl(:));
convex_idx=input_args.ConvexObjectsIndex.Value;
min_nucl_area=input_args.MinBlobArea.Value;
for i=1:nuclei_nr
    cur_obj=nuclei_lbl==i;
    if (convex_idx(i))
        if (maxObjectSize>sum(cur_obj(:)))
            %if the object is smaller than maxObjectSize and convex don't
            %split it
            continue;
        end
    end
    ws_lbl_obj=ws_lbl(cur_obj);
    ws_cluster_ids=unique(ws_lbl_obj);
    ws_cluster_ids=ws_cluster_ids(ws_cluster_ids>0);
    if(isempty(ws_cluster_ids))
        continue;
    end
    if (length(ws_cluster_ids)==1)
        %the blob is one colony no need to modify cyto_lbl
        continue;
    else
        nr_clusters=length(ws_cluster_ids);
        [blob_1 blob_2]=find(cur_obj);
        segmentation_idx=clusterdata([blob_1 blob_2], 'maxclust', nr_clusters, 'linkage',
'average');
                zero_idx = find(segmentation_idx == 0);
                for(k=1:size(zero_idx))
                        if(zero_idx(k) == 1)
                                segmentation_idx(zero_idx(k)) = ...
                                        segmentation_idx(zero_idx(k) + 1);
                        else
                                segmentation_idx(zero_idx(k)) = ...
                                        segmentation_idx(zero_idx(k) - 1);
                        end
                end
        %the segmentation might create objects that are smaller than our
        %minimum object size - we need to unsegment those
        [valid_segmentation_ids valid_len
new_segmentation_idx]=determineValidObjects(segmentation_idx,...
            min_nucl_area,blob_1,blob_2);
        if (valid_len<2)
            %splitting this blob results in objects that are all or all-but-one smaller
            %than our minimum object size - so no split
            continue;
        end
        if (~isempty(new_segmentation_idx))
            segmentation_idx=new_segmentation_idx;
        end
        nr_clusters=valid_len;
        cur_max=max(nuclei_lbl(:));
        for j=2:nr_clusters
            cur_idx=segmentation_idx==valid_segmentation_ids(j);
            cell_coord_1=blob_1(cur_idx);
            cell_coord_2=blob_2(cur_idx);
            cell_coord_lin=sub2ind(img_sz,cell_coord_1,cell_coord_2);
            nuclei_lbl(cell_coord_lin)=cur_max+j-1;
        end
    end
end

output_args.LabelMatrix=nuclei_lbl;
%end polygonalAssistedWatershed
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% populateInputStringsListbox.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function populateInputStringsListbox(handles)
%helper function for assayEditorGUI. update the input arguments listbox

module_struct=handles.ModuleStruct;
```

```
module_path=[module_struct.ModuleName '.m'];
input_arguments=getInputArgs(module_path);
static_params=cellfun(@(x) x{1},module_struct.StaticParameters,'UniformOutput',false);
output_params=cellfun(@(x) x{1},module_struct.OutputArgs,'UniformOutput',false);
input_strings={};

for i=1:length(input_arguments)
    cur_arg=input_arguments{i};
    arg_params=[];
    %check if any static params match
    match_idx=strcmp(cur_arg,static_params);
    if (max(match_idx)==1)
        arg_params.Static=static_params(match_idx);
        static_params(match_idx)=[];
    end
    %check if any output params match
    match_idx=strcmp(cur_arg,output_params);
    if (max(match_idx)==1)
        arg_params.Output=output_params(match_idx);
        output_params(match_idx)=[];
    end
    input_strings=[input_strings formatInputStrings(cur_arg,arg_params)];
end

set(handles.listboxInputArgumens,'String',input_strings);

%end populateInputStringsListbox
end

function input_strings=formatInputStrings(input_arg, arg_params)
%extract and format static parameter for display

if isempty(arg_params)
    input_strings{1}=['<html><font color="red">' input_arg '</font></html>'];
    return;
else
    input_strings{1}=input_arg;
end

field_names=fieldnames(arg_params);
if (max(strcmp('Output',field_names)==1))
    output_params=arg_params.Output;
    for i=2:(1+length(output_params))
        input_strings{i}=['<html><i> Output' num2str(i-1) '</i></html>'];
    end
end

ls=length(input_strings);
if (max(strcmp('Static',field_names)==1))
    static_params=arg_params.Static;
    for i=(ls+1):(ls+length(static_params))
        input_strings{i}=['<html><i> Value' num2str(i-ls) '</i></html>'];
    end
end

%end formatInputStrings
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% populateModuleInstancesPopup.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function current_selection=populateModuleInstancesPopup(handles)
%helper function for assayEditorGUI. populate the module instance popup

modules_list=handles.ModulesList;
module_instances=cellfun(@(x) x.InstanceName, modules_list,'UniformOutput',false);
module_instances=sort(module_instances);
set(handles.popupModuleInstance,'String',module_instances);
current_selection=module_instances{1};

%end populateModuleInstancesPopup
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% rankParams.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function track_ranks=rankParams(cur_shape_params,nearby_shape_params)
%helper function for CA tracking algorithm. rank how similar each previous nearby cell is to the
current cell
```

```
[nr_tracks nr_params]=size(nearby_shape_params);
track_ranks=zeros(nr_tracks, nr_params);

for i=1:nr_params
    diff_from_cur_param=abs(cur_shape_params(:,i)-nearby_shape_params(:,i));
    [dummy param_rank]=sort(diff_from_cur_param);
    equal_vals_idx=~diff(diff_from_cur_param);
    replace_ranks_idx=[false; equal_vals_idx];
    replace_vals_idx=[equal_vals_idx; false];
    param_rank(replace_ranks_idx)=param_rank(replace_vals_idx);
    track_ranks(:,i)=param_rank;
end
%end rankParams
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% readImage.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=readImage(input_args)
%simple wrapper for the MATLAB imread function
%Input Structure Members
%ImageName - Path to the image to be loaded.
%Output Structure Members
%Image - The image matrix.

image_name=input_args.ImageName.Value;
img_channel=input_args.ImageChannel.Value;
img_to_proc=imread(image_name);
switch img_channel
    case 'r'
        img_to_proc=img_to_proc(:,:,1);
    case 'g'
        img_to_proc=img_to_proc(:,:,2);
    case 'b'
        img_to_proc=img_to_proc(:,:,3);
end
output_args.Image=img_to_proc;

%end readImage
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% readImage3D.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=readImage3D(input_args)
%used to load a 3-D image in memory
%Input Structure Members
%ImageName - Path to the image to be loaded.
%ImageChannel - Which channel to load.
%Output Structure Members
%Image - The image matrix.

image_name=input_args.ImageName.Value;
img_channel=input_args.ImageChannel.Value;
img_info = imfinfo(image_name);
nr_images = numel(img_info);
img_width=img_info.Width;
img_height=img_info.Height;
img_3d=zeros(img_width,img_height,nr_images);
for i=1:nr_images
    cur_img=imread(image_name,i);
    switch img_channel
        case 'r'
            cur_img=cur_img(:,:,1);
        case 'g'
            cur_img=cur_img(:,:,2);
        case 'b'
            cur_img=cur_img(:,:,3);
    end
    img_3d(:,:,i)=cur_img;
end
output_args.Image=img_3d;
img_size(1)=img_width;
img_size(2)=img_height;
img_size(3)=nr_images;
output_args.ImageSize=img_size;

%end readImage3D
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% readImageSlice.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=readImageSlice(input_args)
%read a 2-D image slice from a 3-D image
%Input Structure Members
%ImageName - Path to the image to be loaded.
%ImageChannel - Which channel to load.
%SliceIndex - The index of the image slice to be loaded
%Output Structure Members
%Image - The 2-D image matrix for the current index.
image_name=input_args.ImageName.Value;
img_channel=input_args.ImageChannel.Value;
idx=input_args.SliceIndex.Value;
img_to_proc=imread(image_name,idx);
switch img_channel
    case 'r'
        img_to_proc=img_to_proc(:,:,1);
    case 'g'
        img_to_proc=img_to_proc(:,:,2);
    case 'b'
        img_to_proc=img_to_proc(:,:,3);
end
output_args.Image=img_to_proc;

%end readImage
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% reconstructObjects.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=reconstructObjects(input_args)
%simple wrapper to MATLAB imreconstruct function
%Input Structure Members
%GuideImage - Guide image for the reconstruction.
%ImageToReconstruct - The image to reconstruct.
%Output Structure Members
%Image - The resulting image.
img_and=input_args.GuideImage.Value&input_args.ImageToReconstruct.Value;
output_args.Image=imreconstruct(img_and,input_args.ImageToReconstruct.Value);

%end reconstructObjects
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% rectSelection.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=rectSelection(input_args)
%simple wrapper for MATLAB imrect function.
%Input Structure Members
%ParentHandle - Handle to the image where the freehand region will be
%drawn.
%Output Structure Members
%XYPosition - The xmin,ymin coordinates of the rectangle.
%RectSize - The width and height of the rectangle.

parent_handle=input_args.ParentHandle.Value;
region_handle=imrect(parent_handle,[]);
freehand_api=iptgetapi(region_handle);
position=freehand_api.getPosition();
output_args.XYPosition=position(1:2);
output_args.RectSize=position(3:4);

%end freehandSelection
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% refineSegmentation.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=refineSegmentation(input_args)
% Usage
% This module is used to retain only objects in a label matrix that are nearest to objects in
another matrix.
% Input Structure Members
% CurrentLabel – The label matrix from which objects may be removed if they don’t have an object
to which they are nearest in the PreviousLabel matrix.
```

```
% PreviousLabel - The objects in this label will determine the objects that will be retained in
the current label.
% Output Structure Members
% LabelMatrix - The filtered label matrix.


prev_label=input_args.PreviousLabel.Value;
cur_label=input_args.CurrentLabel.Value;

if (isempty(prev_label))
    output_args.LabelMatrix=cur_label;
    return;
end

prev_centroids=getApproximateCentroids(prev_label);
cur_centroids=getApproximateCentroids(cur_label);
cur_triangulation=delaunay(cur_centroids(:,1),cur_centroids(:,2));

%get the indexes of the nearest centroids in the current label to the
%remaining centroids in the previous label
nearest_idx=dsearch(cur_centroids(:,1),cur_centroids(:,2),cur_triangulation,...
    prev_centroids(:,1),prev_centroids(:,2));
label_ids=unique(nearest_idx);
new_label=bwlabeln(ismember(cur_label,label_ids));
output_args.LabelMatrix=new_label;

%end refineSegmentation
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% removeBlob.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function removeBlob(hObject, eventdata, handles)
%helper function for manual segmentation review. remove the currently
%selected blob
global msr_gui_struct;

selected_blob_ids=msr_gui_struct.SelectedBlobID;
if isempty(selected_blob_ids)
    warnDlg('No Blob is Selected');
    return;
end
objects_lbl=msr_gui_struct.ObjectsLabel;
blobs_lbl=msr_gui_struct.BlobsLabel;
blob_idx=ismember(blobs_lbl,selected_blob_ids);
objects_lbl(blob_idx)=0;
blobs_lbl(blob_idx)=0;
msr_gui_struct.ObjectsLabel=objects_lbl;
msr_gui_struct.BlobsLabel=blobs_lbl;
addSegmentationError('BlobThresholding',selected_blob_ids);
image_data=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
image_handle=msr_gui_struct.ImageHandle;
set(image_handle,'CData',image_data);
msr_gui_struct.CurrentAction='SelectBlob';
msr_gui_struct.SelectedBlobID=[];

%end removeBlob
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% removeLayer.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function removeLayer()
%helper function for manual tracking review. remove selected selection layer
global sl_gui_struct;

layers_txt=get(sl_gui_struct.ListboxSelectionLayersHandle,'String');
if isempty(layers_txt)
    return;
end
layer_idx=get(sl_gui_struct.ListboxSelectionLayersHandle,'Value');
if isempty(layer_idx)
    return;
end
selection_layers=sl_gui_struct.SelectionLayers;
selection_layers(layer_idx)=[];
sl_gui_struct.SelectionLayers=selection_layers;
selection_names=sl_gui_struct.SelectionNames;
```

```
selection_names(layer_idx)=[];
sl_gui_struct.SelectionNames=selection_names;
set(sl_gui_struct.ListboxSelectionLayersHandle,'String',selection_names);

%end removeLayer
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% removeModuleFromAssay.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function removeModuleFromAssay(handles)
%helper function for assayEditorGUI. remove the selected module from the assay

%get the selected module struct
assay_list=get(handles.listboxCurrentAssay,'String');
selection_idx=get(handles.listboxCurrentAssay,'Value');
selection_text=assay_list{selection_idx};
if strcmp(selection_text(1:9),'<html><i>')
    warndlg('This is not a module. You cannot delete parts of a module!');
    return;
end
module_instance=stripHTMLFromString(selection_text);
modules_list=handles.ModulesList;
modules_map=handles.ModulesMap;
module_idx=modules_map.get(module_instance);
module_struct=modules_list{module_idx};
if (module_struct.IsParent)
    %this is a control module
    %remove the chain vars from the assay list
    chain_idx=selection_idx+1;
    selection_text=assay_list{chain_idx};
    list_len=length(assay_list);
    while strcmp(selection_text(1:9),'<html><i>')
        %collapse chain before deleting it
        assay_list=collapseText(assay_list,chain_idx,module_struct.Level);
        assay_list(chain_idx)=[];
        list_len=list_len-1;
        if (chain_idx>list_len)
            break;
        end
        selection_text=assay_list{chain_idx};
    end
end

[modules_list modules_map]=deleteModule(module_struct,modules_list,modules_map);
handles.ModulesList=modules_list;
handles.ModulesMap=modules_map;
%save the new handles struct
guidata(handles.figure1,handles);

assay_list(selection_idx)=[];
list_len=length(assay_list);
if (selection_idx>list_len)
    selection_idx=list_len;
end
set(handles.listboxCurrentAssay,'Value',selection_idx);
set(handles.listboxCurrentAssay,'String',assay_list);

%end removeModuleFromAssay
end

function [modules_list modules_map]=deleteModule(module_struct,modules_list,modules_map)
%remove the module from the modules_list and its index from the module_map

if module_struct.IsParent
    %remove the module's children
    children_idx=cellfun(@(x) strcmp(x.Parent,module_struct.InstanceName),modules_list);
    children_list=modules_list(children_idx);
    for i=1:length(children_list)
        [modules_list modules_map]=deleteModule(children_list{i},modules_list,modules_map);
    end
end

module_idx=modules_map.get(module_struct.InstanceName);
modules_list(module_idx)=[];
modules_map.remove(module_struct.InstanceName);
modules_set=modules_map.entrySet;
set_iter=modules_set.iterator;
%update the indexes above the removal point
while (set_iter.hasNext())
```

```
    map_entry=set_iter.next();
    cur_val=double(map_entry.getValue());
    if (cur_val>module_idx)
        map_entry.setValue(cur_val-1);
    end
end

%end deleteModule
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% removeObject.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function removeObject(hObject, eventdata, handles)
%helper function for manual segmentation review. remove selected object.
global msr_gui_struct;

selected_object_ids=msr_gui_struct.SelectedObjectID;
if isempty(selected_object_ids)
    warnDlg('No Object is Selected');
    return;
end
objects_lbl=msr_gui_struct.ObjectsLabel;
object_idx=ismember(objects_lbl,selected_object_ids);
objects_lbl(object_idx)=0;
msr_gui_struct.ObjectsLabel=objects_lbl;
addSegmentationError('ObjectThresholding',msr_gui_struct.SelectedBlobID);
msr_gui_struct.BlobsLabel=bwlabeln(objects_lbl);
image_data=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
image_handle=msr_gui_struct.ImageHandle;
set(image_handle,'CData',image_data);

%end removeBlob
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% removeProvider.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function removeProvider(handles)
%helper function for assayEditorGUI. remove the selected provider from the
%selected input argument
assay_list=get(handles.listboxInputArgumens,'String');
selection_idx=get(handles.listboxInputArgumens,'Value');
selection_text=assay_list{selection_idx};
if (length(selection_text)<9)||~strcmp(selection_text(1:9),'<html><i>')
    warndlg('You need to select a provider name (in italics)');
    return;
end
module_struct=handles.ModuleStruct;
arg_text=selection_text;
orig_idx=selection_idx;
arg_type=regexp(arg_text,'<i> ([a-zA-Z]*)(\d*)</i>','tokens','once');
selection_idx=selection_idx-1;
input_name=assay_list{selection_idx};
while (length(input_name)>9)&&strcmp(input_name(1:9),'<html><i>')
    selection_idx=selection_idx-1;
    input_name=assay_list{selection_idx};
end
if strcmp(arg_type{1},'Output')
    arg_idx=str2double(arg_type{2});
    match_idx=cellfun(@(x) strcmp(x{1},input_name),module_struct.OutputArgs);
    match_idx=find(match_idx);
    module_struct.OutputArgs(match_idx(arg_idx))=[];
else
    arg_idx=str2double(arg_type{2});
    match_idx=cellfun(@(x) strcmp(x{1},input_name),module_struct.StaticParameters);
    match_idx=find(match_idx);
    module_struct.StaticParameters(match_idx(arg_idx))=[];
end
if isempty(module_struct.OutputArgs)&&isempty(module_struct.StaticParameters)
    %set the argument text in red to warn there's no provider
    assay_list{selection_idx}=['<html><font color="red">' assay_list{selection_idx}
'</font></html>'];
end
if (orig_idx==length(assay_list))
    set(handles.listboxInputArgumens,'Value',orig_idx-1);
end
assay_list(orig_idx)=[];
%reduce the numbers on the remaining providers
```

```
switch(arg_type{1})
    case 'Output'
        output_args=module_struct.OutputArgs;
        if ~isempty(output_args)
            match_idx=cellfun(@(x) strcmp(x{1},input_name),output_args);
            cur_args=output_args(match_idx);
            if ~isempty(cur_args)
                for i=arg_idx:length(cur_args)
                    assay_list{selection_idx+i}=['<html><i> Output' num2str(i)
'</i></html>'];
                end
            end
        end
    case 'Value'
        static_args=module_struct.StaticParameters;
        if ~isempty(static_args)
            match_idx=cellfun(@(x) strcmp(x{1},input_name),static_args);
            cur_args=static_args(match_idx);
            if ~isempty(cur_args)
                match_idx=cellfun(@(x) strcmp(x{1},input_name),module_struct.OutputArgs);
                output_args_len=sum(match_idx);
                for i=arg_idx:length(cur_args)
                    assay_list{selection_idx+output_args_len+i}=['<html><i> Value'
num2str(i) '</i></html>'];
                end
            end
        end
end
set(handles.listboxInputArgumens,'String',assay_list);
handles.ModuleStruct=module_struct;
handles.Erased=true;
guidata(handles.figure1,handles);
inputArgumentsSelChange(handles);

%removeProvider
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% removeShortTracks.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=removeShortTracks(input_args)
% Usage
% This module is used to erase tracks with a lifespan shorter than a specified period of time.
% Input Structure Members
% AncestryLayout - Matrix describing the order of the columns in the ancestry matrix.
% Tracks - The tracks matrix to be processed.
% CellsAncestry - Matrix containing the ancestry records for the cells in the time-lapse movie.
% TracksLayout - Matrix describing the order of the columns in the tracks matrix.
%  MinLifespan - Tracks with a lifespan shorter than this value will be erased.
% Output Structure Members
% Tracks - The filtered tracks matrix.


tracks=input_args.Tracks.Value;
ancestry=input_args.CellsAncestry.Value;
ancestry_layout=input_args.AncestryLayout.Value;
tracks_layout=input_args.TracksLayout.Value;
start_time_col=ancestry_layout.StartTimeCol;
stop_time_col=ancestry_layout.StopTimeCol;
parent_id_col=ancestry_layout.ParentIDCol;
cells_life_span=ancestry(:,stop_time_col)-ancestry(:,start_time_col);
max_time=max(ancestry(:,stop_time_col));
min_life_span=input_args.MinLifespan.Value;
invalid_tracks_idx=cells_life_span<min_life_span;
tracks_starting_late_idx=ancestry(:,start_time_col)>=max_time-min_life_span;
%do not remove the tracks that start in the end frames
invalid_tracks_idx(tracks_starting_late_idx)=false;
tracks_that_are_parents_idx=ismember(ancestry(:,ancestry_layout.TrackIDCol),ancestry(:,parent_id_
col));
%do not remove short tracks that are parents
invalid_tracks_idx(tracks_that_are_parents_idx)=false;
invalid_track_ids=ancestry(invalid_tracks_idx,ancestry_layout.TrackIDCol);
output_args.CellsAncestry=ancestry(~invalid_tracks_idx,:);
invalid_tracks_idx=ismember(tracks(:,tracks_layout.TrackIDCol),invalid_track_ids);
output_args.Tracks=tracks(~invalid_tracks_idx,:);

%end removeShortTracks
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% removeSplit.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function removeSplit()
%helper function for manual tracking review. used to remove an erroneous
%mitotic event
global mtr_gui_struct;

ancestry_layout=mtr_gui_struct.AncestryLayout;
daughter_ancestry_record=mtr_gui_struct.CurrentAncestryRecord;
parent_id=daughter_ancestry_record(ancestry_layout.ParentIDCol);
if (parent_id==0)
    warndlg('This cell is not the result of a split!');
    return;
end
cell_start_frame=daughter_ancestry_record(ancestry_layout.StartTimeCol);
if (cell_start_frame~=((mtr_gui_struct.CurFrame-1)*mtr_gui_struct.TimeFrame))
    warndlg('This cell is not the result of a split in this frame!');
    return;
end
tracks_layout=mtr_gui_struct.TracksLayout;
tracks=mtr_gui_struct.Tracks;
daughter_id=mtr_gui_struct.SelectedCellID;
remove_track_idx=daughter_id==tracks(:,tracks_layout.TrackIDCol);
%remove this daughter cell and use it to continue the former parent cell
tracks(remove_track_idx,tracks_layout.TrackIDCol)=parent_id;
mtr_gui_struct.Tracks=tracks;
%update current track record
current_track_record=mtr_gui_struct.CurrentTrackRecord;
current_track_record(tracks_layout.TrackIDCol)=parent_id;
mtr_gui_struct.CurrentTrackRecord=current_track_record;
%downgrade the generation number for all the daughter cells and all their
%children
offsetGenerationNumber(parent_id,-1);
ancestry_records=mtr_gui_struct.CellsAncestry;
%update the ancestry record of the parent cell with the new stop time
parent_ancestry_idx=ancestry_records(:,ancestry_layout.TrackIDCol)==parent_id;
ancestry_records(parent_ancestry_idx,ancestry_layout.StopTimeCol)=...
    daughter_ancestry_record(ancestry_layout.StopTimeCol);
mtr_gui_struct.CurrentAncestryRecord=ancestry_records(parent_ancestry_idx,:);
%update the parent ids of any remaining daughter cells
daughters_idx=ancestry_records(:,ancestry_layout.ParentIDCol)==parent_id;
ancestry_records(daughters_idx,ancestry_layout.ParentIDCol)=0;
mtr_gui_struct.CellsAncestry=ancestry_records;
mtr_gui_struct.SelectedCellID=parent_id;
mtr_gui_struct.SelectedCellStart=(ancestry_records(parent_ancestry_idx,ancestry_layout.StartTimeC
ol)...
    ./mtr_gui_struct.TimeFrame)+1;
updateCellStatus();
updateTrackImage(mtr_gui_struct.CurFrame,mtr_gui_struct.ShowLabels,mtr_gui_struct.ShowOutlines);
selectCell(mtr_gui_struct.SelectedCellLabelID);

%end removeSplit
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% repmat_Wrapper.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=repmat_Wrapper(input_args)
%this a simple wrapper for the MATLAB function repmat
%Matrix - the matrix to be repeated
%RepeatDim - the dimension array indicating how many times the matrix
%should be repeated in each dimension.

A=input_args.Matrix.Value;
rd=input_args.RepeatDim.Value;
output_args.Matrix=repmat(A,rd);

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% resegmentBlob.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function resegmentBlob(stage)
%helper function for manual segmentation review. used to resegment a blob
%into new objects
switch stage
```

```
    case 'initialize'
        intializeResegmentBlob();
    case 'complete'
        completeResegmentBlob();
end

%end resegmentBlob
end

function intializeResegmentBlob()
global msr_gui_struct;

selected_blob_id=msr_gui_struct.SelectedBlobID;
if isempty(selected_blob_id)
    warnDlg('No Blob is Selected');
    return;
end
if (length(selected_blob_id)>1)
    warnDlg('More than one blob is selected!');
    return;
end
msr_gui_struct.CurrentResegmentationIndex=1;
msr_gui_struct.SegmentationTrainingPoints=[];
msr_gui_struct.SegmentationGroups=[];
updateReviewSegGUIStatus('ResegmentBlob');

%end intializeResegmentBlob
end

function completeResegmentBlob()
global msr_gui_struct;

objects_lbl=msr_gui_struct.ObjectsLabel;
blobs_lbl=msr_gui_struct.BlobsLabel;
blob_id=msr_gui_struct.SelectedBlobID;
[blob_1 blob_2]=find(blobs_lbl==blob_id);
old_object_ids=unique(objects_lbl(blobs_lbl==blob_id));
old_objects_nr=length(old_object_ids);
training_points=msr_gui_struct.SegmentationTrainingPoints;
groups=msr_gui_struct.SegmentationGroups;
new_objects_nr=length(unique(groups));
img_sz=size(objects_lbl);
if (new_objects_nr>old_objects_nr)
    max_id=max(objects_lbl(:));
    new_object_ids=[old_object_ids; max_id+(1:(new_objects_nr-old_objects_nr))'];
    error_type='Undersegmentation';
elseif (new_objects_nr<old_objects_nr)
    new_object_ids=old_object_ids(1:new_objects_nr);
    error_type='Oversegmentation';
else
    new_object_ids=old_object_ids;
    error_type='Distribution';
end
training=[[training_points(:,1); training_points(:,1)-1; training_points(:,1)+1]...
    [training_points(:,2); training_points(:,2)-1; training_points(:,2)+1]];
groups=repmat(new_object_ids(groups),3,1);
segmentation_idx=knnclassify([blob_1 blob_2],training,groups);
for i=1:new_objects_nr
    cur_idx=segmentation_idx==new_object_ids(i);
    obj_coord_1=blob_1(cur_idx);
    obj_coord_2=blob_2(cur_idx);
    obj_coord_lin=sub2ind(img_sz,obj_coord_1,obj_coord_2);
    objects_lbl(obj_coord_lin)=new_object_ids(i);
end

msr_gui_struct.ObjectsLabel=objects_lbl;
image_handle=msr_gui_struct.ImageHandle;
image_data=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
set(image_handle,'CData',image_data);
addSegmentationError(error_type,blob_id);
updateReviewSegGUIStatus('SelectBlob');

%end completeResegmentBlob
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% resizeImage.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=resizeImage(input_args)
%wrapper module for Matlab imresize
```

```
%Input Structure Members
%Image - The image to be processed.
%Method - The method by which the image will be resized (see imresize help)
%Scale - Integer indicating by what amount the image should be
%reduced/enlarged
%Output Structure Members
%Image - The resulting image.
input_img=input_args.Image.Value;
if (isempty(input_img))
    warning 'Input image is empty. resizeImage will return an empty image!';
    output_args.Image=[];
else

output_args.Image=imresize(input_args.Image.Value,input_args.Scale.Value,input_args.Method.Value)
;
end

%end resizeImage
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% restoreBlob.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function restoreBlob(hObject, eventdata, handles)
%helper function for manual segmentation review. used to restore a blob
%from the raw label matrix
global msr_gui_struct;

msr_gui_struct.CurrentAction='RestoreBlob';
original_lbl=msr_gui_struct.OriginalObjectsLabel;
image_handle=msr_gui_struct.ImageHandle;
image_data=label2rgb(original_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
set(image_handle,'CData',image_data);

%end restoreBlob
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% runFunctions.m %%%%%%%%%%%%%%%%%%%%%%%%%%
%core CellAnimation function used to execute modules in the list
function []=runFunctions()
global functions_list;
for i=1:size(functions_list,1)
    function_struct=functions_list{i};
    instance_name=function_struct.InstanceName;
    callFunction(instance_name,true);
end

%end runFunctions
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% saveAncestry.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=saveAncestry(input_args)
% Usage
% This module is used to save the cells ancestry matrix. The matrix is saved with the variable
name cells_ancestry.
% Input Structure Members
% AncestryFileName - The file name to which the cell ancestry matrix should be saved.
% CellsAncestry - The matrix containing the cells ancestry records.
% Output Structure Members
% None

cells_ancestry=input_args.CellsAncestry.Value;
file_name=input_args.AncestryFileName.Value;
save_dir_idx=find(file_name=='/',1,'last');
save_dir=file_name(1:(save_dir_idx-1));
if ~isdir(save_dir)
    mkdir(save_dir);
end
save(file_name,'cells_ancestry');
output_args=[];

%end saveTracks
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%% saveAncestrySpreadsheets.m %%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=saveAncestrySpreadsheets(input_args)
% Usage
% This module is used to save the tracks and ancestry records spreadsheets.
% Input Structure Members
% CellsAncestry - Matrix containing the ancestry records for the cells in the time-lapse movie.
% ProlXlsFile - The desired file name for the spreadsheet containing the ancestry records.
% ShapesXlsFile - The desired file name for the spreadsheet containing the tracks and shape
parameters data.
% Tracks - The tracks matrix to be processed.
% TracksLayout - Matrix describing the order of the columns in the tracks matrix.
% Output Structure Members
% None.


tracks=input_args.Tracks.Value;
cells_ancestry=input_args.CellsAncestry.Value;
tracks_layout=input_args.TracksLayout.Value;
trackIDCol=tracks_layout.TrackIDCol;

%sort tracks_with_stats by cell id
[dummy sort_idx]=sort(tracks(:,trackIDCol));
tracks=tracks(sort_idx,:);
column_names=...
    'Cell ID,Time,Centroid 1,Centroid
2,Area,Eccentricity,MajorAxisLength,MinorAxisLength,Orientation,Perimeter,Solidity';
disp('Saving 2D stats...')
xls_file=input_args.ShapesXlsFile.Value;
save_dir_idx=find(xls_file=='/',1,'last');
save_dir=xls_file(1:(save_dir_idx-1));
if ~isdir(save_dir)
    mkdir(save_dir);
end
delete(xls_file);
dlmwrite(xls_file,column_names,'');
dlmwrite(xls_file,tracks,'-append');
column_names='Cells IDs,Parents IDs,Generations,Start Time,Split Time';
disp('Deleting spreadsheet if it exists...')
xls_file=input_args.ProlXlsFile.Value;
delete(xls_file);
disp('Saving ancestry data...')
dlmwrite(xls_file,column_names,'');
dlmwrite(xls_file,cells_ancestry,'-append');

output_args=[];

%end saveAncestrySpreadsheets
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%% saveAssay.m %%%%%%%%%%%%%%%%%%%%%%%%%%%
function saveAssay(handles,file_name,path)
%helper function for assayEditorGUI. write the current assay to file
if (file_name==0)
    return;
end

function_name=file_name(1:(end-2));

assay_text=['function []=' function_name '()' 10 addAssayDescription(handles) 10 'global
functions_list;' 10 'functions_list=[];' 10];

assay_text=[assay_text addScriptVars(handles)];

assay_text=[assay_text addChainVars(handles)];

assay_end=[10 'global dependencies_list;' 10 'global dependencies_index;' 10
'dependencies_list={};' 10 ];
assay_end=[assay_end 'dependencies_index=java.util.Hashtable;' 10 'makeDependencies([]);' 10
'runFunctions();' 10 'end'];

modules_list=traceArgs(handles);

for i=1:length(modules_list)
```

```
    assay_text=[assay_text buildModuleText(modules_list{i})];
end

assay_text=[assay_text assay_end];
fid=fopen([path '/' file_name],'wt');
fwrite(fid,assay_text);
fclose(fid);
handles.CurrentAssay=file_name;
handles.AssayPath=path;
guidata(handles.figure1,handles);
set(handles.figure1,'Name',['CellAnimation Assay Editor - ' file_name(1:(end-2))]);

%end saveAssay
end

function description_text=addAssayDescription(handles)
%build the comment section at the top of the assay

assay_description=handles.AssayDescription;
description_text='';

for i=1:size(assay_description)
    cur_line=strtrim(assay_description(i,:));
    ws_idx=regexp(cur_line,'(\s)','start');
    ws_nr=length(ws_idx);
    prev_pos_idx=1;
    if ws_nr
        for j=1:14:ws_nr
            if (j+14)>ws_nr
                pos_idx=length(cur_line);
            else
                pos_idx=ws_idx(j+14);
            end
            description_text=[description_text '%' cur_line(prev_pos_idx:pos_idx) 10];
            prev_pos_idx=pos_idx+1;
        end
    else
        description_text=[description_text '%' cur_line 10];
    end
end

%end addAssayDescription
end

function chains_text=addChainVars(handles)
%add the chain variables for all the modules in this assay

chains_text=[10];
modules_list=handles.ModulesList;
parents_idx=cellfun(@(x) x.IsParent, modules_list);
parents_list=modules_list(parents_idx);
for i=1:length(parents_list)
    cur_parent=parents_list{i};
    parent_chains=cur_parent.Chains;
    for j=1:length(parent_chains)
        chains_text=[chains_text parent_chains{j} '=[];' 10];
    end
end
chains_text=[chains_text 10];

%end addChainVars
end


function module_text=buildModuleText(module_struct)

%add the instance name
module_text=[];
instance_var=lower(module_struct.InstanceName);
module_text=[module_text instance_var '.InstanceName=''' module_struct.InstanceName ''';' 10];
module_text=[module_text instance_var '.FunctionHandle=@' module_struct.ModuleName ';' 10];
module_text=[module_text addStaticArgs(module_struct,instance_var)];
[output_args_text args_count]=addOutputArgs(module_struct,instance_var);
module_text=[module_text output_args_text];
module_text=[module_text addInputArgs(module_struct,instance_var,args_count)];
if module_struct.IsParent
    %add any output arguments that are being saved
    module_text=[module_text addKeepOutputArgs(module_struct,instance_var)];
    switch module_struct.ModuleName
        case {'forLoop','whileLoop'}
```

```
            module_text=[module_text instance_var '.' module_struct.ChainVars{1} '='
module_struct.Chains{1} ';' 10];
        case {'if_statement'}
            module_text=[module_text instance_var '.' module_struct.ChainVars{1} '='
module_struct.Chains{1} ';' 10];
            if (length(module_struct.ChainVars)>1)
                module_text=[module_text instance_var '.' module_struct.ChainVars{2} '='
module_struct.Chains{2} ';' 10];
            else
                module_text=[module_text instance_var '.ElseFunctions=[];' 10];
            end
    end
end
module_text=[module_text module_struct.ChainName '=addToFunctionChain(' module_struct.ChainName
',' instance_var ');' 10];
module_text=[module_text 10];

%end buildModuleText
end

function args_text=addStaticArgs(module_struct,instance_var)
%write the module's static arguments
%keep track if an argument appears more than once and write out each to text
args_count=java.util.HashMap;
static_args=module_struct.StaticParameters;
args_text=[];
for i=1:length(static_args)
    cur_arg=static_args{i};
    cur_count=args_count.get(cur_arg{1});
    if isempty(cur_count)
        args_text=[args_text instance_var '.FunctionArgs.' cur_arg{1} '.Value=' cur_arg{2} ';'
10];
        args_count.put(cur_arg{1},2);
    else
        args_text=[args_text instance_var '.FunctionArgs.' cur_arg{1} '.Value' num2str(cur_count)
'=' cur_arg{2} ';' 10];
        args_count.put(cur_arg{1},cur_count+1);
    end
end

%end addStaticArgs
end


function [args_text args_count]=addOutputArgs(module_struct,instance_var)
%write the module's output arguments
%keep track if an argument appears more than once and write out each to
%text
args_count=java.util.HashMap;
args_text=[];
output_args=module_struct.OutputArgs;
for i=1:length(output_args)
    cur_arg=output_args{i};
    cur_count=args_count.get(cur_arg{1});
    if isempty(cur_count)
        args_text=[args_text instance_var '.FunctionArgs.' cur_arg{1} '.FunctionInstance='''
cur_arg{2} '';' 10];
        args_text=[args_text instance_var '.FunctionArgs.' cur_arg{1} '.OutputArg=' cur_arg{3}
';' 10];
        args_count.put(cur_arg{1},2);
    else
        args_text=[args_text instance_var '.FunctionArgs.' cur_arg{1} '.FunctionInstance'
num2str(cur_count) '=''' cur_arg{2} '';' 10];
        args_text=[args_text instance_var '.FunctionArgs.' cur_arg{1} '.OutputArg'
num2str(cur_count) '=' cur_arg{3} ';' 10];
    end
end

%end addOutputArgs
end

function args_text=addKeepOutputArgs(module_struct,instance_var)
%write the module's keepoutput arguments
%keep track if an argument appears more than once and write out each to text
args_count=java.util.HashMap;
args_text=[];
output_args=module_struct.KeepOutputArgs;
for i=1:length(output_args)
    cur_arg=output_args{i};
    cur_count=args_count.get(cur_arg{1});
    if isempty(cur_count)
```

```
        args_text=[args_text instance_var '.KeepValues.' cur_arg{1} '.FunctionInstance='''
cur_arg{2} '';' 10];
        args_text=[args_text instance_var '.KeepValues.' cur_arg{1} '.OutputArg=' cur_arg{3} ';'
10];
        args_count.put(cur_arg{1},2);
    else
        args_text=[args_text instance_var '.KeepValues.' cur_arg{1} '.FunctionInstance'
num2str(cur_count) '=''' cur_arg{2} '';' 10];
        args_text=[args_text instance_var '.KeepValues.' cur_arg{1} '.OutputArg'
num2str(cur_count) '=' cur_arg{3} ';' 10];
    end
end

%end addKeepOutputArgs
end


function args_text=addInputArgs(module_struct,instance_var,args_count)
%write the module's input arguments
%keep track if an argument appears more than once and write out each to
%text
args_text=[];
input_args=module_struct.InputArgs;
for i=1:length(input_args)
    cur_arg=input_args{i};
    cur_count=args_count.get(cur_arg{1});
    if isempty(cur_count)
        args_text=[args_text instance_var '.FunctionArgs.' cur_arg{1} '.FunctionInstance='''
cur_arg{2} '';' 10];
        args_text=[args_text instance_var '.FunctionArgs.' cur_arg{1} '.InputArg=' cur_arg{3} ';'
10];
        args_count.put(cur_arg{1},2);
    else
        args_text=[args_text instance_var '.FunctionArgs.' cur_arg{1} '.FunctionInstance'
num2str(cur_count) '=''' cur_arg{2} '';' 10];
        args_text=[args_text instance_var '.FunctionArgs.' cur_arg{1} '.InputArg'
num2str(cur_count) '=' cur_arg{3} ';' 10];
    end
end

%end addInputArgs
end


function modules_list=traceArgs(handles)
%add input args and keepvalues as necessary to connect a destination
%module to its source module (the module needing an input argument to the
%provider of that input argument)

modules_list=handles.ModulesList;
modules_map=handles.ModulesMap;
for i=1:length(modules_list)
    cur_module=modules_list{i};
    original_args=cur_module.OutputArgs;
    new_args={};
    for j=1:length(original_args)
        cur_arg=original_args{j};
        output_idx=modules_map.get(cur_arg{2});
        output_struct=modules_list{output_idx};
        if
(strcmp(cur_module.ChainName,output_struct.ChainName)||strcmp(cur_module.Parent,output_struct.Ins
tanceName)||...

strcmp(cur_module.InstanceName,output_struct.Parent)||strcmp(cur_module.Parent,output_struct.Pare
nt))
            new_args=[new_args {cur_arg}];
        else
            %replace the output arg with an input arg from a
            %control module. also add all the appropriate connections from
            %source module to destination module
            %first erase the current argument as it will be replaced with a properly
            %connected one
            cur_arg_idx=cellfun(@(x)
strcmp(x{1},cur_arg{1})&&strcmp(x{2},cur_arg{2})&&strcmp(x{3},cur_arg{3}),
cur_module.OutputArgs);
            cur_module.OutputArgs(cur_arg_idx)=[];
            modules_list{i}=cur_module;
            modules_list=getArgument(cur_arg,cur_module,output_struct,modules_list,modules_map);
            cur_module=modules_list{i};
        end
    end
```

```
%     cur_module.OutputArgs=new_args;
    modules_list{i}=cur_module;
end

%end extractInputArgs
end

function new_list=getArgument(cur_arg,cur_module,output_module,modules_list,modules_map)
%find the path to the argument and add input arguments or keepvalues at each point
%get the parents for the cur_module
new_list=modules_list;
%create a unique argument name
output_name=cur_arg{3};
output_name=output_name(2:(end-1));
arg_name=[output_module.InstanceName '_' output_name];
dest_modules=getModuleInstances(cur_module,new_list,modules_map);
source_modules=getModuleInstances(output_module,new_list,modules_map);
[dest_chains dest_chains_modules]=getChainsList(cur_module,new_list,modules_map);
[source_chains source_chains_modules]=getChainsList(output_module,new_list,modules_map);
dest_chains_modules(end)=dest_chains(end);
source_chains_modules(end)=source_chains(end);

%the output module may be a parent of one of the cur_module's parents
if isempty(cur_module.Parent)
    common_module=[];
    common_chain={'functions_list'};
    highest_dest=cur_module;
else
    %get the common module
    common_idx=ismember(dest_modules,source_modules);
    common_module_idx=find(common_idx==1,1,'first');
    if isempty(common_module_idx)
        common_module=[];
    else
        common_module=dest_modules(common_module_idx);
    end
    %get the common module
    common_idx=ismember(dest_chains_modules,source_chains_modules);
    common_chain_idx=find(common_idx==1,1,'first');
    if isempty(common_chain_idx)
        common_chain=[];
    else
        common_chain=dest_chains_modules(common_chain_idx);
    end
    %move up from the cur_module to second-highest chain setting up input
    %arguments
    cur_parent=cur_module;
    parent_idx=modules_map.get(cur_parent.InstanceName);
    i=1;

while(~strcmp(cur_parent.InstanceName,common_module)&&~strcmp(cur_parent.Parent,common_chain))
        if isempty(cur_parent.Parent)
            break;
        end
        input_args=cur_parent.InputArgs;
        if (i==1)
            %for the actual destination we must use the proper argument
            %name
            new_arg={{cur_arg{1} cur_parent.Parent ['''' arg_name '''']}};
        else
            %any intermediary control modules we use the unique name we
            %made up
            new_arg={{arg_name cur_parent.Parent ['''' arg_name '''']}};
        end
        existing_arg_names=cellfun(@(x) x{3}, input_args,'UniformOutput',false);
        if (isempty(input_args)||(max(strcmp(existing_arg_names,['''' arg_name '''']))==0))
            %if the argument doesn't already exists add it
            cur_parent.InputArgs=[input_args new_arg];
            new_list{parent_idx}=cur_parent;
        end
        parent_idx=modules_map.get(cur_parent.Parent);
        cur_parent=new_list{parent_idx};
        i=i+1;
    end
    %get the highest level destination
    highest_dest=cur_parent;
end

%move up to first common parent from output_module and make sure the
%output value is saved
cur_child=output_module;
```

```
i=1;
while(~strcmp(cur_child.Parent,common_module)&&~strcmp(cur_child.InstanceName,common_chain))
    if isempty(cur_child.Parent)
        break;
    end
    cur_parent_idx=modules_map.get(cur_child.Parent);
    cur_parent=new_list{cur_parent_idx};
    if (i==1)
        new_val={{arg_name cur_child.InstanceName cur_arg{3}}};
    else
        new_val={{arg_name cur_child.InstanceName ['''' arg_name '''']}};
    end
    keep_vals=cur_parent.KeepOutputArgs;
    if isempty(keep_vals)
        %this keepval doesn't exist so add it
        cur_parent.KeepOutputArgs=[keep_vals new_val];
        new_list{cur_parent_idx}=cur_parent;
    else
        output_idx=cellfun(@(x) strcmp(x{3},new_val{1}{3})&strcmp(x{2},new_val{1}{2}),
keep_vals);
        if (max(output_idx)==0)
            %this keepval doesn't exist so add it
            cur_parent.KeepOutputArgs=[keep_vals new_val];
            new_list{cur_parent_idx}=cur_parent;
        end
    end
    cur_child=cur_parent;
    i=i+1;
end

%get the highest-level source
highest_source=cur_child;
highest_dest_idx=modules_map.get(highest_dest.InstanceName);
highest_dest=new_list{highest_dest_idx};


%finally add an output arg connecting the highest level destination with the highest level source
if strcmp(highest_dest.InstanceName,cur_module.InstanceName)
    new_arg={{cur_arg{1} highest_source.InstanceName ['''' arg_name '''']}};
elseif strcmp(highest_source.InstanceName,output_module.InstanceName)
    new_arg={{arg_name highest_source.InstanceName cur_arg{3}}};
else
    new_arg={{arg_name highest_source.InstanceName ['''' arg_name '''']}};
end
output_args=highest_dest.OutputArgs;
if isempty(output_args)
    %this output arg doesn't exist so add it
    highest_dest.OutputArgs=[output_args new_arg];
else
    output_idx=cellfun(@(x) strcmp(x{3},new_arg{1}{3})&strcmp(x{2},new_arg{1}{2}), output_args);
    if (max(output_idx)==0)
        %this output arg doesn't exist so add it
        highest_dest.OutputArgs=[output_args new_arg];
    end
end
new_list{highest_dest_idx}=highest_dest;


%end getArgument
end

function module_instances=getModuleInstances(cur_module,modules_list,modules_map)

module_instances={cur_module.InstanceName};
while ~isempty(cur_module.Parent)
    module_instances=[module_instances {cur_module.Parent}];
    cur_parent_idx=modules_map.get(cur_module.Parent);
    cur_module=modules_list{cur_parent_idx};
end

%end getModuleChain
end

function [chains_list chains_modules]=getChainsList(cur_module,modules_list,modules_map)

chains_list={};
chains_modules={};
if cur_module.IsParent
    chains=cur_module.Chains;
    for i=1:length(chains)
        chains_list=[chains_list chains(i)];
```

```
        chains_modules=[chains_modules {cur_module.InstanceName}];
    end
end
chains_list=[chains_list {cur_module.ChainName}];
chains_modules=[chains_modules {cur_module.Parent}];
while ~isempty(cur_module.Parent)
    cur_parent_idx=modules_map.get(cur_module.Parent);
    cur_module=modules_list{cur_parent_idx};
    chains_list=[chains_list {cur_module.ChainName}];
    chains_modules=[chains_modules {cur_module.Parent}];
end

%end getModuleChain
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%% saveCellsLabel.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=saveCellsLabel(input_args)
% Usage
% This module is used to save a MATLAB label matrix containing cell objects.
% Input Structure Members
% CellsLabel - The label matrix containing cell objects.
% CurFrame - The index of the frame to which the label matrix corresponds.
% FileRoot - String containing the root of the file name to be used when saving the label matrix.
% NumberFormat - String indicating the number format to be used when formatting the current frame
number to be concatenated to the file root string. See the MATLAB sprintf help file for example
number format strings.
%
% Output Structure Members
% CellsLabel - The label matrix containing cell objects.


cells_lbl=input_args.CellsLabel.Value;
file_root=input_args.FileRoot.Value;
save_dir_idx=find(file_root=='/',1,'last');
save_dir=file_root(1:(save_dir_idx-1));
if ~isdir(save_dir)
    mkdir(save_dir);
end
save([input_args.FileRoot.Value
num2str(input_args.CurFrame.Value,input_args.NumberFormat.Value)],'cells_lbl');
output_args.CellsLabel=cells_lbl;

%end saveCellsLabel
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%% saveCellsLabelAsImage.m %%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=saveCellsLabel(input_args)
% Usage
% This module is used to save a MATLAB label matrix containing cell objects.
% Input Structure Members
% CellsLabel - The label matrix containing cell objects.
% CurFrame - The index of the frame to which the label matrix corresponds.
% FileRoot - String containing the root of the file name to be used when saving the label matrix.
% NumberFormat - String indicating the number format to be used when formatting the current frame
number to be concatenated to the file root string. See the MATLAB sprintf help file for example
number format strings.
%
% Output Structure Members
% CellsLabel - The label matrix containing cell objects.


cells_lbl=input_args.CellsLabel.Value;
file_root=input_args.FileRoot.Value;
save_dir_idx=find(file_root=='/',1,'last');
save_dir=file_root(1:(save_dir_idx-1));
if ~isdir(save_dir)
    mkdir(save_dir);
end
save([input_args.FileRoot.Value
num2str(input_args.CurFrame.Value,input_args.NumberFormat.Value)],'cells_lbl');
output_args.CellsLabel=cells_lbl;

%end saveCellsLabel
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% saveChanges.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function saveChanges(hObject, eventdata, handles)
%helper function for manual segmentation review. save changes made to the
%label matrix and close the GUI
global msr_gui_struct;

gui_handle=msr_gui_struct.GuiHandle;
msr_gui_struct.FigurePosition=get(gui_handle,'Position');
msr_gui_struct.ObjectsLabel=makeContinuousLabelMatrix(msr_gui_struct.ObjectsLabel);
close(gui_handle);

%end saveChanges
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% saveLayer.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function saveLayer()
%helper function for manual tracking review module. save the new selection
%layer
global al_gui_struct;

selection_name=get(al_gui_struct.EditSelectionLayerNameHandle,'String');
if (isempty(selection_name))
    warndlg('No selection name has been entered!');
    return;
end
if (max(strcmp(selection_name,al_gui_struct.SelectionNames)))
    warndlg('This selection name is already in use!');
    return;
end
if (isempty(al_gui_struct.Conditions))
    warndlg('There are no conditions!');
    return;
end
layer_colors=al_gui_struct.LayerColors;
selection_color=layer_colors{get(al_gui_struct.ComboColorHandle,'value')};
selection_layer.Name=selection_name;
selection_layer.Color=selection_color;
selection_layer.Conditions=al_gui_struct.Conditions;
al_gui_struct.NewSelectionLayer=selection_layer;
close(al_gui_struct.GUIHandle);

%end saveLayer
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% saveMatchingGroups.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=saveMatchingGroups(input_args)
%module to save matching group list
%Input Structure Members
%MatchingGroups - Matrix containing the matching groups.
%MatchingGroupsFileName - Path to the location where the matching groups
%data will be saved
%Output Structure Members
%None
matching_groups=input_args.MatchingGroups.Value;
save(input_args.MatchingGroupsFileName.Value,'matching_groups');
output_args=[];

%end saveMatchingGroups
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% saveMatrixToSpreadsheet.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=saveMatrixToSpreadsheet(input_args)
% Usage
% This module is used to export the matrix to a csv file.
% Input Structure Members
% ColumnNames - The names of the column headers in the csv file.
% Matrix - The matrix containing the values to be saved.
```

```
% SpreadsheetFileName - The desired file name for the saved file.
% Output Structure Members
% None.

region_props=input_args.Matrix.Value;
%sort tracks_with_stats by cell id
column_names=input_args.ColumnNames.Value;
disp('Saving matrix to spreadsheet...');
spreadsheet_file=input_args.SpreadsheetFileName.Value;
delete(spreadsheet_file);
dlmwrite(spreadsheet_file,column_names,'');
dlmwrite(spreadsheet_file,region_props,'-append');
output_args=[];

%end saveRegionPropsSpreadsheets
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% saveOffsets.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=saveOffsets(input_args)
%Usage
%This module is used to save the xy offset data to file
% Input Structure Members
% FileName - The path to the location where the offset data should be saved.
% XYOffsets - The matrix containing the offsets data.
% Output Structure Members
% None


xy_offsets=input_args.XYOffsets.Value;
file_name=input_args.FileName.Value;
save_dir_idx=find(file_name=='/',1,'last');
save_dir=file_name(1:(save_dir_idx-1));
if ~isdir(save_dir)
    mkdir(save_dir);
end
save(file_name,'xy_offsets');
output_args=[];

%end saveTracks
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% saveRegionPropsSpreadsheets.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=saveRegionPropsSpreadsheets(input_args)
% Usage
% This module is used to save the shape parameters extracted using the getRegionProps wrapper
module.
% Input Structure Members
% RegionProps - The matrix containing the shape parameters.
% SpreadsheetFileName - The desired file name for the saved file.
% Output Structure Members
% None.

region_props=input_args.RegionProps.Value;
%sort tracks_with_stats by cell id
column_names=...
    'Cell ID,Centroid 1,Centroid
2,Area,Eccentricity,MajorAxisLength,MinorAxisLength,Orientation,Perimeter,Solidity';
disp('Saving region props...')
spreadsheet_file=input_args.SpreadsheetFileName.Value;
delete(spreadsheet_file);
dlmwrite(spreadsheet_file,column_names,'');
dlmwrite(spreadsheet_file,region_props,'-append');
output_args=[];

%end saveRegionPropsSpreadsheets
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% saveSpreadsheet.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=saveSpreadsheet(input_args)
%module to save the ancestry records to spreadsheets
```

```
% Input Structure Members
% Columns - The matrix containing the ancestry data.
% ColumnHeaders - Array containing column header names.
% XlsFile - Path to the location where the spreadsheet will be saved.
% Output Structure Members
% None.

disp('Saving Spreadsheet...')
columns=input_args.Columns.Value;
column_headers=input_args.ColumnHeaders.Value;
xls_file=input_args.XlsFile.Value;
delete(xls_file);
dlmwrite(xls_file,column_headers,'');
dlmwrite(xls_file,columns,'-append');
output_args=[];
disp('Saved!')

%end saveAncestrySpreadsheets
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% saveTrackingImage.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function saveTrackingImage()
%helper function for manual tracking review module. save the current image
global mtr_gui_struct;

[file_name path_name filter_index]=uiputfile({'*.jpg';'*.tif'},'Save Frame Image');
if (~filter_index)
    return;
end
hidden_figure_handle=figure('visible','off');
% copy axes into the new figure
set(hidden_figure_handle,'PaperPositionMode','auto');
subplot('Position', [0 0 1 1]);
hidden_axes=copyobj(mtr_gui_struct.TracksHandle,hidden_figure_handle);
set(hidden_axes, 'units', 'normalized', 'position', [0 0 1 1]);
switch (filter_index)
    case 1
        print(hidden_figure_handle, '-djpeg', [path_name file_name]);
    case 2
        print(hidden_figure_handle, '-dtiff', '-r300', [path_name file_name]);
end

%end saveImage
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% saveTracks.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=saveTracks(input_args)
% Usage
% This module is used to save the tracks matrix.
% Input Structure Members
% Tracks - Matrix containing the tracks to be saved.
% TracksFileName - The desired file name for the saved tracks data.
% Output Structure Members
% None.


tracks=input_args.Tracks.Value;
file_name=input_args.TracksFileName.Value;
save_dir_idx=find(file_name=='/',1,'last');
save_dir=file_name(1:(save_dir_idx-1));
if ~isdir(save_dir)
    mkdir(save_dir);
end
save(file_name,'tracks');
output_args=[];

%end saveTracks
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% saveWrapper.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=saveWrapper(input_args)
```

```
%imple module to wrap the Matlab save function
% Input Structure Members
% SaveData - Data to be saved.
% FileName - Path to the location where the data will be saved.
% Output Structure Members
% None.


saved_data=input_args.SaveData.Value;
save(input_args.FileName.Value,'saved_data');
output_args=[];

%end saveCellsLabel
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% scriptVariablesGUI.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function varargout = scriptVariablesGUI(varargin)
% SCRIPTVARIABLESGUI M-file for scriptVariablesGUI.fig
%      SCRIPTVARIABLESGUI, by itself, creates a new SCRIPTVARIABLESGUI or raises the existing
%      singleton*.
%
%      H = SCRIPTVARIABLESGUI returns the handle to a new SCRIPTVARIABLESGUI or the handle to
%      the existing singleton*.
%
%      SCRIPTVARIABLESGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SCRIPTVARIABLESGUI.M with the given input arguments.
%
%      SCRIPTVARIABLESGUI('Property','Value',...) creates a new SCRIPTVARIABLESGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before scriptVariablesGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to scriptVariablesGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help scriptVariablesGUI

% Last Modified by GUIDE v2.5 19-Sep-2011 22:24:03

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @scriptVariablesGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @scriptVariablesGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before scriptVariablesGUI is made visible.
function scriptVariablesGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to scriptVariablesGUI (see VARARGIN)

% Choose default command line output for scriptVariablesGUI
handles.output = hObject;
handles.OK=false;
handles.PrevSelectionIdx=1;
mt_idx=find(strcmp(varargin, 'ScriptVariables'))+1;
script_vars=varargin{mt_idx};
handles.ScriptVariables=script_vars;
%show the script variables names
```

```
if ~isempty(script_vars)
    var_names=cellfun(@(x) x{1}, script_vars,'UniformOutput',false);
    set(handles.listboxScriptVars,'String',var_names);
    %show the value of the first variable
    var1=script_vars{1};
    var_val=var1{2};
    set(handles.editScriptVarVal,'String',var_val);
else
    set(handles.listboxScriptVars,'String','');
end

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes scriptVariablesGUI wait for user response (see UIRESUME)
uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = scriptVariablesGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1}=handles.OK;
varargout{2}=handles.ScriptVariables;
delete(hObject);


% --- Executes on selection change in listboxScriptVars.
function listboxScriptVars_Callback(hObject, eventdata, handles)
% hObject    handle to listboxScriptVars (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listboxScriptVars contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listboxScriptVars
%first update the value of the previously selected variable
prev_idx=handles.PrevSelectionIdx;
script_vars=handles.ScriptVariables;
if isempty(script_vars)
    return;
end
script_vars{prev_idx}{2}=get(handles.editScriptVarVal,'String');
var_idx=get(hObject,'Value');
cur_var=script_vars{var_idx};
%update the edit box with the value of the current variable
set(handles.editScriptVarVal,'String',cur_var{2});
handles.PrevSelectionIdx=var_idx;
handles.ScriptVariables=script_vars;
guidata(handles.figure1,handles);


% --- Executes during object creation, after setting all properties.
function listboxScriptVars_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listboxScriptVars (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editScriptVarVal_Callback(hObject, eventdata, handles)
% hObject    handle to editScriptVarVal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editScriptVarVal as text
%        str2double(get(hObject,'String')) returns contents of
%        editScriptVarVal as a double


% --- Executes during object creation, after setting all properties.
function editScriptVarVal_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to editScriptVarVal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbuttonOK.
function pushbuttonOK_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonOK (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.OK=true;
%update the value of the current variable
cur_idx=get(handles.listboxScriptVars,'Value');
script_vars=handles.ScriptVariables;
script_vars{cur_idx}{2}=get(handles.editScriptVarVal,'String');
handles.ScriptVariables=script_vars;
guidata(handles.figure1,handles);
uiresume(handles.figure1);


% --- Executes on button press in pushbuttonCancel.
function pushbuttonCancel_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonCancel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
uiresume(handles.figure1);


% --- Executes on button press in pushbuttonAdd.
function pushbuttonAdd_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonAdd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
var_name=inputdlg('Name','Enter variable name',1,{''},'on');
if isempty(var_name)
    return;
end
var_val=get(handles.editScriptVarVal,'String');
new_var={var_name{1},var_val};
script_vars=handles.ScriptVariables;
script_vars=[script_vars {new_var}];
var_names=cellfun(@(x) x{1}, script_vars,'UniformOutput',false);
set(handles.listboxScriptVars,'String',var_names);
set(handles.listboxScriptVars,'Value',length(script_vars));
handles.PrevSelectionIdx=length(script_vars);
handles.ScriptVariables=script_vars;
guidata(handles.figure1,handles);


% --- Executes on button press in pushbuttonDelete.
function pushbuttonDelete_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonDelete (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%delete the current script variable
var_idx=get(handles.listboxScriptVars,'Value');
script_vars=handles.ScriptVariables;
script_vars(var_idx)=[];
handles.ScriptVariables=script_vars;
if isempty(script_vars)
    return;
end
%update the variables list
var_names=cellfun(@(x) x{1}, script_vars,'UniformOutput',false);
set(handles.listboxScriptVars,'String',var_names);
if (var_idx>1)
    set(handles.listboxScriptVars,'Value',(var_idx-1));
    %update the var value
    set(handles.editScriptVarVal,'String',script_vars{var_idx-1}{2});
else
    %update the var value
    set(handles.editScriptVarVal,'String',script_vars{1}{2});
end

guidata(handles.figure1,handles);
```

```
% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: delete(hObject) closes the figure
uiresume(hObject);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% segmentCytoUsingNuclei.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=segmentCytoUsingNuclei(input_args)
%segments cytoplasm binary image using data from nuclear label matrix
img_cyto=input_args.CytoImage.Value;
nucl_lbl=input_args.NuclearLabel.Value;

%do preliminary segmentation of cytoplasm
cyto_lbl=bwlabeln(img_cyto);
new_lbl=zeros(size(cyto_lbl));
%segmenting the clusters into individual cells


%get the nuclei ids present in the cluster
nr_clusters=max(cyto_lbl(:));
for i=1:nr_clusters
    cur_cluster=(cyto_lbl==i);
    %get the nuclei ids present in the cluster
    nucl_ids=nucl_lbl(cur_cluster);
    nucl_ids=unique(nucl_ids);
    %remove the background id
    nucl_ids(nucl_ids==0)=[];
    if isempty(nucl_ids)
        %don't add objects without nuclei
        continue;
    end
    if (length(nucl_ids)==1)
        %only one nucleus - assign the entire cluster to that id
        new_lbl(cur_cluster)=nucl_ids;
        continue;
    end
    %get an index to only the nuclei
    nucl_idx=ismember(nucl_lbl,nucl_ids);
    %get the x-y coordinates
    [nucl_x nucl_y]=find(nucl_idx);
    [cluster_x cluster_y]=find(cur_cluster);
    group_data=nucl_lbl(nucl_idx);
    %classify each pixel in the cluster
    ss=200;
    pixel_class=knnclassify([cluster_x cluster_y],[nucl_x(1:ss:end)
nucl_y(1:ss:end)],group_data(1:ss:end));
    new_lbl(cur_cluster)=pixel_class;
end

output_args.LabelMatrix=makeContinuousLabelMatrix(new_lbl);

%end segmentCytoUsingNuclei
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% segmentObjectsUsingClusters.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=segmentObjectsUsingClusters(input_args)
% Usage
% This module is used to segment objects in a label matrix using hierarchical clustering.
% Input Structure Members
% ClusterDistance - The height threshold for the cluster tree. All leaves below this value will
be grouped in a cluster. See documentation for the MATLAB function cluster for more details.
% MinimumObjectArea - Objects with an area smaller than this value will be unsegmented and
distributed between the neighboring objects.
% ObjectsLabel - The label matrix containing the objects to be segmented.
% ObjectReduce - Used to reduce the size of the objects. If the objects are too large the
clustering function will run out of memory. When this happens set ObjectReduce to a value lower
than one.
```

```
% Output Structure Members
% LabelMatrix - The label matrix containing the segmented objects.


cells_lbl=input_args.ObjectsLabel.Value;
%should only set obj_reduce to a value smaller than 1 if the
%getBlobClusters runs out of memory with it set to 1.
obj_reduce=input_args.ObjectReduce.Value;
cluster_dist=input_args.ClusterDistance.Value;
min_object_area=input_args.MinimumObjectArea.Value;
img_sz=size(cells_lbl);

obj_nr=max(cells_lbl(:));
% assign each unassigned pixel in a blob to the closest polygon
% showmaxfigure(1),imshow(img_to_proc_norm)
for i=1:obj_nr
    cur_obj=cells_lbl==i;
    if (obj_reduce<1)
        simple_obj=imresize(cur_obj, obj_reduce,'nearest');
    else
        simple_obj=cur_obj;
    end
    [blob_1 blob_2]=find(simple_obj);
    if (size(blob_1,1)<2)
        continue;
    end
    [nr_clusters linkage_clusters]=getBlobClusters([blob_1 blob_2],cluster_dist);
    if (nr_clusters==1)
        %the blob is one colony no need to modify cells_lbl
        continue;
    end

    if (obj_reduce<1)
        %bring the blob coord back to original size
        cluster_1=blob_1/obj_reduce;
        cluster_2=blob_2/obj_reduce;
        [blob_1 blob_2]=find(cur_obj);
        %use nearest neighbor to assign all the pixels in the blob to
        %the clusters
        segmentation_idx=knnclassify([blob_1 blob_2],[cluster_1 cluster_2],linkage_clusters);
    else
        segmentation_idx=linkage_clusters;
    end
     %the segmentation might create objects that are smaller than our
     %minimum object size - we need to unsegment those
    [valid_segmentation_ids valid_len
new_segmentation_idx]=determineValidObjects(segmentation_idx,...
        min_object_area,blob_1,blob_2);
    if (valid_len<2)
        %splitting this blob results in objects that are all or all-but-one smaller
        %than our minimum object size - so no split
        continue;
    end
    if (~isempty(new_segmentation_idx))
        segmentation_idx=new_segmentation_idx;
    end
    nr_clusters=valid_len;
    cur_max=max(cells_lbl(:));
    nr_points=accumarray(segmentation_idx,1);
    centroid_1=accumarray(segmentation_idx,blob_1)./nr_points;
    centroid_2=accumarray(segmentation_idx,blob_2)./nr_points;
    invalid_ids_idx=(nr_points==0);
    centroid_1(invalid_ids_idx)=[];
    centroid_2(invalid_ids_idx)=[];
    training=[[centroid_1; centroid_1-1; centroid_1+1] [centroid_2; centroid_2-1; centroid_2+1]];
    groups=repmat(valid_segmentation_ids',3,1);
    segmentation_idx=classify([blob_1 blob_2],training,groups,'diaglinear');
    for j=2:nr_clusters
        cur_idx=segmentation_idx==valid_segmentation_ids(j);
        cell_coord_1=blob_1(cur_idx);
        cell_coord_2=blob_2(cur_idx);
        cell_coord_lin=sub2ind(img_sz,cell_coord_1,cell_coord_2);
        cells_lbl(cell_coord_lin)=cur_max+j-1;
    end
end

output_args.LabelMatrix=cells_lbl;
%end segmentObjectsUsingClusters
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% segmentObjectsUsingMarkers.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=segmentObjectsUsingMarkers(input_args)
% Usage
% This module is used to segment objects in a label matrix using markers from another label
matrix.
% Input Structure Members
% MarkersLabel - The label matrix containing the marker objects.
% ObjectsLabel - The label matrix containing the objects to be segmented.
% Output Structure Members
% LabelMatrix - Label matrix containing the segmented objects.

nuclei_lbl=input_args.MarkersLabel.Value;
cyto_lbl=input_args.ObjectsLabel.Value;
img_sz=size(nuclei_lbl);

nuclei_nr=max(nuclei_lbl(:));
nuclei_centroids_1=zeros(nuclei_nr,1);
nuclei_centroids_2=zeros(nuclei_nr,1);
for i=1:nuclei_nr
    [cur_obj_1 cur_obj_2]=find(nuclei_lbl==i);
    obj_length=length(cur_obj_1);
    nuclei_centroids_1(i)=sum(cur_obj_1)./obj_length;
    nuclei_centroids_2(i)=sum(cur_obj_2)./obj_length;
end
nuclei_centroids_lin=sub2ind(img_sz,round(nuclei_centroids_1),round(nuclei_centroids_2));
cyto_idx=cyto_lbl(nuclei_centroids_lin);

cells_nr=max(cyto_lbl(:));
for i=1:cells_nr
    cluster_ids=find(cyto_idx==i);
    nr_clusters=length(cluster_ids);
    if (nr_clusters<2)
        continue;
    end
    %this is assuming the centroids are synched with the nuclei label ids
    training_idx=ismember(nuclei_lbl,cluster_ids);
    [blob_1 blob_2]=find(cyto_lbl==i);
    [training_1 training_2]=find(training_idx);
    training_group=nuclei_lbl(training_idx);
    %we'll use only every fifth point as a training point or we will
    %run out of memory
    k_idx=knnclassify([blob_1 blob_2],[training_1(1:5:end)
training_2(1:5:end)],training_group(1:5:end));
    cur_max=max(cyto_lbl(:));
    for j=2:nr_clusters
        cell_coord_1=blob_1(k_idx==cluster_ids(j));
        cell_coord_2=blob_2(k_idx==cluster_ids(j));
        cell_coord_lin=sub2ind(img_sz,cell_coord_1,cell_coord_2);
        cyto_lbl(cell_coord_lin)=cur_max+j-1;
    end
end
output_args.LabelMatrix=cyto_lbl;

%end segmentObjectsUsingMarkers
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% selectBlobButtonPressed.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function selectBlobButtonPressed(hObject,eventdata,handles)
%helper function for manual segmentation review module. handler for the
%select button
global msr_gui_struct;
button_value=get(hObject,'Value');
objects_lbl=msr_gui_struct.ObjectsLabel;
image_data=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
set(msr_gui_struct.ImageHandle,'CData',image_data);
if (button_value==1)
    %toggle select object button off
    set(msr_gui_struct.SelectObjectButtonHandle,'Value',0);
    %disable object action buttons
    set(msr_gui_struct.JoinObjectsButtonHandle,'Enable','off');
    set(msr_gui_struct.RemoveObjectButtonHandle,'Enable','off');
    %enable blob action buttons
    set(msr_gui_struct.ResegmentBlobButtonHandle,'Enable','on');
    set(msr_gui_struct.RemoveBlobButtonHandle,'Enable','on');
```

```
    set(msr_gui_struct.RestoreBlobButtonHandle,'Enable','on');
    updateReviewSegGUIStatus('SelectBlob');
    msr_gui_struct.SelectedObjectID=[];
else
    %disable blob action buttons
    set(msr_gui_struct.ResegmentBlobButtonHandle,'Enable','off');
    set(msr_gui_struct.RemoveBlobButtonHandle,'Enable','off');
    set(msr_gui_struct.RestoreBlobButtonHandle,'Enable','off');
    msr_gui_struct.CurrentAction='';
    msr_gui_struct.SelectedBlobID=[];
    msr_gui_struct.SelectedObjectID=[];
    updateReviewSegGUIStatus('InitialStatus');
end

%end selectBlobButtonPressed
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% selectBlobByID.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function selectBlobByID(blob_id)
%blob id may contain multiple blob ids
global msr_gui_struct;

blobs_lbl=msr_gui_struct.BlobsLabel;
image_handle=msr_gui_struct.ImageHandle;
objects_lbl=msr_gui_struct.ObjectsLabel;
if (blob_id==0)
    warnDlg('You clicked on the background!');
    return;
end

cur_blob=ismember(blobs_lbl,blob_id);
blob_mask=repmat(cur_blob,[1 1 3]);

if (msr_gui_struct.SelectMultiple)
    selected_blob_ids=msr_gui_struct.SelectedBlobID;
    cur_selected_idx=ismember(selected_blob_ids,blob_id);
    if (max(cur_selected_idx))
        %blob is already selected so unselect it
        selected_blob_ids(cur_selected_idx)=[];
        msr_gui_struct.SelectedBlobID=selected_blob_ids;

label_rgb=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
        image_data=get(image_handle,'CData');
        image_data(blob_mask)=label_rgb(blob_mask);
        set(image_handle,'CData',image_data);
    else
        image_data=get(image_handle,'CData');
        image_data(blob_mask)=createCheckerBoardPattern(cur_blob);
        set(image_handle,'CData',image_data);
        selected_blob_ids=[selected_blob_ids blob_id];
        msr_gui_struct.SelectedBlobID=selected_blob_ids;
    end
else
    image_data=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
    image_data(blob_mask)=createCheckerBoardPattern(cur_blob);
    set(image_handle,'CData',image_data);
    msr_gui_struct.SelectedBlobID=blob_id;
end


msr_gui_struct.SelectedObjectID=[];

%end selectBlobByID
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% selectCell.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function selectCell(cur_cell_lbl_id)
%helper function for manual tracking review module. used to select a cell
%in the GUI
global mtr_gui_struct;

cells_lbl=mtr_gui_struct.CellsLabel;
cur_cell=(cells_lbl==cur_cell_lbl_id);
max_pxl=intmax('uint8');
%check if we're dealing with a fragmented cell
```

```
cur_cell_lbl=bwlabeln(cur_cell);
if max(cur_cell_lbl(:)>1)
    cur_cell=joinFragmentedBlob(cur_cell);
end
mtr_gui_struct.CurCellBlob=cur_cell;

%end selectCell
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% selectionLayersGUI.m %%%%%%%%%%%%%%%%%%%%%%%%%%%
function varargout = selectionLayersGUI(varargin)
% SELECTIONLAYERSGUI M-file for selectionLayersGUI.fig
%      SELECTIONLAYERSGUI, by itself, creates a new SELECTIONLAYERSGUI or raises the existing
%      singleton*.
%
%      H = SELECTIONLAYERSGUI returns the handle to a new SELECTIONLAYERSGUI or the handle to
%      the existing singleton*.
%
%      SELECTIONLAYERSGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SELECTIONLAYERSGUI.M with the given input arguments.
%
%      SELECTIONLAYERSGUI('Property','Value',...) creates a new SELECTIONLAYERSGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before selectionLayersGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to selectionLayersGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help selectionLayersGUI

% Last Modified by GUIDE v2.5 26-Jul-2010 15:40:29

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @selectionLayersGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @selectionLayersGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before selectionLayersGUI is made visible.
function selectionLayersGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to selectionLayersGUI (see VARARGIN)

% Choose default command line output for selectionLayersGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes selectionLayersGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = selectionLayersGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in listboxSelectionLayers.
function listboxSelectionLayers_Callback(hObject, eventdata, handles)
% hObject    handle to listboxSelectionLayers (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listboxSelectionLayers contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from listboxSelectionLayers


% --- Executes during object creation, after setting all properties.
function listboxSelectionLayers_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listboxSelectionLayers (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in buttonAddLayer.
function buttonAddLayer_Callback(hObject, eventdata, handles)
% hObject    handle to buttonAddLayer (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
addLayer();


% --- Executes on button press in buttonRemoveLayer.
function buttonRemoveLayer_Callback(hObject, eventdata, handles)
% hObject    handle to buttonRemoveLayer (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
removeLayer();

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% selectObjectButtonPressed.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function selectObjectButtonPressed(hObject,eventdata,handles)
%helper function for manual segmentation review. select object button
%handler.
global msr_gui_struct;
button_value=get(hObject,'Value');
objects_lbl=msr_gui_struct.ObjectsLabel;
image_data=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
set(msr_gui_struct.ImageHandle,'CData',image_data);
if (button_value==1)
    %toggle select blob button off
    set(msr_gui_struct.SelectBlobButtonHandle,'Value',0);
    %disable blob action buttons
    set(msr_gui_struct.ResegmentBlobButtonHandle,'Enable','off');
    set(msr_gui_struct.RemoveBlobButtonHandle,'Enable','off');
    set(msr_gui_struct.RestoreBlobButtonHandle,'Enable','off');
    %enable object action buttons
    set(msr_gui_struct.JoinObjectsButtonHandle,'Enable','on');
    set(msr_gui_struct.RemoveObjectButtonHandle,'Enable','on');
    updateReviewSegGUIStatus('SelectObject');
    msr_gui_struct.SelectedBlobID=[];
else
    %disable object action buttons
    set(msr_gui_struct.JoinObjectsButtonHandle,'Enable','off');
    set(msr_gui_struct.RemoveObjectButtonHandle,'Enable','off');
    msr_gui_struct.CurrentAction='';
    msr_gui_struct.SelectedBlobID=[];
    msr_gui_struct.SelectedObjectID=[];
    updateReviewSegGUIStatus('InitialStatus');
end
```

```
%end selectBlobButtonPressed
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% selectObjectByID.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function selectObjectByID(obj_id)
%helper function for manual segmentation review. select the object with
%id=obj_id
global msr_gui_struct;

objects_lbl=msr_gui_struct.ObjectsLabel;
image_handle=msr_gui_struct.ImageHandle;

if (obj_id==0)
    msr_gui_struct.SelectedObjectID=[];
    image_data=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
    set(image_handle,'CData',image_data);
    warndlg('You clicked on the background!');
    return;
end

cur_obj=ismember(objects_lbl,obj_id);
obj_mask=repmat(cur_obj,[1 1 3]);

if (msr_gui_struct.SelectMultiple)
    selected_obj_ids=msr_gui_struct.SelectedObjectID;
    cur_selected_idx=ismember(selected_obj_ids,obj_id);
    if (max(cur_selected_idx))
        %blob is already selected so unselect it
        selected_obj_ids(cur_selected_idx)=[];
        msr_gui_struct.SelectedObjectID=selected_obj_ids;

label_rgb=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
        image_data=get(image_handle,'CData');
        image_data(obj_mask)=label_rgb(obj_mask);
        set(image_handle,'CData',image_data);
    else
        image_data=get(image_handle,'CData');
        image_data(obj_mask)=createCheckerBoardPattern(cur_obj);
        set(image_handle,'CData',image_data);
        selected_obj_ids=[selected_obj_ids obj_id];
        msr_gui_struct.SelectedObjectID=selected_obj_ids;
    end
else
    image_data=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
    image_data(obj_mask)=createCheckerBoardPattern(cur_obj);
    set(image_handle,'CData',image_data);
    msr_gui_struct.SelectedObjectID=obj_id;
end


msr_gui_struct.SelectedBlobID=[];



%end selectObjectByID
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% setArrayVar.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=setArrayVar(input_args)
% Usage
% This module is used to set a set of values in an array.
% Input Structure Members
% Array - The array where the values will be entered.
% Index - The index in the array where the values will be entered.
% Var - The set of values that will be entered in the array.
% Output Structure Members
% Array - The array with the new set of values.

array=input_args.Array.Value;
array(input_args.Index.Value,:)=input_args.Var.Value;
output_args.Array=array;

%end setArrayVar
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% setGroupIndex.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=setGroupIndex(input_args)
%helper function for CA tracking module. sets the matching group index for
%the current object
shape_parameters=input_args.ShapeParameters.Value;
shape_parameters(input_args.CellID.Value,input_args.GroupIDCol.Value-
input_args.AreaCol.Value+1)=input_args.GroupIndex.Value;
output_args.ShapeParameters=shape_parameters;

%end setGroupIndex
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% showArgValue.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function arg_type=showArgValue(arg_text,provider_text,module_struct,handles)
%update the inputArgumentsGUI with the current value of the argument
%determine if selected value it's manual or output
%extract the argument name and provider
search_pattern='>*(\w*)<*';
arg_name=regexp(arg_text,search_pattern,'once','tokens');
search_pattern='> ([a-zA-Z]*)\d*<';
provider_type=regexp(provider_text,search_pattern,'once','tokens');
j=1;
if strcmp(provider_type,'Value')
    %display a manually set value
    %disable the module output boxes
    arg_type='Manual';
    arg_idx=regexp(provider_text,'Value([0-9]*)','once','tokens');
    arg_idx=str2double(arg_idx{1});
    set(handles.popupOutputArgument,'Enable','off');
    set(handles.popupModuleInstance,'Enable','off');
    %enable the manual edit box
    set(handles.editManualValue,'Enable','on');
    static_params=module_struct.StaticParameters;
    for i=1:length(static_params)
        cur_val=static_params{i};
        if strcmp(arg_name,cur_val{1})
            if (arg_idx~=j)
                j=j+1;
            else
                %found the argument show the value
                set(handles.editManualValue,'String',cur_val{2});
                break;
            end
        end
    end
else
    %display a value obtained as an output arg from a module
    %disable the manual edit box
    arg_type='Output';
    arg_idx=regexp(provider_text,'Output([0-9]*)','once','tokens');
    arg_idx=str2double(arg_idx{1});
    %enable the module output boxes
    set(handles.popupOutputArgument,'Enable','on');
    set(handles.popupModuleInstance,'Enable','on');
    output_params=module_struct.OutputArgs;
    for i=1:length(output_params)
        cur_val=output_params{i};
        if strcmp(arg_name,cur_val{1})
            if (arg_idx~=j)
                j=j+1;
            else
                %found the argument update the popup lists
                module_id=cur_val{2};
                modules_list=get(handles.popupModuleInstance,'String');
                module_idx=find(strcmp(module_id,modules_list));
                %update the popup selection
                set(handles.popupModuleInstance,'Value',module_idx);
                %update the args popup
                updateOutputArgPopup(handles);
                args_list=get(handles.popupOutputArgument,'String');
                arg_name=cur_val{3};
                arg_name=arg_name(2:(end-1));
                arg_idx=find(strcmp(arg_name,args_list));
                %update args popup selection
```

```
                set(handles.popupOutputArgument,'Value',arg_idx);
                break;
            end
        end
    end
    set(handles.editManualValue,'Enable','off');
end

guidata(handles.figure1,handles);
%end showArgValue
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% showImage.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function showImage()
%helper function for manual segmentation review module. display the image
%of the current label matrix
global msr_gui_struct;
show_image=get(msr_gui_struct.CheckBoxImageHandle,'Value');
if (show_image)
    %toggle select blob button off
    set(msr_gui_struct.SelectBlobButtonHandle,'Enable','off');
    %toggle select object button off
    set(msr_gui_struct.SelectObjectButtonHandle,'Enable','off');
    %disable blob action buttons
    set(msr_gui_struct.ResegmentBlobButtonHandle,'Enable','off');
    set(msr_gui_struct.RemoveBlobButtonHandle,'Enable','off');
    set(msr_gui_struct.RestoreBlobButtonHandle,'Enable','off');
    %disable object action buttons
    set(msr_gui_struct.JoinObjectsButtonHandle,'Enable','off');
    set(msr_gui_struct.RemoveObjectButtonHandle,'Enable','off');
    %disable show raw label checkbox
    set(msr_gui_struct.CheckBoxRawLabelHandle,'Enable','off');
    %disable show previous label checkbox
    set(msr_gui_struct.CheckBoxPrevLabelHandle,'Enable','off');
    %save selections
    msr_gui_struct.SavedAction=msr_gui_struct.CurrentAction;
    field_names=fieldnames(msr_gui_struct);
    if (max(strcmp(field_names,'SelectedBlobID')))
        msr_gui_struct.SavedBlobID=msr_gui_struct.SelectedBlobID;
    else
        msr_gui_struct.SavedBlobID=[];
    end
    if (max(strcmp(field_names,'SavedObjectID')))
        msr_gui_struct.SavedObjectID=msr_gui_struct.SelectedObjectID;
    else
        msr_gui_struct.SavedObjectID=[];
    end
    msr_gui_struct.CurrentAction='ShowImage';
    msr_gui_struct.SelectedBlobID=[];
    msr_gui_struct.SelectedObjectID=[];
    %show previous label
    colormap(gray);
    msr_gui_struct.ImageHandle=imagesc(msr_gui_struct.Image,'Parent',msr_gui_struct.AxesHandle);
else
    cur_action=msr_gui_struct.SavedAction;
    msr_gui_struct.CurrentAction=cur_action;
    %toggle select blob button on
    set(msr_gui_struct.SelectBlobButtonHandle,'Enable','on');
    %toggle select object button on
    set(msr_gui_struct.SelectObjectButtonHandle,'Enable','on');
    %enable show raw label checkbox
    set(msr_gui_struct.CheckBoxRawLabelHandle,'Enable','on');
    %enable show raw label checkbox
    set(msr_gui_struct.CheckBoxPrevLabelHandle,'Enable','on');
    %show current label
    objects_lbl=msr_gui_struct.ObjectsLabel;

objects_lbl_rgb=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
    msr_gui_struct.ImageHandle=image(objects_lbl_rgb,'Parent',msr_gui_struct.AxesHandle);
    msr_gui_struct.CurrentAction=msr_gui_struct.SavedAction;
    blob_id=msr_gui_struct.SavedBlobID;
    switch(cur_action)
        case 'SelectBlob'
            %enable blob action buttons
            set(msr_gui_struct.ResegmentBlobButtonHandle,'Enable','on');
            set(msr_gui_struct.RemoveBlobButtonHandle,'Enable','on');
            set(msr_gui_struct.RestoreBlobButtonHandle,'Enable','on');
            updateReviewSegGUIStatus('SelectBlob');
```

```
        case 'SelectObject'
            %enable object action buttons
            set(msr_gui_struct.JoinObjectsButtonHandle,'Enable','on');
            set(msr_gui_struct.RemoveObjectButtonHandle,'Enable','on');
            updateReviewSegGUIStatus('SelectObject');
    end
    if (~isempty(blob_id))
        selectBlobByID(blob_id);
    end
    obj_id=msr_gui_struct.SavedObjectID;
    if (~isempty(obj_id))
        msr_gui_struct.SelectedObjectID=obj_id;
        selectObjectByID(obj_id);
    end
    msr_gui_struct.SavedAction=[];
    msr_gui_struct.SavedBlobID=[];
    msr_gui_struct.SavedObjectID=[];
    %set the function handle for a mouse click in the objects image
    set(msr_gui_struct.ImageHandle,'buttondownfcn','mouseClickInLabel');
    set(msr_gui_struct.GuiHandle,'KeyPressFcn','keyPressInManualSegmentationGUI');
end

%end showImage
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% showImageAndPause.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=showImageAndPause(input_args)
% Usage
% This module is used to show an image and pause execution.
% Input Structure Members
% FigureNr – The handle number of the MATLAB figure. If it doesn't exist it will be created.
% Image – Matrix containing the image to be shown.
% Output Structure Members
% None.


showmaxfigure(input_args.FigureNr.Value), imshow(input_args.Image.Value);
pause;
output_args=[];

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% showLabelMatrix.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=showLabelMatrix(input_args)
% Usage
% This module is used to show a MATLAB label matrix and pause execution.
% Input Structure Members
% FigureNr – The handle number of the MATLAB figure. If it doesn't exist it will be created.
% LabelMatrix – The label matrix to be displayed.
% Output Structure Members
% None.

showmaxfigure(input_args.FigureNr.Value), imshow(label2rgb(input_args.LabelMatrix.Value));
output_args=[];

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% showLabelMatrixAndPause.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=showLabelMatrixAndPause(input_args)
% Usage
% This module is used to show a MATLAB label matrix and pause execution.
% Input Structure Members
% FigureNr – The handle number of the MATLAB figure. If it doesn't exist it will be created.
% LabelMatrix – The label matrix to be displayed.
% Output Structure Members
% None.

showmaxfigure(input_args.FigureNr.Value), imshow(label2rgb(input_args.LabelMatrix.Value));
pause;
output_args=[];
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% showmaxfigure.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function figure_handle=showmaxfigure(figure_nr)
%show figure maximized and with no annoying borders
figure_handle=figure(figure_nr);
set(figure_handle,'PaperPositionMode','auto');
subplot('Position', [0 0 1 1]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% showPreviousLabel.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function showPreviousLabel()
%helper function for manual segmentation review. overlay the previous label
%image on the current image label
global msr_gui_struct;
show_prev_label=get(msr_gui_struct.CheckBoxPrevLabelHandle,'Value');
if (show_prev_label)
    %toggle select blob button off
    set(msr_gui_struct.SelectBlobButtonHandle,'Enable','off');
    %toggle select object button off
    set(msr_gui_struct.SelectObjectButtonHandle,'Enable','off');
    %disable blob action buttons
    set(msr_gui_struct.ResegmentBlobButtonHandle,'Enable','off');
    set(msr_gui_struct.RemoveBlobButtonHandle,'Enable','off');
    set(msr_gui_struct.RestoreBlobButtonHandle,'Enable','off');
    %disable object action buttons
    set(msr_gui_struct.JoinObjectsButtonHandle,'Enable','off');
    set(msr_gui_struct.RemoveObjectButtonHandle,'Enable','off');
    %disable show raw label checkbox
    set(msr_gui_struct.CheckBoxRawLabelHandle,'Enable','off');
    %disable show image checkbox
    set(msr_gui_struct.CheckBoxImageHandle,'Enable','off');
    %remove selections
    msr_gui_struct.CurrentAction='ShowPreviousLabel';
    msr_gui_struct.SelectedBlobID=[];
    msr_gui_struct.SelectedObjectID=[];
    %show previous label
    prev_label=msr_gui_struct.PreviousLabel;

prev_label_rgb=label2rgb(prev_label,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
    msr_gui_struct.ImageHandle=image(prev_label_rgb,'Parent',msr_gui_struct.AxesHandle);
else
    %toggle select blob button on
    set(msr_gui_struct.SelectBlobButtonHandle,'Enable','on');
    %toggle select object button on
    set(msr_gui_struct.SelectObjectButtonHandle,'Enable','on');
    %enable show raw label checkbox
    set(msr_gui_struct.CheckBoxRawLabelHandle,'Enable','on');
    %enable show image checkbox
    set(msr_gui_struct.CheckBoxImageHandle,'Enable','on');
    %show current label
    objects_lbl=msr_gui_struct.ObjectsLabel;

objects_lbl_rgb=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
    msr_gui_struct.ImageHandle=image(objects_lbl_rgb,'Parent',msr_gui_struct.AxesHandle);
    msr_gui_struct.CurrentAction='';
    %set the function handle for a mouse click in the objects image
    set(msr_gui_struct.ImageHandle,'buttondownfcn','mouseClickInLabel');
    set(msr_gui_struct.GuiHandle,'KeyPressFcn','keyPressInManualSegmentationGUI');
end

%end showPreviousLabel
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% showRawLabel.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function showRawLabel()
%helper function for manual segmentation review module. show the original
%label without any corrections
global msr_gui_struct;
show_raw_label=get(msr_gui_struct.CheckBoxRawLabelHandle,'Value');
if (show_raw_label)
```

```
    %toggle select blob button on
    set(msr_gui_struct.SelectBlobButtonHandle,'Enable','off');
    %toggle select object button on
    set(msr_gui_struct.SelectObjectButtonHandle,'Enable','off');
    %disable blob action buttons
    set(msr_gui_struct.ResegmentBlobButtonHandle,'Enable','off');
    set(msr_gui_struct.RemoveBlobButtonHandle,'Enable','off');
    set(msr_gui_struct.RestoreBlobButtonHandle,'Enable','off');
    %disable object action buttons
    set(msr_gui_struct.JoinObjectsButtonHandle,'Enable','off');
    set(msr_gui_struct.RemoveObjectButtonHandle,'Enable','off');
    %disable show previous label checkbox
    set(msr_gui_struct.CheckBoxPrevLabelHandle,'Enable','off');
    %disable show image checkbox
    set(msr_gui_struct.CheckBoxImageHandle,'Enable','off');
    %remove selections
    msr_gui_struct.CurrentAction='ShowRawLabel';
    msr_gui_struct.SelectedBlobID=[];
    msr_gui_struct.SelectedObjectID=[];
    %show previous label
    raw_label=msr_gui_struct.OriginalObjectsLabel;
    raw_label_rgb=label2rgb(raw_label,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
    msr_gui_struct.ImageHandle=image(raw_label_rgb,'Parent',msr_gui_struct.AxesHandle);
else
    %toggle select blob button off
    set(msr_gui_struct.SelectBlobButtonHandle,'Enable','on');
    %toggle select object button off
    set(msr_gui_struct.SelectObjectButtonHandle,'Enable','on');
    %enable show previous label checkbox
    set(msr_gui_struct.CheckBoxPrevLabelHandle,'Enable','on');
    %enable show image checkbox
    set(msr_gui_struct.CheckBoxImageHandle,'Enable','on');
    %show current label
    objects_lbl=msr_gui_struct.ObjectsLabel;

objects_lbl_rgb=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
    msr_gui_struct.ImageHandle=image(objects_lbl_rgb,'Parent',msr_gui_struct.AxesHandle);
    msr_gui_struct.CurrentAction='';
    %set the function handle for a mouse click in the objects image
    set(msr_gui_struct.ImageHandle,'buttondownfcn','mouseClickInLabel');
    set(msr_gui_struct.GuiHandle,'KeyPressFcn','keyPressInManualSegmentationGUI');
end

%end showPreviousLabel
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% sliderTracksEvent.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function sliderTracksEvent()
%manual tracking review module. track slider event handler
global mtr_gui_struct;

cur_frame=round(get(mtr_gui_struct.SliderHandle,'Value'));
mtr_gui_struct.CurFrame=cur_frame;
[mitotic_cells_ids other_new_cells_ids mitotic_events_nr]=getMitoticEventsIDs(cur_frame);
status_text=[' Frame: ' num2str(cur_frame) ' Mitotic Events Detected: '
num2str(mitotic_events_nr)];
set(mtr_gui_struct.StatusTextHandle,'String', status_text);
status_text=['New Cells From Split: ' num2str(mitotic_cells_ids')];
set(mtr_gui_struct.EditStatus1Handle,'String', status_text);
status_text=['Other New Cells In Frame: ' num2str(other_new_cells_ids')];
set(mtr_gui_struct.EditStatus2Handle,'String', status_text);
updateTrackImage(cur_frame,mtr_gui_struct.ShowLabels,mtr_gui_struct.ShowOutlines);
mtr_gui_struct.CurCentroids=getApproximateCentroids(mtr_gui_struct.CellsLabel);
addSelectionLayers();
displayFrameMSD();
if (mtr_gui_struct.SelectedCellID&&(cur_frame>=mtr_gui_struct.SelectedCellStart)...
        &&(cur_frame<=mtr_gui_struct.SelectedCellEnd))
    updateTrackRecord();
    mtr_gui_struct.SelectedCellLabelID=getCurLabelID();
    selectCell(mtr_gui_struct.SelectedCellLabelID);
    updateCellStatus();
end

%end sliderTracksEvent
end

function updateTrackRecord()
```

```
global mtr_gui_struct;

tracks_layout=mtr_gui_struct.TracksLayout;
tracks=mtr_gui_struct.Tracks;
cur_time=(mtr_gui_struct.CurFrame-1)*mtr_gui_struct.TimeFrame;
cur_tracks_idx=tracks(:,tracks_layout.TimeCol)==cur_time;
cur_tracks=tracks(cur_tracks_idx,:);
cur_cell_idx=cur_tracks(:,tracks_layout.TrackIDCol)==mtr_gui_struct.SelectedCellID;
cur_track_record=cur_tracks(cur_cell_idx,:);
mtr_gui_struct.CurrentTrackRecord=cur_track_record;
cell_speeds=mtr_gui_struct.CellSpeeds;
cur_speeds=cell_speeds(cur_tracks_idx,2);
cell_sq_disps=mtr_gui_struct.CellSquareDisplacements;
cur_sq_disps=cell_sq_disps(cur_tracks_idx,2);

mtr_gui_struct.CurrentSpeed=cur_speeds(cur_cell_idx);
mtr_gui_struct.CurrentSquareDisplacement=cur_sq_disps(cur_cell_idx);

%end updateTrackRecord
end

function [mitotic_cells_ids other_new_cells_ids mitotic_events_nr]=getMitoticEventsIDs(frame_nr)
global mtr_gui_struct;

ancestry_records=mtr_gui_struct.CellsAncestry;
ancestry_layout=mtr_gui_struct.AncestryLayout;
time_frame=mtr_gui_struct.TimeFrame;
cur_time=(frame_nr-1).*time_frame;
new_cells_idx=ancestry_records(:,ancestry_layout.StartTimeCol)==cur_time;
mitotic_events_idx=ancestry_records(:,ancestry_layout.ParentIDCol)>0;
mitotic_cells_ids=ancestry_records(new_cells_idx&mitotic_events_idx,ancestry_layout.TrackIDCol);
other_new_cells_ids=ancestry_records(new_cells_idx&(~mitotic_events_idx),ancestry_layout.TrackIDC
ol);
mitotic_events_nr=size(mitotic_cells_ids,1)/2;

%end getNumberOfMitoticEvents
end

function label_id=getCurLabelID()
global mtr_gui_struct;

tracks_layout=mtr_gui_struct.TracksLayout;
cur_track_record=mtr_gui_struct.CurrentTrackRecord;
if (isempty(cur_track_record))
    label_id=-1;
    return;
end
cell_centroid=cur_track_record(:,tracks_layout.Centroid1Col:tracks_layout.Centroid2Col);
cur_centroids=mtr_gui_struct.CurCentroids;
cur_dist=hypot(cell_centroid(1)-cur_centroids(:,1),cell_centroid(2)-cur_centroids(:,2));
[dummy label_id]=min(cur_dist);

%end getCurLabelID
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% solidityFilter.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=solidityFilter(input_args)
% Usage
% This module is used to remove objects below or above a threshold solidity from a binary image.
% Input Structure Members
% Image - The binary image from which objects will be removed.
% MaxSolidity - Objects whose solidity is above this value will be removed from the image.
% MinSolidity - Objects whose solidity is below this value will be removed from the image.
% Output Structure Members
% Image - The filtered binary image.

cells_lbl=bwlabeln(input_args.Image.Value);
cells_props=regionprops(cells_lbl,'Solidity');
field_names=fieldnames(input_args);
if (max(strcmp(field_names,'MinSolidity')))
    b_min=true;
else
    b_min=false;
end
if (max(strcmp(field_names,'MaxSolidity')))
    b_max=true;
else
    b_max=false;
```

```
end
cells_solidity=[cells_props.Solidity];
cells_nr=length(cells_solidity);
valid_solidities_idx=false(1,cells_nr);
if (b_min)
    valid_solidities_idx=valid_solidities_idx|(cells_solidity>=input_args.MinSolidity.Value);
end
if (b_max)
    valid_solidities_idx=valid_solidities_idx|(cells_solidity<=input_args.MaxSolidity.Value);
end

output_args.Image=ismember(cells_lbl,find(valid_solidities_idx));

%end solidityFilter
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% solidityFilterLabel.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=solidityFilterLabel(input_args)
% Usage
% This module is used to remove objects below or above a threshold solidity value from a MATLAB
label matrix.
% Input Structure Members
% ObjectsLabel - The label matrix from which objects will be removed.
% MaxSolidity - Objects whose solidity is above this value will be removed from the image.
% MinSolidity - Objects whose solidity is below this value will be removed from the image.
% Output Structure Members
% LabelMatrix - The filtered label matrix.

cells_lbl=input_args.ObjectsLabel.Value;
cells_props=regionprops(cells_lbl,'Solidity');
field_names=fieldnames(input_args);
if (max(strcmp(field_names,'MinSolidity')))
    b_min=true;
else
    b_min=false;
end
if (max(strcmp(field_names,'MaxSolidity')))
    b_max=true;
else
    b_max=false;
end
cells_solidity=[cells_props.Solidity];
cells_nr=length(cells_solidity);
valid_solidities_idx=false(1,cells_nr);
if (b_min)
    valid_solidities_idx=valid_solidities_idx|(cells_solidity>=input_args.MinSolidity.Value);
end
if (b_max)
    valid_solidities_idx=valid_solidities_idx|(cells_solidity<=input_args.MaxSolidity.Value);
end
if (min(valid_solidities_idx)==1)
    %no invalid objects return the same label back
    output_args.LabelMatrix=cells_lbl;
else
    valid_object_numbers=find(valid_solidities_idx);
    new_object_numbers=1:length(valid_object_numbers);
    %we will replace valid numbers with new and everything else will be set to
    %zero
    object_idx=cells_lbl>0;
    new_object_index=zeros(max(cells_lbl(object_idx)),1);
    new_object_index(valid_object_numbers)=new_object_numbers;
    new_cells_lbl=cells_lbl;
    %replace the old object numbers to prevent skips in numbering
    new_cells_lbl(object_idx)=new_object_index(cells_lbl(object_idx));
    output_args.LabelMatrix=new_cells_lbl;
end

%end solidityFilter
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% sortManyToOneUsingPairs.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function [many_params_sorted sort_idx]=sortManyToOneUsingPairs(single_params,
many_params,b_use_direction,unknown_param_weights,...
```

```
    param_weights,ranking_order,matching_group_idx,relevant_params_idx)
%helper function for CA tracking algorithm
%sort tracks by how well they match a particular cell or cells by how well
%they match a particular track. look at one pair at a time so each
%track/cell is pitted against another individually otherwise the best
%matching track/cell might not be picked. when ranking all the
%cells/tracks at once we have situations where the cell/tracks that has the
%most parameters matching is not the best match due to other poorly
%matching cells/tracks robbing parameter matches from the true best matching cell. by
%matching them head to head this situation is avoided.
many_nr=size(many_params,1);
many_params_sorted=many_params;
sort_idx=[1:many_nr]';
%can't do a straight insertion sort or any sorting algorithm that doesn't
%compare each element with every other element because for track rankings
%a<b and b<c does not imply a<c
for i=1:many_nr-1
    for j=i:many_nr
        param1=many_params_sorted(j,:);
        sort1=sort_idx(j);
        b_smallest=true;
        for k=i:many_nr
            if (j==k)
                continue;
            end
            param2=many_params_sorted(k,:);

many_scores=getPairScoresToSingle([param1;param2],single_params,b_use_direction,unknown_param_wei
ghts,...
                param_weights,ranking_order,matching_group_idx,relevant_params_idx);
            if (many_scores(2)<many_scores(1))
                %this cannot be the smallest element
                b_smallest=false;
                break;
            end
        end
        if (b_smallest)
            if (i~=j)
                param2=many_params_sorted(i,:);
                sort2=sort_idx(i);
                many_params_sorted(i,:)=param1;
                sort_idx(i)=sort1;
                many_params_sorted(j,:)=param2;
                sort_idx(j)=sort2;
            end
            break;
        end
    end
end

%end sortManyToOneUsingPairs
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% sortTracks.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function [tracks_sorted sort_idx
track_scores]=sortTracks(track_ranks,tracks,ranking_order,params_weights)
%helper function for CA tracking module
%sort the tracks according to ranking order
nr_tracks=size(track_ranks,1);
if (nr_tracks==1)
    tracks_sorted=tracks;
    sort_idx=1;
    track_scores=1;
    return;
end
track_ranks_by_relevance=track_ranks(:,ranking_order);
weighted_track_ranks=track_ranks_by_relevance.*repmat(params_weights,nr_tracks,1);
track_scores=sum(weighted_track_ranks,2);
[dummy sort_idx]=sort(track_scores);
tracks_sorted=tracks(sort_idx,:);

%end sortTracks
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% splitTracks.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=splitTracks(input_args)
%helper function for ancestry module
%split tracks where we detected mitotic events and record cell ancestry

split_cells=input_args.SplitCells.Value;
tracks=input_args.Tracks.Value;
cells_ancestry=input_args.CellsAncestry.Value;
if isempty(split_cells)
    output_args.Tracks=tracks;
    output_args.CellsAncestry=cells_ancestry;
    return;
end

tracks_layout=input_args.TracksLayout.Value;
trackIDCol=tracks_layout.TrackIDCol;
timeCol=tracks_layout.TimeCol;
max_track_id=max(tracks(:,trackIDCol));
ancestry_layout=input_args.AncestryLayout.Value;
stopTimeCol=ancestry_layout.StopTimeCol;
generationCol=ancestry_layout.GenerationCol;
ancestryIDCol=ancestry_layout.TrackIDCol;
time_frame=input_args.TimeFrame.Value;

if (~isempty(split_cells))
    [dummy sort_idx]=sort(split_cells(:,3));
    split_cells=split_cells(sort_idx,:);
    split_cells_len=size(split_cells,1);
else
    split_cells_len=0;
end


for i=1:split_cells_len
    %get the parent track id
    parent_track_id=split_cells(i,1);
    %get the split time
    split_time=split_cells(i,3);
    %check if parent track has already been split
    parent_ancestry_idx=(cells_ancestry(:,ancestryIDCol)==parent_track_id);
    parent_track_stop_time=cells_ancestry(parent_ancestry_idx,stopTimeCol);
    parent_track_generation=cells_ancestry(parent_ancestry_idx,generationCol);
    if (parent_track_stop_time>=split_time)
        %parent track needs to be split
        new_track_id=max_track_id+1;
        max_track_id=new_track_id;
        %set the new end time for the parent track
        new_stop_time=split_time-time_frame;
        cells_ancestry(parent_ancestry_idx,stopTimeCol)=new_stop_time;
        %update the parent track id after the split with the new track id
        new_track_idx=(tracks(:,trackIDCol)==parent_track_id)&(tracks(:,timeCol)>new_stop_time);
        tracks(new_track_idx,trackIDCol)=new_track_id;
        %create an ancestry record for the new track
        cells_ancestry=[cells_ancestry; ...
            [new_track_id parent_track_id parent_track_generation+1 split_time
parent_track_stop_time]];
    end
    %add an ancestry record for the other daughter cell resulting from the split
    daughter_id=split_cells(i,2);
    daughter_stop_time=split_cells(i,4);
    cells_ancestry=[cells_ancestry; ...
        [daughter_id parent_track_id parent_track_generation+1 split_time daughter_stop_time]];
end

output_args.Tracks=tracks;
output_args.CellsAncestry=cells_ancestry;

%end splitTracks
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% startTracks.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=startTracks(input_args)
% Usage
% This module is used to start a tracks matrix.
% Input Structure Members
% CellsLabel - The label matrix identifying the objects in the first frame of the time-lapse.
% CurFrame - The index value of the current frame.
```

```
% ShapeParameters - The shape parameters for the objects in the first frame (Area, Eccentricity,
etc.).
% TimeFrame - The time interval between consecutive frames in the time-lapse.
% Output Structure Members
% Tracks - The new tracks matrix.

cells_props=regionprops(input_args.CellsLabel.Value,'Centroid');
cells_centroids=[cells_props.Centroid]';
cells_centroids_2=cells_centroids(1:2:length(cells_centroids));
cells_centroids_1=cells_centroids(2:2:length(cells_centroids));
cur_time=repmat((input_args.CurFrame.Value-
1)*input_args.TimeFrame.Value,size(cells_centroids_1,1),1);
track_ids=[1:size(cells_centroids_1,1)]';
output_args.Tracks=[track_ids cur_time cells_centroids_1 cells_centroids_2
input_args.ShapeParameters.Value];
output_args.PotentialCellParams=[];
output_args.CellsLabel=input_args.CellsLabel.Value;
output_args.MatchingGroups=[];

%end startTracks
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% startTracks3D.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=startTracks3D(input_args)
% Usage
% This module is used to start a tracks matrix.
% Input Structure Members
% ObjectCentroids - The centroids of the objects in the label matrix.
% CurFrame - The index value of the current frame.
% ShapeParameters - Any additional parameters to be added to the tracks
% matrix
% TimeFrame - The time interval between consecutive frames in the time-lapse.
% Output Structure Members
% Tracks - The new tracks matrix.

object_centroids=input_args.ObjectCentroids.Value;
cur_time=repmat((input_args.CurFrame.Value-
1)*input_args.TimeFrame.Value,size(object_centroids,1),1);
track_ids=[1:size(object_centroids,1)]';
output_args.Tracks=[track_ids cur_time object_centroids input_args.ShapeParameters.Value];

%end startTracks
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% stripHTMLFromString.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function stripped_text=stripHTMLFromString(html_string)
%helper function for assayEditorGUI. get a module ID from a listbox html string
%remove the html code
search_pattern='<html>(?:<\w>)*(?: )*(\w*)(?:</\w>)*</html>';
stripped_text=regexp(html_string,search_pattern,'once','tokens');
stripped_text=stripped_text{1};

%end getIDFromString
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% subtractFunction.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=subtractFunction(input_args)
%Usage
%This module subtracts one variable from another.
%
%Input Structure Members
%Number1 - The first variable.
%Number2 - The variable to be subtracted.
%
%Output Structure Members
%Difference - The result of the subtraction.


arg_1=input_args.Number1.Value;
arg_2=input_args.Number2.Value;
```

```
output_args.Difference=(arg_1-arg_2);

%end addFunction
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% switchTracks.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function switchTracks()
%helper for manual tracking review module. initialize switch track
%process
global mtr_gui_struct;

mtr_gui_struct.SwitchTrackID=mtr_gui_struct.SelectedCellID;
mtr_gui_struct.SwitchTrackRecord=mtr_gui_struct.CurrentTrackRecord;
mtr_gui_struct.SwitchTrackAncestry=mtr_gui_struct.CurrentAncestryRecord;
mtr_gui_struct.SwitchTrack=true;

%end switchTracks
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% undoAllChanges.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function undoAllChanges(hObject, eventdata, handles)
%helper for manual segmentation review module. undo all changes to the
%current label matrix
global msr_gui_struct;

button_pressed=questdlg('Are you sure you want to undo all the changes?',...
    'Undo Segmentation Changes','Yes','No','No');
switch(button_pressed)
    case 'Yes'
        objects_lbl=msr_gui_struct.OriginalObjectsLabel;
        msr_gui_struct.ObjectsLabel=objects_lbl;
        msr_gui_struct.BlobsLabel=bwlabeln(objects_lbl);
        msr_gui_struct.ErrorTypes=[];
        msr_gui_struct.ErrorBlobIDs=[];
        msr_gui_struct.TotalErrors=0;
        image_handle=msr_gui_struct.ImageHandle;

image_data=label2rgb(objects_lbl,msr_gui_struct.ColorMap,msr_gui_struct.BkgColor,'shuffle');
        set(image_handle,'CData',image_data);
    case 'No'
        return;
end

%end undoAllChanges
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% updateArg.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function []=updateArg(instance_name,update_name,var_name,var_val)
%CellAnimation core function
%update input parameters of dependent functions for the variable var_name only
global dependencies_list;
global dependencies_index;

cur_idx=dependencies_index.get(instance_name);
dependency_item=dependencies_list{cur_idx};
dependent_functions=dependency_item.DependentFunctions;
for i=1:size(dependent_functions,1)
    cur_function=dependent_functions{i};
    function_instance=cur_function.InstanceName;
    function_idx=dependencies_index.get(function_instance);
    dependent_args=cur_function.DependentArgs;
    for j=1:size(dependent_args,1)
        args_struct=dependent_args{j};
        args_fields=fieldnames(args_struct);
        switch(update_name)
            case {'input'}
                if max(strcmp(args_fields,'InputArg'))
                    this_arg_name=args_struct.InputArg;
                    update_type=2;
                else
                    continue;
```

```
                end
            case {'output'}
                if max(strcmp(args_fields,'OutputArg'))
                    this_arg_name=args_struct.OutputArg;
                    update_type=1;
                else
                    continue;
                end
        end
        if (~strcmp(var_name,this_arg_name))
            continue;
        end
        if (args_struct.Type~=update_type)
            continue;
        end
        dependent_arg_name=args_struct.ArgumentName;
        if (args_struct.DependencyType==1)
            dependencies_list{function_idx}.FunctionArgs.(dependent_arg_name).Value=var_val;
        elseif (args_struct.DependencyType==2)
            dependencies_list{function_idx}.KeepValues.(dependent_arg_name).Value=var_val;
        end
    end
end

%end updateArgs
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% updateArgs.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function []=updateArgs(instance_name,function_struct,update_type)
%CellAnimation core function
%update input parameters of dependent functions using either input or
%output parameters of function
global dependencies_list;
global dependencies_index;

if isempty(function_struct)
    return;
end

cur_idx=dependencies_index.get(instance_name);
dependency_item=dependencies_list{cur_idx};
dependent_functions=dependency_item.DependentFunctions;
for i=1:size(dependent_functions,1)
    cur_function=dependent_functions{i};
    function_instance=cur_function.InstanceName;
    function_idx=dependencies_index.get(function_instance);
    dependent_args=cur_function.DependentArgs;
    for j=1:size(dependent_args,1)
        args_struct=dependent_args{j};
        arg_name=args_struct.ArgumentName;
        switch(update_type)
            case('input')
                if (args_struct.Type==2)
                    input_arg_name=args_struct.InputArg;
                    output_fun_field_names=fieldnames(function_struct.(input_arg_name));
                    if (max(strcmp(output_fun_field_names,'Value'))==0)
                        %the output value was not provided by this module
                        %another module may provide it (such is the case in
                        %an if-else branch for example
                        continue;
                    end
                    if (args_struct.DependencyType==1)

dependencies_list{function_idx}.FunctionArgs.(arg_name).Value=function_struct.(input_arg_name).Va
lue;
                    elseif (args_struct.DependencyType==2)

dependencies_list{function_idx}.KeepValues.(arg_name).Value=function_struct.(input_arg_name).Valu
e;
                    end
                end
            case('output')
                if (args_struct.Type==1)
                    output_arg_name=args_struct.OutputArg;
                    output_fun_field_names=fieldnames(function_struct);
                    if (max(strcmp(output_fun_field_names,output_arg_name))==0)
                        %the output value was not provided by this module
                        %another module may provide it (such is the case in
```

```
                        %an if-else branch for example
                        continue;
                    end
                    if (args_struct.DependencyType==1)

dependencies_list{function_idx}.FunctionArgs.(arg_name).Value=function_struct.(output_arg_name);
                    elseif (args_struct.DependencyType==2)

dependencies_list{function_idx}.KeepValues.(arg_name).Value=function_struct.(output_arg_name);
                    end
                end
        end
    end
end

%end updateArgs
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% updateArgValue.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function module_struct=updateArgValue(handles)
%helper function for assayEditorGUI. update manual argument value

if strcmp(handles.ArgType,'Output')
    module_struct=handles.ModuleStruct;
    return;
end
man_value=get(handles.editManualValue,'String');
assay_list=get(handles.listboxInputArgumens,'String');
selection_idx=handles.SelectionIndex;
selection_text=assay_list{selection_idx};
arg_name=selection_text;
%get the argument name
while strcmp(arg_name(1:9),'<html><i>')
    selection_idx=selection_idx-1;
    arg_name=assay_list{selection_idx};
    if length(arg_name)<9
        break;
    end
end
module_struct=handles.ModuleStruct;
arg_nr=regexp(selection_text,'([0-9])','tokens','once');
arg_nr=str2double(arg_nr{1});
%get the argument indices
static_idx=find(cellfun(@(x) strcmp(x{1},arg_name), module_struct.StaticParameters));
static_idx=static_idx(arg_nr);
%update the value
module_struct.StaticParameters{static_idx}{2}=man_value;

%end updateArgValue
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% updateCellStatus.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function []=updateCellStatus()
%helper function for the manual tracking review gui. used to update the properties of the
%selected cell

global mtr_gui_struct;

num_fmt='%1.2f';

ancestry_layout=mtr_gui_struct.AncestryLayout;
tracks_layout=mtr_gui_struct.TracksLayout;
cell_track_record=mtr_gui_struct.CurrentTrackRecord;
cell_id=mtr_gui_struct.SelectedCellID;
cell_ancestry_record=mtr_gui_struct.CurrentAncestryRecord;
status_text=['Cell ID: ' num2str(cell_id) ' \n'];
parent_id=cell_ancestry_record(ancestry_layout.ParentIDCol);
status_text=[status_text 'Parent ID: ' num2str(parent_id) ' \n'];
start_frame=(cell_ancestry_record(ancestry_layout.StartTimeCol)./mtr_gui_struct.TimeFrame)+1;
status_text=[status_text 'Start Frame: ' num2str(start_frame) ' \n'];
end_frame=(cell_ancestry_record(ancestry_layout.StopTimeCol)./mtr_gui_struct.TimeFrame)+1;
status_text=[status_text 'End Frame: ' num2str(end_frame) ' \n'];
cell_generation=cell_ancestry_record(ancestry_layout.GenerationCol);
status_text=[status_text 'Generation: ' num2str(cell_generation) ' \n'];
cell_area=cell_track_record(tracks_layout.AreaCol);
```

```
status_text=[status_text 'Area: ' num2str(cell_area,num_fmt) ' \n'];
cell_ecc=cell_track_record(tracks_layout.EccCol);
status_text=[status_text 'Eccentricity: ' num2str(cell_ecc,num_fmt) ' \n'];
cell_mal=cell_track_record(tracks_layout.MalCol);
status_text=[status_text 'Major Axis Length: ' num2str(cell_mal,num_fmt) ' \n'];
cell_mil=cell_track_record(tracks_layout.MilCol);
status_text=[status_text 'Minor Axis Length: ' num2str(cell_mil,num_fmt) ' \n'];
cell_ori=cell_track_record(tracks_layout.OriCol);
status_text=[status_text 'Orientation: ' num2str(cell_ori,num_fmt) ' \n'];
cell_per=cell_track_record(tracks_layout.PerCol);
status_text=[status_text 'Perimeter: ' num2str(cell_per,num_fmt) ' \n'];
cell_sol=cell_track_record(tracks_layout.SolCol);
status_text=[status_text 'Solidity: ' num2str(cell_sol,num_fmt) ' \n'];
status_text=sprintf(status_text);
cur_speed=mtr_gui_struct.CurrentSpeed;
status_text=[status_text 'Current Speed: ' num2str(cur_speed,num_fmt) ' \n'];
cell_speeds=mtr_gui_struct.CellSpeeds;
overall_speed=cell_speeds(cell_speeds(:,1)==cell_id,2);
%remove the start point where we have zero speed
overall_speed(1)=[];
mean_cell_speed=mean(overall_speed);
status_text=[status_text 'Mean Speed: ' num2str(mean_cell_speed,num_fmt) ' \n'];
status_text=sprintf(status_text);
cell_sq_disp=mtr_gui_struct.CurrentSquareDisplacement;
status_text=[status_text 'Square Displacement: ' num2str(cell_sq_disp,num_fmt) ' \n'];
status_text=sprintf(status_text);
set(mtr_gui_struct.EditCellStatusHandle,'String',status_text);

%end updateCellStatus
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% updateOutputArgPopup.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function updateOutputArgPopup(handles)
%helper function for assayEditorGUI. update the list of output args for the current module
%enable the popup boxes
%disable the manual edit box
set(handles.editManualValue,'Enable','on');
%enable the module output boxes
set(handles.popupOutputArgument,'Enable','on');
set(handles.popupModuleInstance,'Enable','on');
modules_list=get(handles.popupModuleInstance,'String');
selection_idx=get(handles.popupModuleInstance,'Value');
module_id=modules_list{selection_idx};
modules_map=handles.ModulesMap;
module_idx=modules_map.get(module_id);
modules_list=handles.ModulesList;
module_struct=modules_list{module_idx};
file_name=[module_struct.ModuleName '.m'];
output_args=getOutputArgs(file_name);
if ~isempty(output_args)
    set(handles.popupOutputArgument,'String',output_args);
    set(handles.popupOutputArgument,'Value',1);
end

%end updateOutputArgPopup
end

function output_args=getOutputArgs(module_path)
%read the output arguments from a module file
file_text=fileread(module_path);
%find the output args
search_pattern='output_args.(\w*)\s*=\s*';
output_tokens=regexp(file_text,search_pattern,'tokens');
%convert to cell array of strings
output_args=cellfun(@(x) x{1},output_tokens,'UniformOutput',false);
%remove duplicates
output_args=unique(output_args);

%end getOutputArgs
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% updateReviewSegGUIStatus.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function updateReviewSegGUIStatus(gui_status)
%helper function for manual segmentation review. update the GUI status bar
global msr_gui_struct;
```

```
status_text_handle=msr_gui_struct.StatusTextHandle;

switch (gui_status)
    case 'InitialStatus'
        msr_gui_struct.CurrentAction='';
        set(status_text_handle,'String','Click on "Select Blob" or "Select Object" to begin.');
    case 'JoinObjects'
        msr_gui_struct.CurrentAction='JoinObjects';
        set(status_text_handle,'String','Click on objects to be joined. Click again to remove
them from the join list. Type "d" when done.');
    case 'SelectBlob'
        msr_gui_struct.CurrentAction='SelectBlob';
        set(status_text_handle,'String','Click on a blob to select it.');
    case 'SelectObject'
        msr_gui_struct.CurrentAction='SelectObject';
        set(status_text_handle,'String','Click on an object to select it.');
    case 'ResegmentBlob'
        msr_gui_struct.CurrentAction='ResegmentBlob';
        set(status_text_handle,'String',...
            'Click on points inside an object (minimum one) then type "n" to select points inside
the next object. Type "d" when done.');
end


%end updateReviewSegGUIStatus
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% updateTrackImage.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function []=updateTrackImage(frame_nr,b_show_labels,b_show_outlines)
%helper function for manual tracking review module. display the new image
%and cell and population statistics
global mtr_gui_struct;

file_name_args.FileBase.Value=mtr_gui_struct.ImageFileBase;
file_name_args.NumberFmt.Value=mtr_gui_struct.NumberFormat;
file_name_args.FileExt.Value=mtr_gui_struct.ImgExt;
file_name_args.CurFrame.Value=frame_nr;
file_name_struct=makeImgFileName(file_name_args);

read_image_args.ImageName.Value=file_name_struct.FileName;
read_image_args.ImageChannel.Value='';
image_struct=readImage(read_image_args);

img=imadjust(image_struct.Image);

label_name_args.FileBase.Value=mtr_gui_struct.SegFileRoot;
label_name_args.CurFrame.Value=frame_nr;
label_name_args.NumberFmt.Value=mtr_gui_struct.NumberFormat;
label_name_args.FileExt.Value='.mat';
file_name_struct=makeImgFileName(label_name_args);

load_label_args.FileName.Value=file_name_struct.FileName;
label_struct=loadCellsLabel(load_label_args);

cur_tracks_args.CurFrame.Value=frame_nr;
cur_tracks_args.OffsetFrame.Value=0;
cur_tracks_args.TimeFrame.Value=mtr_gui_struct.TimeFrame;
cur_tracks_args.TimeCol.Value=mtr_gui_struct.TimeCol;
cur_tracks_args.TrackIDCol.Value=mtr_gui_struct.TrackIDCol;
cur_tracks_args.Tracks.Value=mtr_gui_struct.Tracks;
cur_tracks_args.MaxMissingFrames.Value=mtr_gui_struct.MaxMissingFrames;
cur_tracks_args.FrameStep.Value=mtr_gui_struct.FrameStep;
cur_tracks_struct=getCurrentTracks(cur_tracks_args);

overlay_ancestry_args.Image.Value=img;
overlay_ancestry_args.CurrentTracks.Value=cur_tracks_struct.Tracks;
overlay_ancestry_args.CellsLabel.Value=label_struct.LabelMatrix;
overlay_ancestry_args.CellsAncestry.Value=mtr_gui_struct.CellsAncestry;
overlay_ancestry_args.CurFrame.Value=frame_nr;
overlay_ancestry_args.ColorMap.Value=mtr_gui_struct.ColorMap;
overlay_ancestry_args.TracksLayout.Value=mtr_gui_struct.TracksLayout;
overlay_ancestry_args.AncestryLayout.Value=mtr_gui_struct.AncestryLayout;
overlay_ancestry_args.ShowLabels.Value=b_show_labels;
overlay_ancestry_args.ShowOutlines.Value=b_show_outlines;
overlay_ancestry_struct=overlayAncestry(overlay_ancestry_args);

%display objects image in the objectAxes
mtr_gui_struct.ImageData=overlay_ancestry_struct.Image;
```

```
mtr_gui_struct.ImageHandle=image(overlay_ancestry_struct.Image,'Parent',mtr_gui_struct.TracksHand
le);
%set the function handle for a mouse click in the objects image
set(mtr_gui_struct.ImageHandle,'buttondownfcn','mouseClickInTrackingFrame');
%update the cells label
mtr_gui_struct.CellsLabel=label_struct.LabelMatrix;
mtr_gui_struct.FrameTracks=cur_tracks_struct.Tracks;

%end updateTrackImage
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% validateModuleArgs.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function module_struct=validateModuleArgs(handles)
%helper function for assayEditorGUI. remove any parameters that refer to modules which are no
longer part of
%the assay
module_struct=handles.ModuleStruct;
output_modules=cellfun(@(x) x{2},module_struct.OutputArgs,'UniformOutput',false);
modules_names=cellfun(@(x) x.InstanceName,handles.ModulesList,'UniformOutput',false);
params_idx=ismember(output_modules,modules_names);
module_struct.OutputArgs(~params_idx)=[];

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% vertcat_Wrapper.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=vertcat_Wrapper(input_args)
%wrapper module for matlab vertcat function
%Input Structure Members
%Matrix1 - The first matrix to be joined.
%Matrix2 - The second matrix to be joined.
%Output Structure Members
%Matrix - The joined matrix.

output_args.Matrix=vertcat(input_args.Matrix1.Value, input_args.Matrix2.Value);

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% whileLoop.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_args=whileLoop(function_struct)
% Usage
% This module is used to loop through a series of modules. The modules that will be looped
through are listed in the LoopFunctions input structure member.
% Input Structure Members
% TestFunction - The module which will be used to determine how many times the loop will iterate
over the modules in LoopFunctions. This module needs to return a true/false value and be part of
the whileLoop's module loop functions.


global dependencies_list;
global dependencies_index;

instance_name=function_struct.InstanceName;
cur_idx=dependencies_index.get(instance_name);

%propagate any input args needed by the loop functions from outside
updateArgs(instance_name,function_struct.FunctionArgs,'input');
input_args=function_struct.FunctionArgs;
loop_functions=function_struct.LoopFunctions;

test_function=input_args.TestFunction;
test_names=fieldnames(test_function);
%propagate any updated input args to the loop functions
dependency_item=dependencies_list{cur_idx};
updateArgs(instance_name,dependency_item.FunctionArgs,'input');

if (max(strcmp('FunctionInstance',test_names))==1)
    test_function_instance=input_args.TestFunction.FunctionInstance;
    test_function_dependecy_idx=dependencies_index.get(test_function_instance);
    test_function_dependency_item=dependencies_list{test_function_dependecy_idx};
    test_function_handle=test_function_dependency_item.FunctionHandle;
    test_output_name=input_args.TestFunction.OutputArg;
    test_output=test_function_handle(test_function_dependency_item.FunctionArgs);
```

```
    while(test_output.(test_output_name))
        for j=1:size(loop_functions,1)
            loop_function_instance_name=loop_functions{j}.InstanceName;
            callFunction(loop_function_instance_name,false);
        end
        output_args=makeOutputStruct(function_struct);
        updateArgs(instance_name,output_args,'output');
        test_function_dependency_item=dependencies_list{test_function_dependecy_idx};
        test_output=test_function_handle(test_function_dependency_item.FunctionArgs);
    end
else
    %really this only makes sense for debugging purposes
    while(input_args.TestFunction.Value)
        for j=1:size(loop_functions,1)
            loop_function_instance_name=loop_functions{j}.InstanceName;
            callFunction(loop_function_instance_name,false);
        end
        output_args=makeOutputStruct(function_struct);
        updateArgs(instance_name,output_args,'output');
    end
end


for i=1:size(loop_functions,1)
    loop_function_instance_name=loop_functions{i}.InstanceName;
    clearArgs(loop_function_instance_name);
end


%end while
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% wrapFunction.m %%%%%%%%%%%%%%%%%%%%%%%%%%
function wrapFunction(handles)
%helper function for assayEditorGUI. wrap a MATLAB function in a CellAnimation wrapper so it can
be used as a
%module in a CA assay

%select the function
[file_name,path_name] = uigetfile('*.m','Select MATLAB function to wrap');
if ~file_name
    return;
end
%read the function text
fi=fopen([path_name file_name]);
tl=fgetl(fi);
while 1
    if (tl==-1)
        break;
    end
    %remove white spaces
    tl(isspace(tl))=[];
    if (length(tl)<1)||(tl(1)=='%')
        tl=fgetl(fi);
        continue;
    end
    if strfind(tl,'function')
        while strcmp(tl((end-2):end),'...')
            %line continues. read the entire statement
            cl=fgetl(fi);
            cl(isspace(cl))=[];
            tl=[tl(1:(end-3)) cl];
        end
        arg_names=regexp(tl,'\((.*)\)','tokens','once');
        if ~isempty(arg_names)
            break;
        end
    end
    tl=fgetl(fi);
end
fclose(fi);
if (tl==-1)
    warndlg('Could not locate the function definition. Function wrapper was not created.');
    return;
end
%find the function name
fun_name=regexp(tl,'=([^\(=]*)\(','tokens','once');
```

```
if isempty(fun_name)
    fun_name=regexp(tl,'function(\w*)\(','tokens','once');
    if isempty(fun_name)
        warndlg('Could not locate the function definition. Function wrapper was not created.');
        return;
    end
end
fun_text=['function output_args=' fun_name{1} '_Wrapper(input_args)' 10 ];
fun_text=[fun_text '%Wrapper module for the function ' fun_name{1} 10 10];
%get the input argument names
arg_list=regexp(arg_names{1},'([^,]*)','tokens');
fun_text=[fun_text addInputArgs(arg_list)];
output_args=regexp(tl,'\[([^\[\]]*)\]','tokens','once');
if isempty(output_args)
    %one or none output args
    output_args=regexp(tl,'function(\w*)=','tokens','once');
    args_list={output_args};
else
    args_list=regexp(output_args{1},'([^,]*)','tokens');
end

%add call to the wrapped function replace the function name with the file
%name
name_start=strfind(tl,[fun_name{1} '(']);
fun_text=[fun_text tl(9:(name_start-1)) file_name(1:(end-2))
tl((name_start+length(fun_name{1})):end) ';' 10];

if isempty(output_args)
    fun_text=[fun_text 'output_args=[];' 10];
else
    %populate the output args
    fun_text=[fun_text addOutputArgs(args_list)];
end

fun_text=[fun_text 10 'end'];

%write the wrapped function module to disk
[file_name path_name]=uiputfile('*.m','Save Module as:',[fun_name{1} '_Wrapper']);
if (file_name==0)
    return;
end
fid=fopen([path_name file_name],'wt');
fwrite(fid,fun_text);
fclose(fid);

if strcmp(path_name(1:(end-1)),pwd)
    %module saved in the current path - add it to available modules list
    set(handles.listboxAvailableModules,'String',getModuleList());
end


%end wrapFunction
end

function args_text=addOutputArgs(args_list)
%create the wrapper text for the input arguments
args_text=[];
for i=1:length(args_list)
    cur_arg=args_list{i}{1};
    args_text=[args_text 'output_args.' cur_arg '=' cur_arg ';' 10];
end

%end addInputArgs
end

function args_text=addInputArgs(arg_list)
%create the wrapper text for the input arguments
args_text=[];
for i=1:length(arg_list)
    cur_arg=arg_list{i}{1};
    args_text=[args_text cur_arg '=input_args.' cur_arg '.Value;' 10];
end

%end addInputArgs
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end module%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```