COGNITIVE MAP GENERATION FOR LOCALIZATION AND NAVIGATION FROM 3-D LASER SCAN IMAGES

By

Stephen Michael Gordon

Thesis
Submitted to the Faculty of the
Graduate School of Vanderbilt University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE
in
Electrical Engineering

May, 2005

Nashville, Tennessee

Approved:
Professor Kazuhiko Kawamura
Professor D. Mitch Wilkes

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I would like to first and foremost thank Dr. Kazuhiko Kawamura for his support and guidance as I have worked in the Center for Intelligent Systems (CIS) and as I have researched my thesis. In addition to Dr. Kawamura I would like to thank Dr. Mitch Wilkes for working with me and helping me to understand the tasks that were before me.

I would like to acknowledge and thank my fellow graduate students in the CIS: Will, Karla, Albert (who has since left), Paul, Katherine, and Jonathan, each of whom has helped me along the way.

Finally, I would like to thank Flo Fottrell whose tireless work in the CIS should never go unnoticed and to whom I personally owe a great debt of gratitude.

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Overview

The research presented in this master's thesis demonstrates the integration of a variety of different control systems onto a mobile robot as well as present a method for acquiring 3-D sensor input from a 2-D laser scanner. Once acquired this thesis will present methods for processing the 3-D data to detect landmarks in the environment. The sensor data collected, once processed, will be used to create a map of the environment that the robot can use for localization and navigation.

Mobile Robot Navigation History

The problem of enabling a mobile robot to autonomously navigate its environment is not new to robotics research. In fact, the problem has existed as long as mobile robots have existed. Researchers working on mobile robots have, since the beginning, used laser scanners to detect range information from the environment and to generate working world models that the robot could use.

One of the original mobile robots developed was Shakey [13], a mobile robot developed by researchers at Stanford Research Institute (SRI) and introduced in 1968. Shakey was outfitted with a camera for vision, a simple laser scanner, and bump sensors. Shakey had separate programs for perception, world modeling, and acting. These programs ran on a computer that was the size of a small room.

Figure 1: Shakey, a mobile robot developed in 1968

Since Shakey's development mobile robots have advanced quite dramatically. In that time as well, computers and processors have advanced exponentially. However, mobile robot navigation is still a problem for researchers. To better relate where navigation research currently stands: In 2004 the Defense Advanced Research Projects Agency (DARPA) began their Grand Challenge – an under-300 mile competition through the desert. The goal of the Grand Challenge was to reward the mobile robot that finished the course in the quickest time. No group finished. The best group traveled 7.4 miles before flipping over on a hairpin turn.

The results of the Grand Challenge show exactly where robot navigation is, but the results also show that researchers are on the right track. 7.4 miles of autonomous navigation is a long ways from Shakey.

Mobile Robot Navigation Techniques

In her book Introduction to AI Robotics [10] Robin Murphy discusses navigation, localization, and map-making for mobile robots. Murphy poses four questions:

1. Where am I going?

2. What is the best way to get there?

3. Where have I been?

4. Where am I?

The answer to the first question typically comes from a human user or a mission planner.

The second question is the standard navigation question. It is the problem of path planning. Murphy [10] states that there are two forms of path planning, qualitative and

quantitative. Qualitative navigation is defined as path planning based on directives. Example directives for qualitative navigation can be:

1. Go out the door turn right
2. Follow the sidewalk until you get to the building
3. Go in the building
4. Go to the second floor
5. Go to the last room on the left

Quantitative navigation chooses routes "according to some measure of best" [10]. In addition, quantitative navigation contains waypoints that are often fixed points that may not correspond to objects in the world.

The answer to the third question lies in the creation of maps of the environment. As a robot explores an environment it is quite useful for that robot to create some form of map of that environment. With that map, the robot can then choose its own form of navigation as well as localize itself, which is then the answer to the fourth question. Localization is the process of determining position from some form of input, typically sensor input.

The intent of this master's thesis is to better understand the problems faced in mobile robot localization and navigation through the creation and application of mental maps of the environment.

Summary and Organization

Chapter II of this thesis talks about qualitative navigation and the creation of cognitive maps. This chapter also gives the definition for the term *landmark* as well as discusses related work in the field of robotics. Chapter III defines the approach used in designing this system and creating the agents that run this system. Chapter III also discusses possible applications of this system as well as the current constraints on running the system in the real world. Chapter IV overviews the individual hardware and software systems that were involved in creating an overall system capable of obtaining 3-D laser scans from an environment, processing the data, and generating a cognitive map. Chapter V discusses the algorithms used to process the sensor data. This chapter presents much of the math that was involved in this thesis and also presents several sections of pseudo-code to better describe each algorithm. Chapter VI begins by discussing the localization and navigation algorithms used and Chapter VI ends by presenting the results that these algorithms produced utilizing the cognitive map generated in Chapter V.

Chapter VII discusses conclusions from this work as well as possible future work.

CHAPTER II

QUALITATIVE NAVIGATION

Cognitive Mapping

Navigation is as important of a concept in the world of robotics as it is in the world of humans and animals. Knowing where one's self is and how to traverse the current environment to reach a goal position is a problem which nature tackled eons ago. Certainly, with so many years of experience, there are examples in nature that the robotics world can draw upon.

How does a bee, after traveling great distance to find pollen, return to the hive? How do birds migrate year after year to the same spot? How do ants travel hundreds of meters and yet return to a dime-size nest entrance? The simple answer to the questions above is memory. A more complicated answer is that these animals must rely on the presence of landmarks and their ability to create a cognitive map of the environment. This cognitive map is then stored in the creature's memory and recalled for navigation. Studies have shown that animals that travel over a wide range have a larger hippocampus [4], the portion of the brain that helps handle the memory part of navigation, than animals that do not travel over wide ranges. This indicates that these animals have evolved more memory specific brains than animals that do not travel over wide ranges.

In all three cases mentioned above, landmarks are sensed and only relative spatial information is recorded (information of the form: to the left/right of, between, and next to) in the form of a cognitive map. The bee records the positions of trees and food sources. The birds utilize the positioning of heavenly bodies. The ant uses landmarks such as trees and plants, which, to the ant must appear, as mountains in the distance. From all of this, two questions must be asked:

1. What is a landmark in an environment?

Murphy [10] defines a landmark as “one or more perceptually distinctive features of interest on an object or locale of interest”. Murphy also breaks landmarks into groups such as *artificial* and *natural*. An “artificial” landmark is a landmark onto which features have been added to facilitate better recognition. A “natural” landmark is therefore a landmark on which features have not been added.

Since the sensing module for this work is a laser scan, which returns distance

information, it is important that landmarks be spatially separable in the environment. This body of work will use landmarks that can be reliably and repeatable sensed in the environment and detected spatially to be in a different location than other landmarks. For example, a tree would be a landmark whereas the leaves on the trees would not do to the fact that the individual leaves could not be reliably, repeatable detected and separated.

2. What is a cognitive map and how can one be created?

In an article on cognitive mapping [2], Richard Dagan defines creating a cognitive map as a "process composed of a series of psychological transformations by which an individual acquires, codes, stores, recalls, and decodes information about the relative locations and attributes of phenomena in their everyday spatial environment". Jefferies and Yeap [5] term a cognitive map for an agent as a "memory, i.e. map in the head... for the places it [that agent] visits". For this body of work a cognitive map shall be used to refer to recorded spatial relationships between landmarks detected within an environment. In other words, a cognitive map is a mental map of the environment that includes relative position information (the tree is x number of meters LEFT of the lamppost, which is w number of meters RIGHT of the trashcan).

An interesting example of animal cognitive mapping is the Clark's nutcracker, which stores food for the winter over a wide range. What is interesting about this bird and its use of the cognitive map is its apparent ability to use relationships between landmarks in its environment. Research by Kamil and Jones [6] show that the nutcracker "can learn to find the point halfway between two landmarks that vary in the distance that separates them... This demonstrates the ability to find a point defined not by the relationship between a goal and a landmark but by the relationship between landmarks". In a forest full of trees that lose their leaves in winter, it is this ability to only utilize the relation between different landmarks in the search of a goal that enables the nutcracker to find in the winter, seeds stashed during the summer.

Work by Yeap [5] details the application of the theory of cognitive maps [18] to the creation and building of a map for a robot outfitted with a bank of sonar sensors. Similar methods have been implemented by Thrun [16] and Kortenkamp [8].

Sensory & Landmark EgoSpheres

Work by Peters [11] and Kawamura, et al. [7] define similar methods for the creation of a cognitive map. The sensory egosphere (SES) proposed in [11] is a means of holding in a short-term memory structure information about objects sensed from the environment. The SES stores

information about an object's location as well as other descriptors for the object such as color, type, and name.

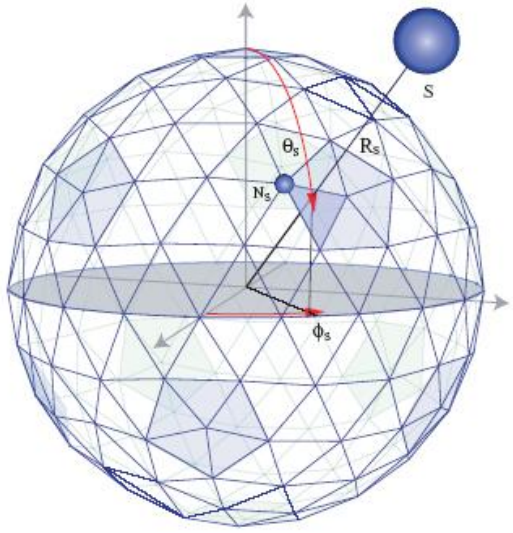

Figure 2: Sensory Egosphere displaying an object $\mathbf{S}$ *detected at* ϕ_s *&* θ_s *with range* $\mathbf{R}_s$ *stored at location* $\mathbf{N}_s$*. Taken from Peters [11]*

The objects stored in the SES are posted on a geodesic dome centered about the robot. Detecting objects and storing their position relative to an individual, or in this case a robot, is the basis for creating an SES as a structure representing what is in the environment right now. The humanoid robot, ISAC, is shown in figure 3 using the SES as it records information about objects placed in front of it. Once these objects have been recorded onto the SES, ISAC can then draw upon this database to accomplish tasks. An SES can be any structure that provides a means of storing objects detected in the immediate, surrounding environment.

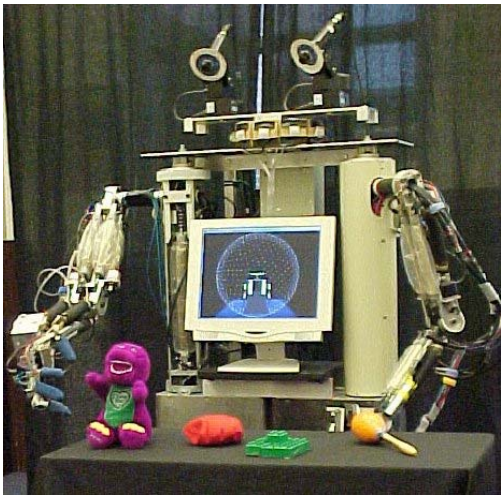

Figure 3: ISAC uses the SES while grasping a Barney Doll. Picture taken at the Center for Intelligent Systems http://eecs.vanderbilt.edu/cis

Storing the location of an object relative to a robot is also, in essence, creating a cognitive map. The idea of the SES is an extension of ideas proposed by Albus in [1] where he defines an egosphere as "a spherical surface that is a map of the world as seen by an observer at the center of the sphere". The SES serves a more complicated purpose, but the simple concept of an egosphere [1], a structure that simply relates the location of objects around the robot to the position of the robot, should be considered interchangeable with the concept of a cognitive map for the robot's current position. The combination of these structures should be considered an egocentric cognitive map.

Further work by Kawamura, et al. [7] defines the use of a landmark egosphere (LES) for navigation in mobile robots. The LES is a representation of detectable objects in relation to each other. An LES is used to specify goal locations as well as target positions en route to the goal. The difference between an SES and an LES is that an LES defines how objects relate to a desired, not current position.

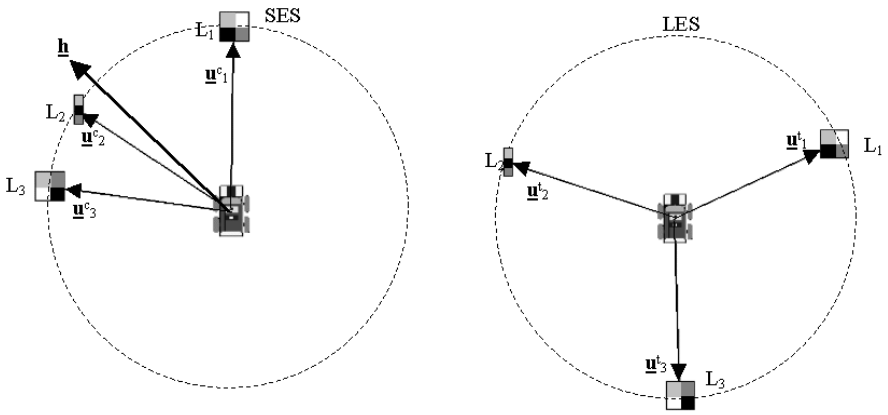

Figure 4: The SES detects that objects are not in the correct locations to match the LES. Moving along the vector ***h*** *will bring the robot towards the position specified by the LES. Figure taken from Kawamura, et al [7]*

Like the work in [7], the Clark nutcracker can be thought of as using its egosphere to act as an SES and LES in the generation of an egocentric cognitive map of where it has stored its seeds for the winter.

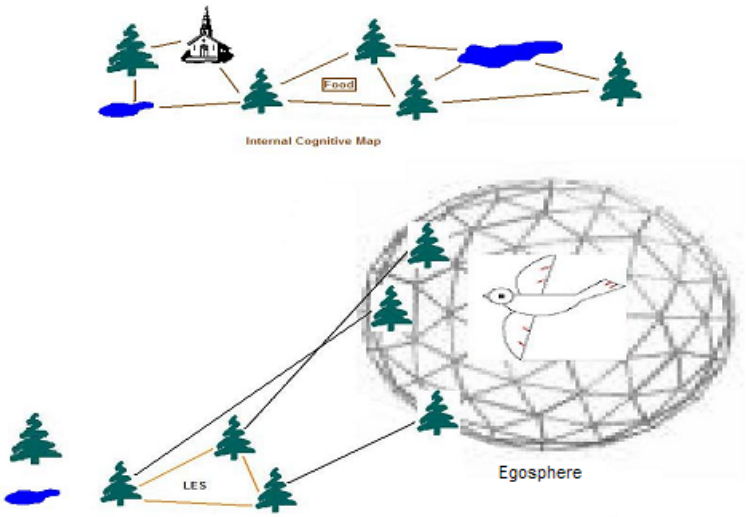

Figure 5: Clark Nutcracker demonstrating the use of a cognitive map and navigating via its own egospheres

This research is intended to create an egocentric cognitive map through driving a mobile robot in an environment and recording several different egospheres [1] at different locations and combining them. Once a cognitive map has been created, newly recorded egospheres can then be used to localize the robot within the map as well as to determine where to navigate and what

obstacles to expect along the way. This function parallels the LES work by Kawamura, et al [7].

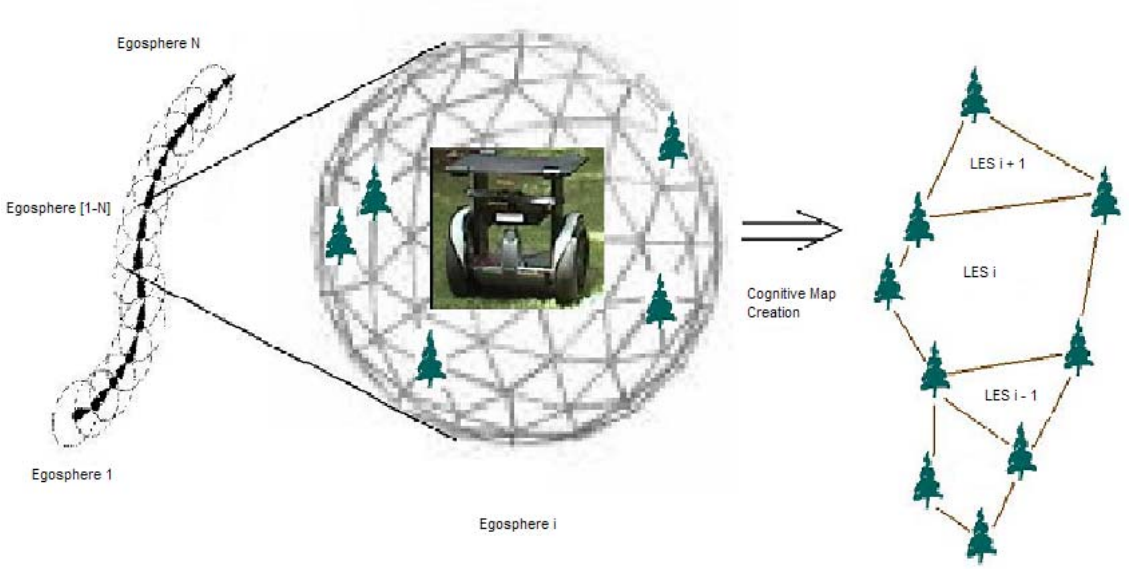

Figure 6: The robot moves through an environment recording several egospheres and linking them to create a cognitive map

3-D Landmark Detection

Creating a 3-D egocentric cognitive map requires the ability to sense 3-D information about the environment. Work by Thrun [17], Montemerlo [9], and Surmann [15] show how 3-D images of an environment can be constructed using either one or two 2-D laser range finders. The works by Thrun [17] and Montemerlo [9] both use two 2-D scanners, one facing forward and the other looking up, to create real-time 3-D images of an environment. The work by Surmann [15] shows how one 2-D laser scan can be used to learn 3-D information about an environment. This research will follow along the lines of Surmann's work. One 2-D laser scanner will be used, mounted on an axis of revolution to obtain 3-D parametric information about the environment. This will not be real-time, but can be used to create a working SES for the robot, which in turn can be applied to the generation of a cognitive map. Surmann also demonstrates simple surface detection using a completed 3-D scan. The surface detection is mainly suitable for indoor environments. The work presented in this thesis will focus on surface detection for outdoor environments.

CHAPTER III

APPROACH OVERVIEW

This thesis will cover a large amount of work done to apply and integrate different hardware and software systems. In addition to system integration, specially developed software agents for control will be mentioned. This section is intended to relay the reasoning behind each system created as well as to discuss how this system may be applied and what constraints this system faces.

Methodology

This system was constructed utilizing a variety of different and complex components. The methodology used in the development of this system was to build the required new components in such a way that would enable other researchers to quickly learn how to use this system and build upon it. Not all of the components, however, of this system were new, some were standard components in modern robotics research or re-used components from prior research. Chapters IV and V will discuss this further.

The communication agents in particular were designed so that the information that they handled could easily be added to or subtracted from or listened to by a new agent. The data processing agents and functions were designed to retrieve pertinent information from a 3-D laser scan, to quickly identify landmark size, position, and orientation, and then to compare the appropriately derived information to past scans.

The intention behind only storing landmark size, position, and orientation was to show that, while complex information about a landmark is helpful in navigation, the most important information is simply the size of the landmark and its position and orientation relative to other landmarks. In addition to the system integration discussed above, this thesis will show that through simple numerical techniques:

1. Useful landmarks can be identified in an environment
2. Important information can be quickly extracted for each landmark
3. Information about successive scans of the environment can be combined to construct a 3-D map, useful to both a human user and a mobile robot
4. A simple 3-D map of the environment containing only simple geometric information about

landmarks and their relation to each other is sufficient to allow a mobile robot to localize its position and navigate from point to point

Applications

Obviously, this system was not intended to apply to every environment or every scenario. This system was designed for use in outdoor environments that contain the presence of some landmarks, but are not densely overpopulated with landmarks. The idea is: not enough landmarks and the robot has trouble localizing itself, too many landmarks and the robot has trouble navigating or could find several possible localizations. This research was not aimed at overcoming every possible obstacle that may arise in an environment, only at demonstrating effective navigation through outdoor walkways and courtyards.

The types of landmarks that work best for this research are landmarks that "stick out" in an environment, in other words, landmarks that can be separated from each other. For example, when a laser system looks at a bunch of bushes packed closely together it is likely that the system will interpret what it sees as one big "glob" of points. This is due to the fact that the leaves and branches from the bushes intermingle in such a way that it is extremely difficult to separate them. However, the same laser system can look at a bunch of bushes with some slight separation and see several separate landmarks. It is simply important that landmarks be separately distinguishable. In this way, other landmarks that are effective include, but are not limited to:

1. Light posts
2. Trash cans
3. Benches
4. Sides of building
5. Tree trunks
6. Signs

Constraints

There were many constraints involved in this research. One of the main constraints was time. The time it takes to process the vast amount of points returned by one laser scan was a constant factor. Indeed, some programs were written just to pre-process the laser scans in order to throw away points that need not be looked at before more time expensive programs were run.

A lot of work and effort went into determining the best way to process the laser scans so that useful information could be extracted quickly. Since this research was intended to prove the concept that simple geometric information about landmarks can be used to create maps, localize, and navigate, it was decided to enforce the K.I.S.S. (Keep It Simple Stupid) concept when designing the laser processing and landmark identification routines. All of the processing routines were intended to be simple algorithms that can, if needed, be improved upon in the future.

One further constraint was that the laser was the only sensing module on the robot, which meant that there were no other sensors to detect the emergence of non-stationary obstacles (people, pets, etc) while navigating. Until a second sensor system is installed, this research system does not serve much of a practical purpose.

CHAPTER IV

SYSTEM OVERVIEW

Hardware

This section discusses the hardware system used for this research including the Segway robot, the SICK laser scanner, and the laser rotation platform.

Figure 7: Segway RMP with rotating laser mounted

Segway Robotic Mobility Platform

Vanderbilt University was one of twelve research institutions to receive the Segway robotic mobility platform (RMP) in the summer of 2003 as part of research grant sponsored by DARPA. The purpose of the grant was to allow universities the opportunity to experiment with a new form of mobile robot. No other robot exhibits the Segway RMP's unique capabilities.

Figure 8: The Segway HT, a commercial product produced by Segway LLC. The RMP is its mobile robot cousin.

Based on the concept of the inverted pendulum, the Segway RMP, produced by Segway LLC, is a robotic variation of the Segway HT, originally designed as a human transport. Like the Segway HT, the RMP has a working payload in excess of 200 lbs. The RMP can reach a maximum speed of 8 mph, and can maintain both specifications mentioned prior for a period of approximately two hours.

The most unique capability of the RMP, however, is its ability to self-balance on just two wheels while carrying maximum payload. A bank of five internal gyroscopes monitors the pitch and the roll of the RMP and enables it to operate without falling over. In addition to its self-balancing nature, one further aspect of the RMP worth noting is its potential striking similarity in size to the size of a human (the RMP is approximately 3' tall, however, with sensors and hardware added to the top, it can reach heights near 4 ½' to 5'). The Segway RMP has virtually the same footprint as a person, 2' x 2 ½'. The turning radius of the RMP is also the same as a person, zero. Sensors can be added so that they interact with the environment much closer to human "eye-level" than sensors on typical mobile robots. The RMP can operate more naturally in environments designed for people by people.

The RMP must be controlled from a computer that sends signals through a CAN bus to an on-board microcontroller. CAN stands for Controller Area Network and is a control protocol primarily adhered to in the automotive industry for robust control of engine management and brake control. More information about the CAN protocol can be found at www.kvaser.com.

For safety purposes the signal sent to the RMP must be sent through two CAN lines and received by the robot in sync. In addition to this safety measure, there are two kill switches

located on the RMP. If the CAN signals are received out of sync or if one of these kill switches is triggered then the RMP will immediately shut itself off. If the internal gyroscopes detect a pitch angle in excess of 45^o or a roll angle in excess of 20^o it will shut itself off as well.

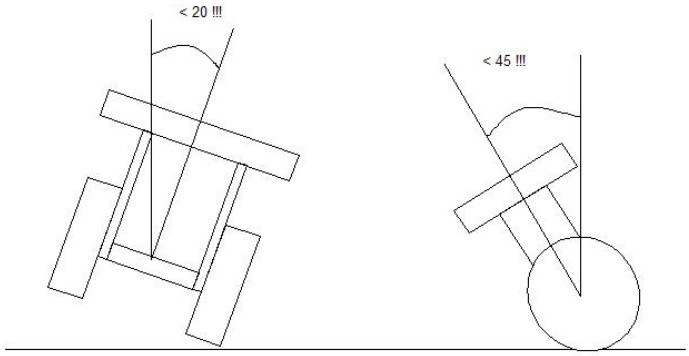

Figure 9: Pitch and roll limits that force the RMP to shut itself off

To protect the laser and control laptop from such disasters, a simple safety system constructed of PVC pipe was implemented to catch the robot as soon as it passed the 45^o limit.

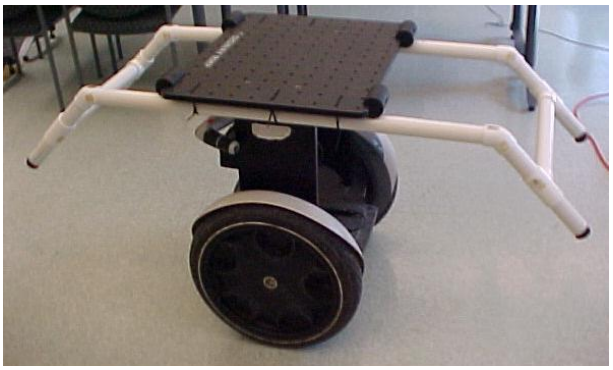

Figure 10: PVC pipes help catch the RMP if it falls, and protects delicate sensor equipment

All of its unique capabilities make the Segway RMP an interesting variation on the standard mobile robot. The near human dimensions of the RMP allow researchers to operate this robot better in environments designed for people. The high payload and speed capabilities give researchers much more design freedom with the "gadgets" they choose to use on the Segway RMP.

SICK LMS 200

This SICK laser scanner, model LMS 200 produced by SICK Inc., is a standard sensing module for practically all mobile robot research. The laser scanner, shown right, takes scans every 0.5^o for the forward 180^o. The scanner has a range of 8.2 meters and a resolution of 10

mm. The scanner is adequate for both indoor environments as well as outdoor environments. One drawback of the SICK laser scanner is that it does not detect glass very well. This inability is acceptable for the research described here, though, due to the fact that this work is conducted outside in environments without a lot of glass walls.

Figure 11: SICK LMS 200 developed by SICK Inc.

Laser Rotation Platform

An extremely vital part of this research is the ability to acquire 3-D laser images from the environment using a laser scanner only capable of returning 2-D information. The solution was to rotate the 2-D laser through the third dimension. This parallels the work mentioned in [15], however, for this application it was decided to turn the laser on its side and sweep the environment from side to side, rather than mount the laser as usual and sweep up and down. The task of rotating a laser is much less demanding on a system than lifting a laser.

As luck would have it, an adequate rotation platform capable of smoothly moving the laser had already been developed by a previous graduate student, Arnon Ruengcharungpong [14]. The only modification needed was the installment of a stronger motor.

To maximize torque as well as to compliment the fact that this would be an open-loop system a high torque stepper motor was chosen. The stepper motor chosen was The HT23-400 produced by Applied Motions Inc.

The final step in the construction of the laser rotation platform was the implementation of a personal computer for control of the stepper motor. A controller board that connects to a computer's parallel port and drives the stepper motor was ordered from http://kitsrus.com and constructed in the lab.

Software

This section discusses the software systems used to control the robot, laser, laser rotation

platform, and to handle the communication. In addition, this section offers a brief overview of the user interfaces developed for this research. Below is a diagram of the overall software system.

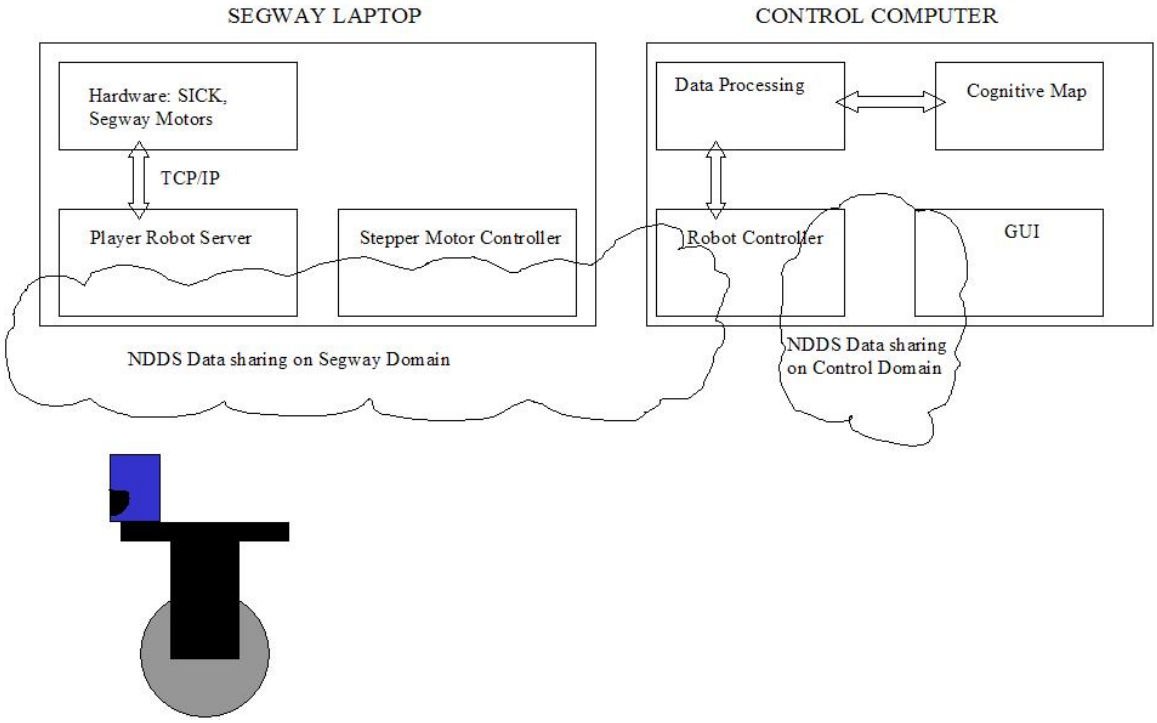

Figure 12: Different software agents are required to run all of the Segway's many processes.

The intention behind the system shown above is to ensure that only ONE program sends commands to the robot. However, other programs can be added to listen to the data being shared with the RMP by subscribing to the Segway domain. Programs can also be added to communicate with the robot control program. An example set of programs that could be added is:

1. Program to listen to other sensor information (if further sensors were added such as vision) coming off of the robot
2. Program to process this sensor information and derive control commands to be sent to the robot controller

If more programs were added to process sensor information and derive control commands, the robot controller would need to act in an executive manner and choose which set of control commands to pass on to the robot. The design of this system was done with the intent that future systems would need to interact with it, and that this interaction would occur through NDDS [12]. It is also worth mentioning at this point that the protocol used to handle the

communication, NDDS, is slowly being implemented on other systems in the lab. A description of NDDS is detailed later in this thesis.

Robot Control

Control of the robot is achieved through the use of the Player robot device server [3] primarily developed by researchers at the Robotics Research Lab at the University of Southern California. The Player server runs on a host computer connected to the robot. To control the robot, clients must simply subscribe to this server. Player provides high-level client utilities that are used to send the actual commands to the robot as well as access data from the robot.

For this thesis, one client program was written. This client was run on the computer mounted on the robot. This client listened for commands from the NDDS communication agents discussed below and published information to those agents. Once connected to the robot, this client continually accessed and published all data coming from the robot, as well as continually listened for new commands to give to the robot. This client's only purpose was to ensure that data to and from the robot was current and as up-to-date as possible, therefore no other tasks were assigned to this client.

Laser Control

Two types of control were implemented for the laser system. The first form of control was used to access the data coming from the laser. This was achieved through the use of the same software system, Player, that was used to control the robot. Player also acts as a device server for most forms of sensors and other devices commonly used by researchers. For a complete list of robots, sensors, and devices currently supported by the Player device server please refer to http://playerstage.sourceforge.net.

Accessing laser data was not the only form of laser control required by this research. In order to take 3-D scans of the environment, the laser had to be rotated about the third axis. As mentioned above, the controller board, used to control the laser rotation, connected to the computer's parallel port. A control agent was written to listen for commands and then to rotate the laser platform by sending the appropriate ASCII code to the parallel port so that the controller board would drive the stepper motor. The commands that this agent listened for were commands that specified when to begin a laser scan and in what direction the scan should proceed.

NDDS

As in everything in life, communication was vital to the success of this work. Proper communication between processes and programs in this system was an important part of this research. The communication protocol chosen was NDDS [12], which stands for Network Data Distribution Service. NDDS is a real-time application developed by Real Time Innovations (RTI) Inc. There were several pros for using NDDS:

1. Easy to use and handles most of the complicated components on its own
2. Can operate cross-platform (Linux & Windows OS's)
3. It is already being implemented by fellow researchers at NASA-JSC and is currently being considered for use on other robots within the lab

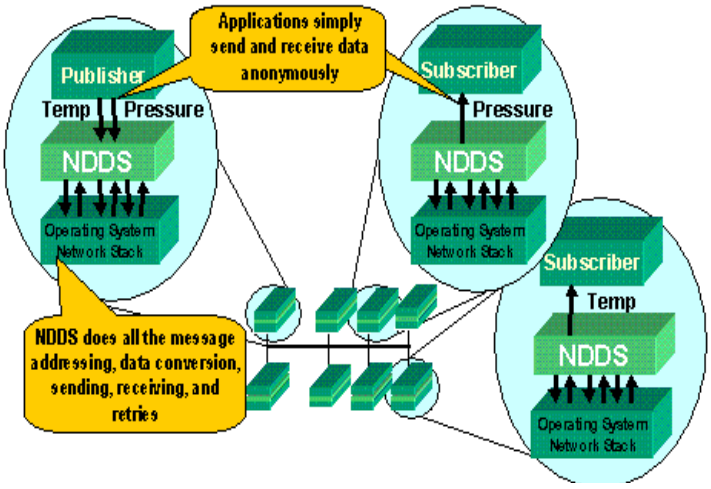

Figure 13: NDDS applications can easily send & receive data across different platforms. For more information visit www.rti.com

NDDS works by creating a domain to which processes can subscribe, publish, or both. Any number of subscribers can listen to the information published on this domain, however, a particular data set can only be published by one publisher. The types of data that can be published are message strings, data arrays, and boolean commands. In order for a subscriber to access a NDDS domain, the subscriber must simply know:

1. The domain number to be accessed
2. The format of the data from that domain the subscriber is interested in

User Interfaces

A number of different user interfaces were written to work with this system. As the

system was being created it was necessary to write different user interfaces to help understand as well as represent how the research was progressing. Only a handful of these user interfaces were intended to be used with the final system:

1. OpenGL graphic used to display the actual X, Y, Z points for a 3-D scan

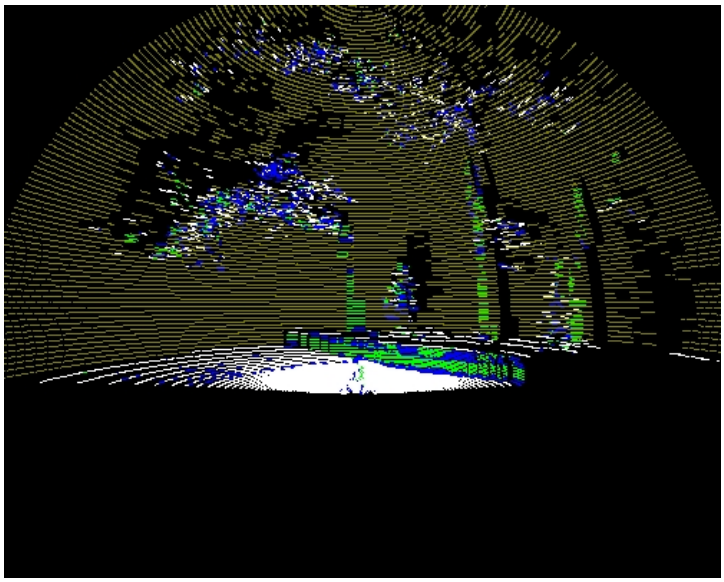

Figure 14: OpenGL graphic displaying points from a 3-D laser scan

This graphic allows the user to understand what the scanned environment looks like as well as to serve as a tool for displaying features determined about the individual X, Y, and Z points. This graphic will be used in the later chapters in this thesis.

2. GTK based GUI used to display forward and aerial view of the detected landmarks center of masses.

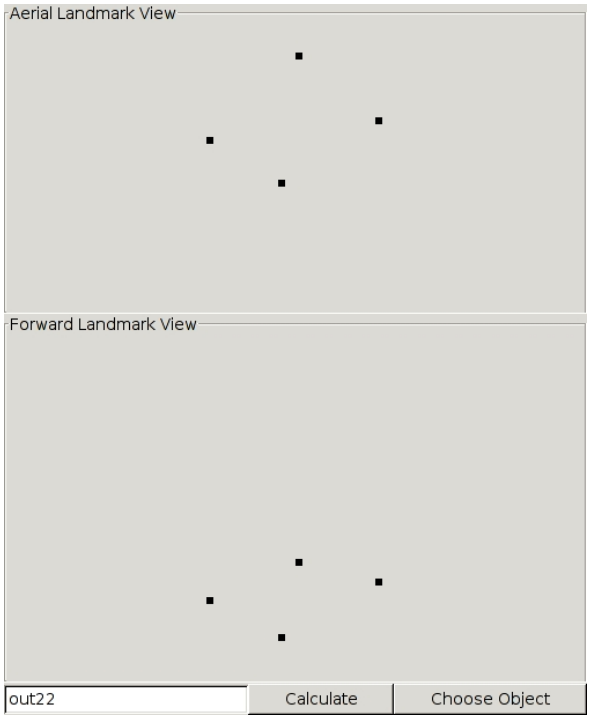

Figure 15: Displaying landmark mass center information for an inputted scan file

This GTK based graphic can be used to show a forward and aerial view of the center of mass locations for landmarks detected from a scan.

3. GTK based GUI used to accept commands from a person during training or running of the system

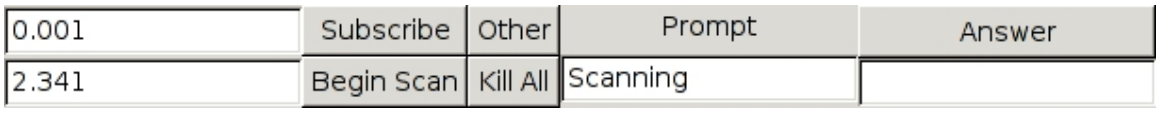

Figure 16: GUI developed to train and control this system

This GTK based graphic was used to control the system while training. It was designed to be small and simple so that it could be built upon.

4. OpenGL graphic used after training to display the resulting cognitive map of the environment

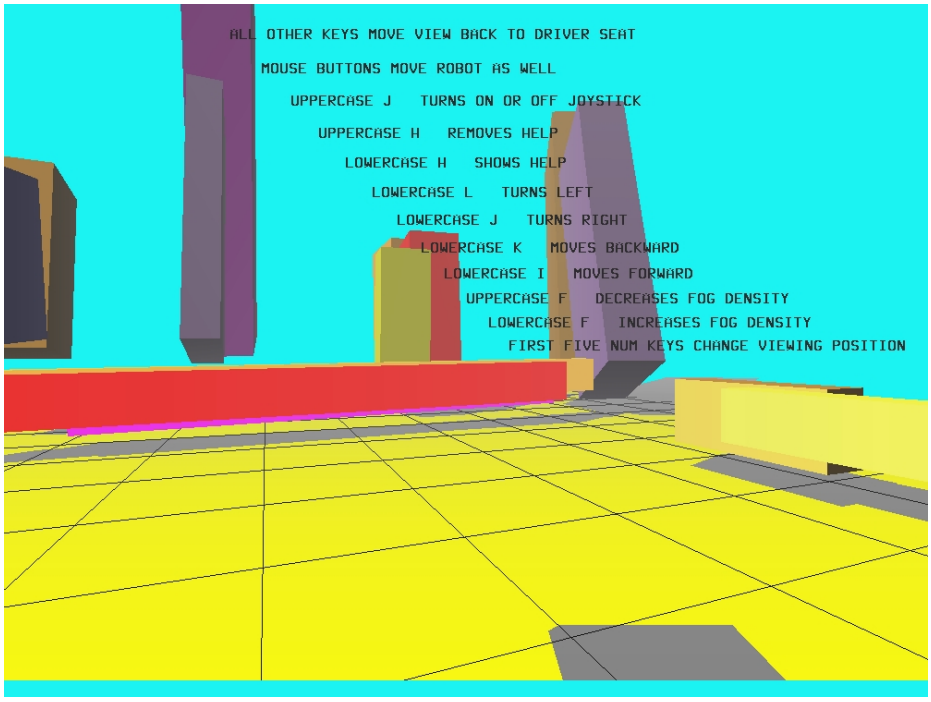

Figure 17: Display of a cognitive map represented with OpenGL

This display is an OpenGL representation of the cognitive map stored within the robot. It works as an interactive environment that the user can drive a simulated robot around in. The user can also adjust the viewing position to be either in the driver's seat of the robot, or one of five aerial views looking down. The mouse, keyboard, or an attached joystick can be used to navigate within this environment. In this representation, blocks that overlap each other typically represent the same landmark detected in different scans and therefore shifted an incremental amount.

CHAPTER V

SYSTEM TRAINING

Data Acquisition

To produce a 3-D laser scan image of an environment requires the integration of all of the systems discussed above.

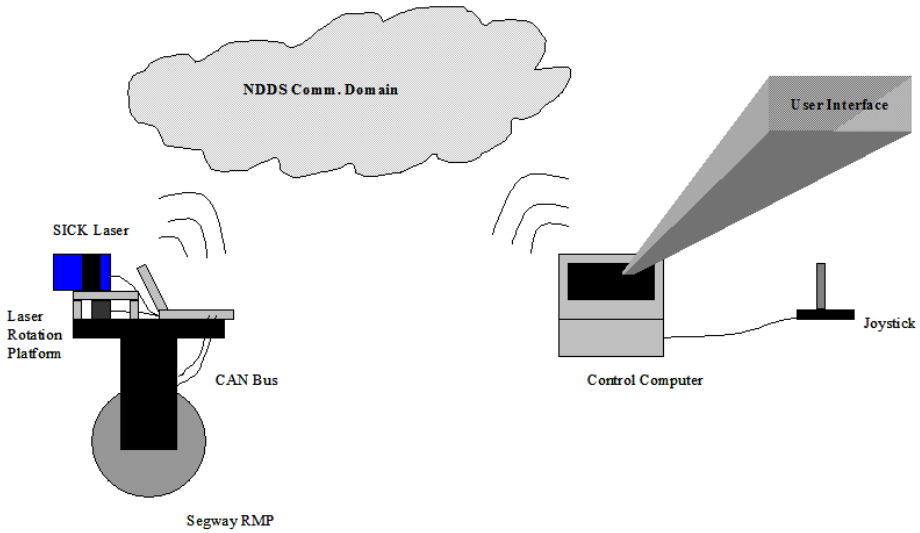

Figure 18: A person interacts with the user interface & joystick, sending signals over NDDS to the RMP

It is not necessary to have the robot autonomously acquire data for training. Having a human counterpart, called the trainer, to control the data collection helps make the entire training process run quickly and smoothly. Once the Segway has been turned on and the appropriate programs have been started, the trainer begins the training process by subscribing to the robot by hitting the "*Subscribe*" button on the GUI.

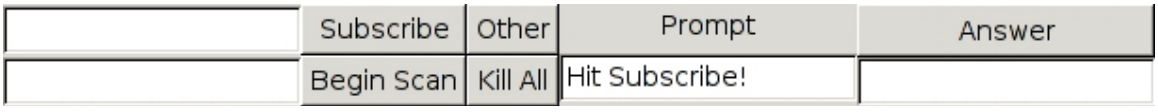

Figure 19: The GUI used to control the system

This sends a command over NDDS that tells all of the processes on the trainer's side to begin listening for data. If this command is successful, the current X-position and Y-position of the robot will be displayed in the boxes on the left-hand side of the GUI. Then, using the joystick

to control the robot, the trainer positions the robot for the first scan. To begin a scan, the trainer simply hits the button "*Begin Scan*". If this is successful, the output prompt will display "*Scanning*".

The "*Begin Scan*" button issues a command over NDDS to the program controlling the stepper motor. This program then turns the laser rotation platform 600 steps = 180^{o}. This program also keeps track of the direction of the last completed laser scan, so that the next laser scan will proceed in the opposite direction. For every one of the 600 steps, this agent publishes over NDDS the current step number as well as the laser scan associated with this step. This results in a 3-D scan of 108000 data points.

On the training side, a program monitors the signals coming over NDDS and for every new step that program records the step number and the corresponding laser scan.

Once the scan is completed, a message is sent back to the training side that the scan has been completed. The GUI prompt then asks the trainer to enter a filename. The format for the filename is a three-letter descriptor of the current training set followed by a numerical number that denotes the scan number for this training set. For example, *out23* would be the filename for the 23rd scan for the training set *out*, in this case *outside*.

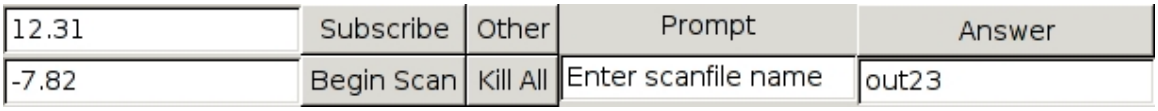

Figure 20: Demonstrating entering a scanfile name

After labeling this file, the trainer hits the "*Answer*" button and the filename is assigned to the data, a directory is created for the processed data, and the data is sent to be processed. The data processing, which will be discussed in the next section, takes a couple minutes after which the prompt will inform the trainer that the processing is finished. The trainer then re-positions the robot in preparation for a new scan.

There are two other buttons on the GUI the trainer can use. The button labeled "*Other*" is currently an extra button intended to be used if the trainer wished to add another capability to this system. The button labeled "*Kill All*" does exactly this. Once this button is hit, the prompt asks the trainer "*Do you want to Exit*". If the trainer answers yes, a signal is sent over NDDS for all processes to stop what they are doing and exit.

Data Processing

When the entire 3-D laser scan is sent to be processed, several actions take place. In this section, the initial data processing will be discussed. The following two sections, **Landmark Identification** and **Map Generation**, will continue with the data processing and discuss how landmarks are identified and the cognitive map of the system is generated. Throughout these sections a sample laser scan from the courtyard outside Featheringill Hall will be displayed as it goes through the various stages of processing.

The first step in processing this data is to transform the 108000 data points from polar to cartesian form (x, y, z). The current coordinate frame being used is:

X – runs out in front of the robot

Y – runs up and down

Z – runs right to left

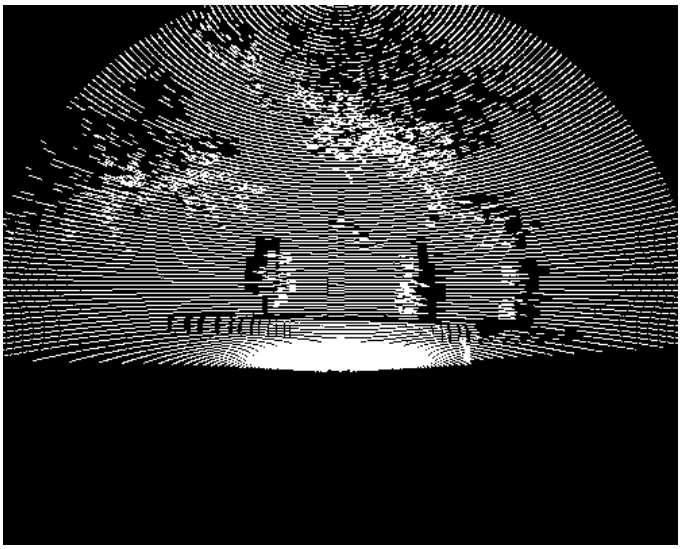

Figure 21: X, Y, & Z points for a sample laser scan

Once the X, Y, & Z points are calculated they are stored in a file (in the directory created for this scan) in case the trainer should ever like to know what the original scan looked like. After storing these points the data processing program then begins to look at the patterns in these points.

The X, Y, & Z points are first processed to determine, for each point, how well that point matches to a plane. However, processing 108000 simultaneously is far too much to ask from most standard computer processors. Therefore, the data is broken into smaller, more manageable chunks. The current size of the chunks is 1500 points at a time. While testing the system, this

number seemed to give the best results in the shortest amount of time. Increasing this number drastically increases the processing time. Decreasing this number further, however, only provided minimal timesavings.

The algorithm currently being used returns two values for every point in the data set. The first value returned is an angle value that corresponds to the angular offset, α, from the horizontal (x-z) plane for the plane onto which the current point would map. The second value returned is a smoothness value, λ. The lower λ is, typically between 0 & 0.1, the better the current point corresponds to a plane with its surrounding points. When λ is not very low (> 0.2) is when the overall average of a particular group of points is flat but there is a high variance between the points themselves (i.e. over rocky sections of ground, or the leaves of trees, or the intersections between a horizontal and vertical plane).

The math behind calculating α and λ:

A 3-D plane is defined by the equation:

$$ax + by + cz + d = 0 \tag{1}$$

For all x, y, & z points (1…n) the above equation can be written in matrix form where the terms $e_{1\text{-}n}$ represent the error terms that arise because these points may not match perfectly to a plane.

$$\begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ . & . & . & . \\ . & . & . & . \\ x_n & y_n & z_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ . \\ . \\ e_n \end{bmatrix} \tag{2}$$

In order to match these points to a plane, the sum of squared error terms, $e_{1\text{-}n}$, should be minimized. To accomplish this the coefficient matrix, $[a, b, c, d]^T = \alpha$, needs to be made as small as possible while adhering to the constraint :

$$\|\alpha\| = 1 \tag{3}$$

which can be written

$$1 - \alpha^T \alpha = 0 \tag{4}$$

This produces

$$e = X \bullet \alpha \tag{5}$$

$$e^T e = \alpha^T \left(X^T X\right) \alpha = E \tag{6}$$

Applying a Lagrange multiplier, λ, to the constraint and adding the constraint to the sum of squared errors term yields

$$E = \alpha^T (X^T X)\alpha + \lambda(1 - \alpha^T \alpha) \quad (7)$$

which is a function of α and λ.

The next step is to differentiate E in terms of α and λ and set the resultants equal to zero, due to the minimizing sum of squared errors constraint.

$$\frac{\partial E}{\partial \alpha} = 0 = (X^T X)\alpha + \lambda\alpha \quad (8)$$

$$\frac{\partial E}{\partial \lambda} = 0 = 1 - \alpha^T \alpha \quad (9)$$

Solving for λ gives

$$\lambda = \frac{\alpha^T X^T X \alpha}{\alpha^T \alpha} \quad (10)$$

which is a Rayleigh quotient. From all of this, the sum of squared errors will be minimized when

λ = smallest eigenvalue of X^TX

α = associated eigenvector (scaled to fit the constraint, $1 - \alpha^T\alpha = 0$)

Below is the pseudo-code for this algorithm:

1. For each data chunk of 1500 points DO:
    1. For each point in this data chunk DO:
        1. Find all points within a specified radius
        2. Collect these points in a matrix
        3. Generate the eigenvalues & eigenvectors for this matrix
        4. From the eigenvalues & eigenvectors calculate α & λ.

This process results in the following image:

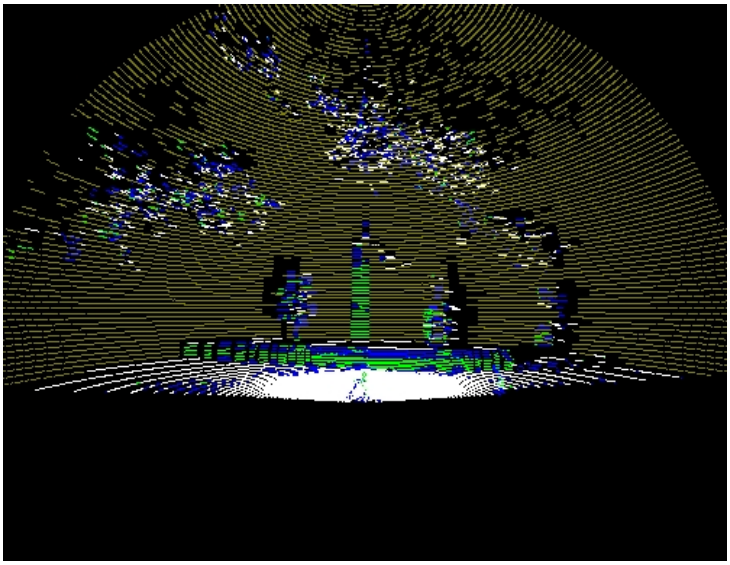

Figure 22: Vertical & Horizontal planes calculated for the sample laser scan

The color code for this image:

Green = Vertical plane

Red = Out of range

White = Horizontal plane

Blue = Does not match to plane

The interpretations that can be made from this are:

1. Trees covered with leaves tend to appear as clusters of blue points
2. Tree trunks, though round in nature, are approximated well as vertical planes
3. Benches and other small vertical landmarks will appear as collections of green points with blue lining the edges
4. Flat surfaces, such as the ground, are detected very well

Landmark Identification

Once the data has been processed in the manner described above, the next step is to begin identifying landmarks in the laser scan. The technique chosen to identify landmarks was simple, yet surprisingly robust. Below is the pseudo-code for this technique:

1. For every point in the image not currently associated with a landmark DO:
    1. Place this point on a stack
    2. Find all other points in the image within a certain radius from this point
    3. Add these points to the stack
    4. RECURSIVELY:
        1. For all points in the stack DO:
            1. Find all other points within a certain radius
            2. Add these points to the stack

This process does not require the pre-processing of points discussed in the previous section. This process could also, theoretically connect all points in the image as one single landmark. However, by combining this process with knowledge gained from the pre-processing discussed above, useful landmarks can be identified quickly and robustly.

The information used from the pre-processing step includes:

1. Ignore points above and below a certain height: Points below a certain height are considered to be part of the floor and undergo their own processing discussed shortly. Since a continuous canopy of leaves can interconnect all landmarks, for the time being points higher than 2.5 meters are ignored.
2. Ignore points that are out of range. This alone greatly speeds up processing.
3. Ignore points that have a very high smoothness factor, λ. Even bushes and trees have relatively low smoothness factors (< 1.0).

The radius defining whether to add a point to the current stack or not was chosen to depend on the distance of the point in question from the origin. The radius used varied between six inches and eighteen inches. APPENDIX A discusses the radius chosen and offers examples as to why it was chosen as it was. APPENDIX A also gives examples of vertical and horizontal plane identification for different tolerances on α and λ.

As mentioned above, points below a certain height that were considered a part of the floor underwent their own processing. The height threshold set was the height of the robot. This would assume that whereever the robot was, it was standing on the floor. Since the Segway RMP does not have the ability to go over bumps (steps) on the ground, all points higher or lower than where the Segway was currently standing had to be considered obstacles. The algorithm that processed points to identify the usable floor was the exact same algorithm used to identify landmarks, but with two exceptions:

1. Only points within a certain height range were looked at

2. Very tight tolerances were kept on α & λ so that points that were not on the same plane as the current floor, or points that were on the same plane but could not be considered smooth would not be added to the floor

The figure below shows the different landmarks detected as well as the floor. Each separate landmark has a different color, though color assignment is arbitrary.

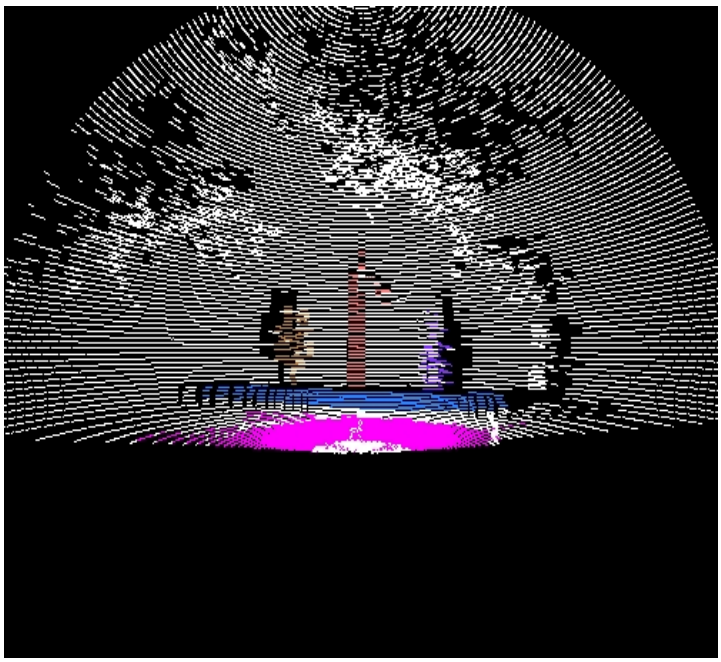

Figure 23: Landmarks detected for the sample laser scan as well as the useful floor area

Cognitive Map Generation

In order to generate a cognitive map from a laser scan, more information is required than simple knowledge that landmarks exist in the laser scan. The most important bit of knowledge is to know where exactly the landmarks are located. It would be nice to know a little bit about the size and shape of the landmarks in question. Once this knowledge is gained, a 3-D map representation of the environment can be created.

The first step is to calculate the center of mass of each landmark relative to the robot. Center of mass algorithms are relatively straightforward. The algorithm used for this research is described in the pseudo-code below:

1. For each detected landmark DO:
    1. For each point in the landmark DO:
        1. Find all other points within a certain radius of this point
        2. Treat the number of points within that radius as the weight for that particular point
    2. Sum X, Y, & Z multiplied by the weight for all points
    3. Divide this sum by the total weight

The radius chosen for this algorithm was 0.05 meters ensuring that objects, like trees with long skinny branches, do not have their center of mass moved from the actual center, near the trunk.

Once the center of mass for each landmark is calculated, that landmark's location in the map is known. However, calculating parametric information about each landmark would be extremely useful in navigation. To calculate the distribution of points for a landmark, first construct the matrix:

$$R_{xx} = \frac{1}{N}\sum_{i=1}^{N}\{P_i - P_{bar}\}\{P_i - P_{bar}\}^T \tag{11}$$

Where,

$$P_i = \{X_i, Y_i, Z_i\} \tag{12}$$

$$P_{bar} = \{X_{bar}, Y_{bar}, Z_{bar}\} \tag{13}$$

$$X_{bar} = \frac{1}{N}\sum_{i=1}^{N} X_i \tag{14}$$

$$Y_{bar} = \frac{1}{N}\sum_{i=1}^{N} Y_i \tag{15}$$

$$Z_{bar} = \frac{1}{N}\sum_{i=1}^{N} Z_i \tag{16}$$

N is the number of points for that landmark.

The eigenvalues of R_{xx} are the distribution of points in each direction. The eigenvectors of R_{xx} are the direction vectors for each eigenvalue in the main X, Y, & Z directions. Therefore, the first row and first column of the eigenvector matrix corresponds to the component of the first eigenvalue in the X direction. The second row, first column of the eigenvector matrix corresponds to the component of the first eigenvalue in the Y direction and the third row, first column corresponds to the component in the Z direction. Assuming uniform distribution of

points, the actual length in each direction can be calculated from the eigenvalues as:

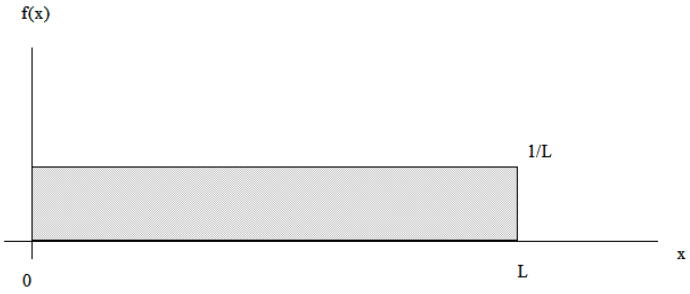

Figure 24: Uniform distribution of points for an object

The mean of $x = L/2$, where L is the length of the object.

Therefore the variance of x can be calculated as:

$$\sigma_x^2 = \int_0^L \left(x - \frac{L}{2} \right)^2 \frac{1}{L} dx \tag{17}$$

Which, when evaluated yields

$$\sigma_x^2 = \frac{L^2}{12} \tag{18}$$

The eigenvalues returned by the equations above are, in essence, the variance of the distributions. With the uniform distribution assumption the estimate length is determined to be:

$$L = \sqrt{12\gamma} \tag{19}$$

Where γ corresponds to one of the eigenvalues (the variance of the distribution in a particular direction).

Figure 25: Two landmarks detected from the sample scan

An example of what this algorithm returns for two objects taken from the scan above: The first landmark is a long bench. The eigenvalues and eigenvectors were calculated to be:

Table 1:Eigenvalues & eigenvectors for a long bench landmark

Eigenvalues	9367.85	17649.7	4.83985e+06
Eigenvector: X component of eigenvalue	0.925795	-0.308432	-0.218572
Eigenvector: Y component of eigenvalue	0.318313	0.947926	0.010620
Eigenvector: Z component of eigenvalue	0.203915	-0.079406	0.975763

From this, the third and largest eigenvalue for the bench protrudes primarily in the Z direction.

The second landmark is a tree behind the bench. The eigenvalues and eigenvectors were calculated to be:

Table 2: Eigenvalues & eigenvectors for a tree landmark

Eigenvalues	51469.7	70702.2	366542.0
Eigenvectors X component of eigenvalue	0.986215	0.015724	-0.164719
Eigenvector Y component of eigenvalue	-0.162859	-0.083827	-0.983082
Eigenvector Z component of eigenvalue	0.029266	-0.996356	0.080111

Again, the third eigenvalue is the largest but this time it protrudes mainly in the Y direction.

From the uniform distribution assumption the lengths can be calculated as

Bench: 0.335 meters deep, 0.460 meters tall, 7.620 meters wide
Tree: 0.785 meters deep, 2.097 meters tall, 0.921 meters wide

The odometry information taken from the robot during the training period can be used, along with the relative position, size, and orientation values just calculated, to generate an overall map of the environment. Two views of the cognitive map generated for the trial scan are shown in figures 26 and 27.

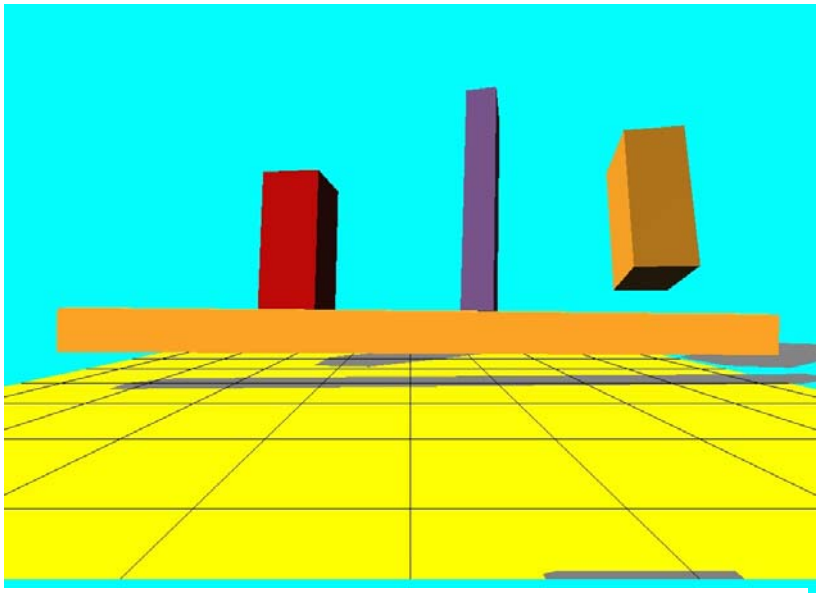

Figure 26: Front view of the cognitive map from the sample laser scan

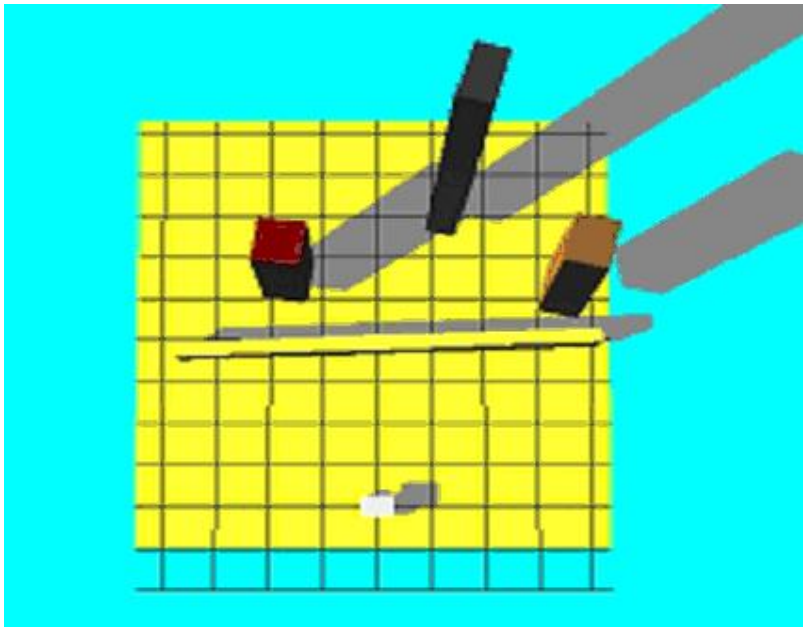

Figure 27: Aerial view of the cognitive map from the sample laser scan

The program used to make the map also creates a robot the size of the Segway RMP and places that robot within the map at the point that is considered to be the origin of the map. Effects such as shadows, fog, and the floor grid were added to aid the trainer in understanding and realizing the different 3-D locations of landmarks.

Figures 28 through 33 show some laser scans taken from different locations of the same area, along with the map generated by combining these scans.

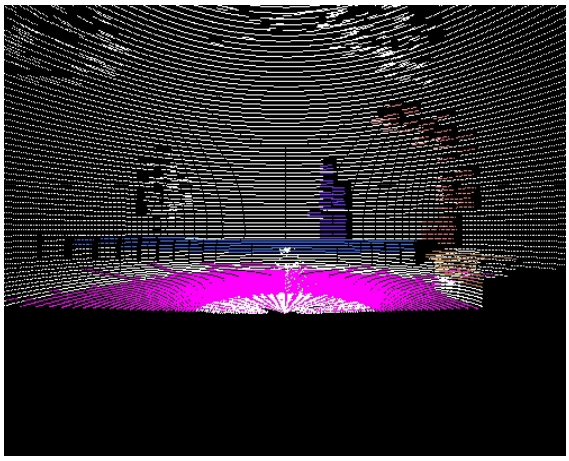

Figure 28: A 3-D laser scan of the environment

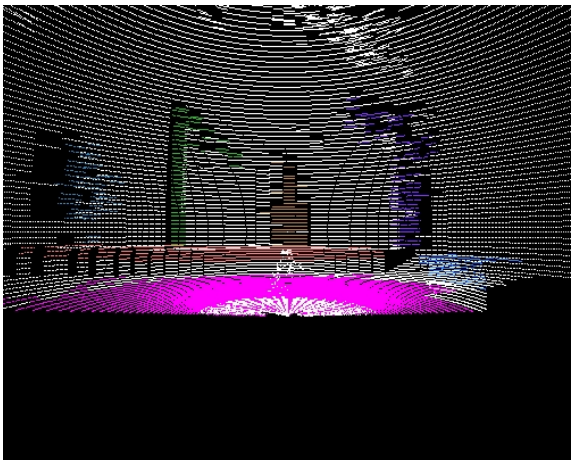

Figure 29: A scan taken after moving forward 0.9 meters

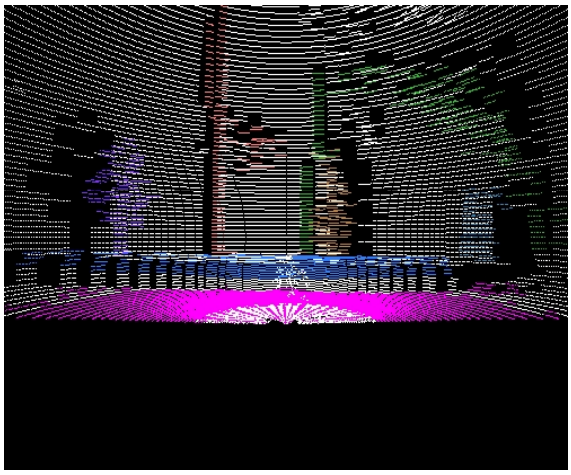

Figure 30: A scan taken after moving forward an additional 1.3 meters

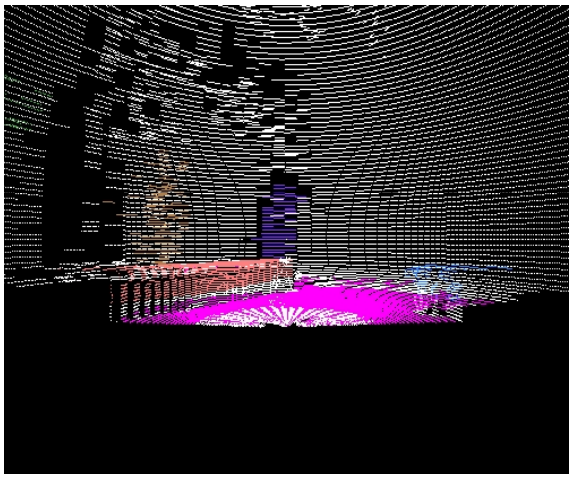

Figure 31: The final scan taken after moving forward an additional 0.5 meters, right 0.5 meters and turning 45^{o} right

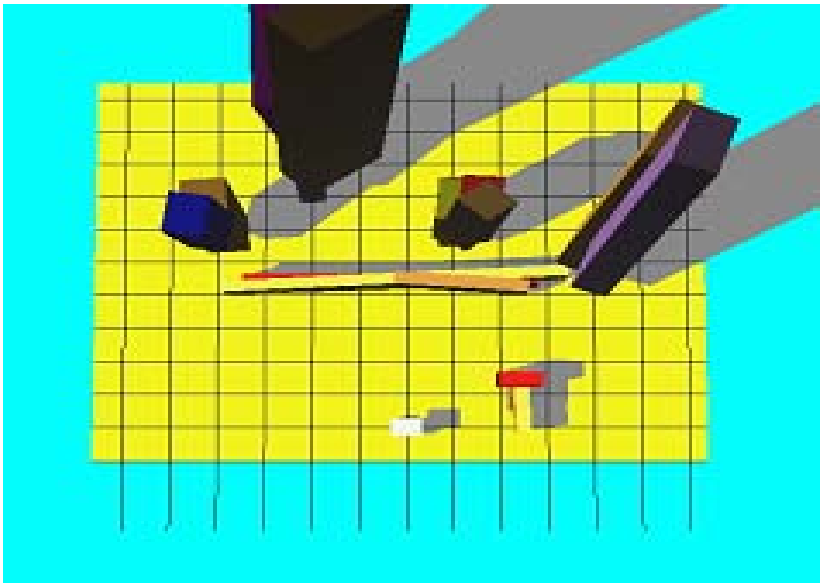

Figure 32: Aerial view of the cognitive map generated from the above set of sample scans

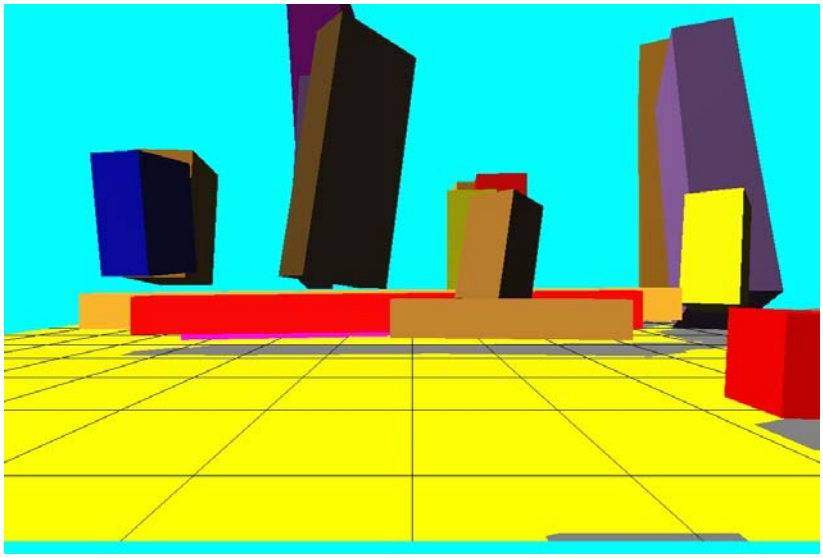

Figure 33: Forward view of the cognitive map generated from the above set of sample scans

Even though each algorithm for data processing mentioned above as well as the landmark identification routines all act in a general sort of way (i.e. different landmarks such as trees, benches, bushes, light posts, etc are not identified as such, rather identified as a box with dimensions dx, dy, dz and orientation dtheta). Landmarks from different scans seem to coincide quite well. Each of the four scans detected the long bench in front of the robot, the bench to the

right of the robot, as well as the same trees in the environment. The map generation shows that these groups of landmarks detectably maintain their geometric configuration as the robot moves about in the environment.

Following the process and techniques mentioned above, 27 scans from the area just outside Featheringill Hall were taken. Odometry information between scan locations was recorded and all of the data was processed using the algorithms outlined in this document. Different views of the overall cognitive map generated are shown in figures 34 through 39.

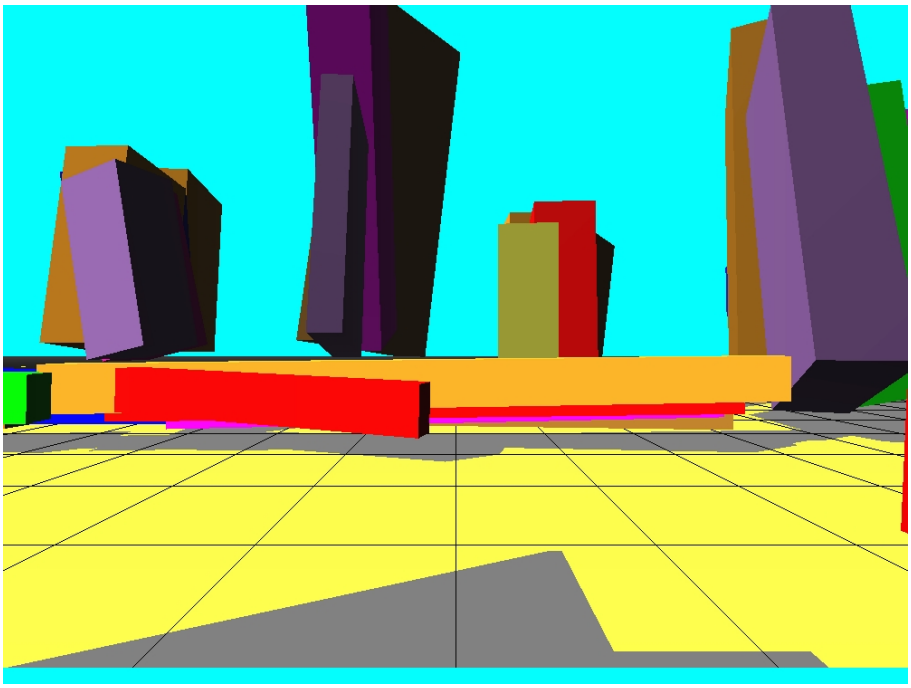

Figure 34: Driver's seat view of the overall cognitive map

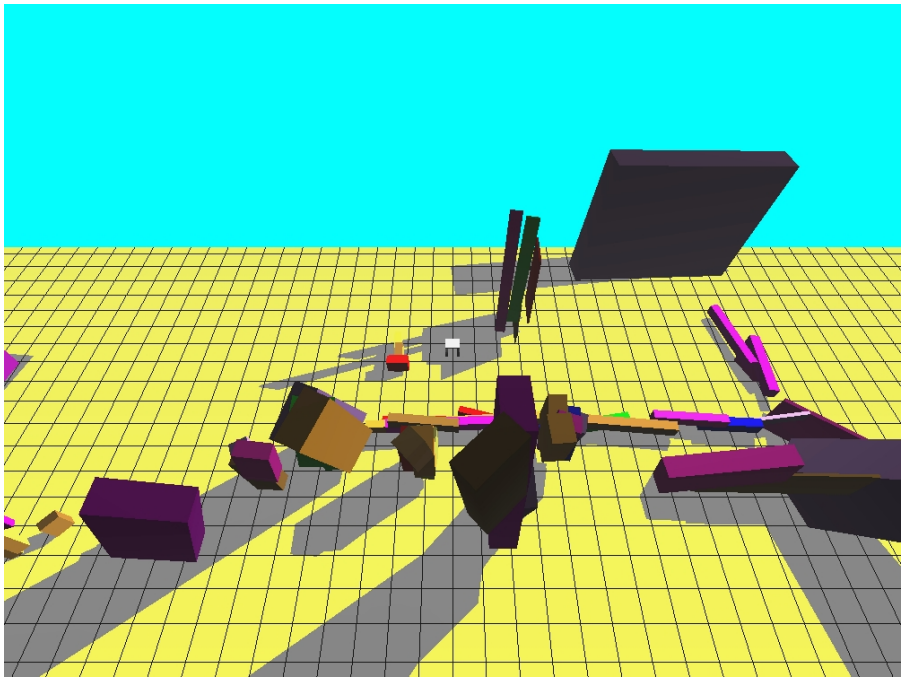

Figure 35: Above the robot & facing towards the front of the robot in the overall cognitive map

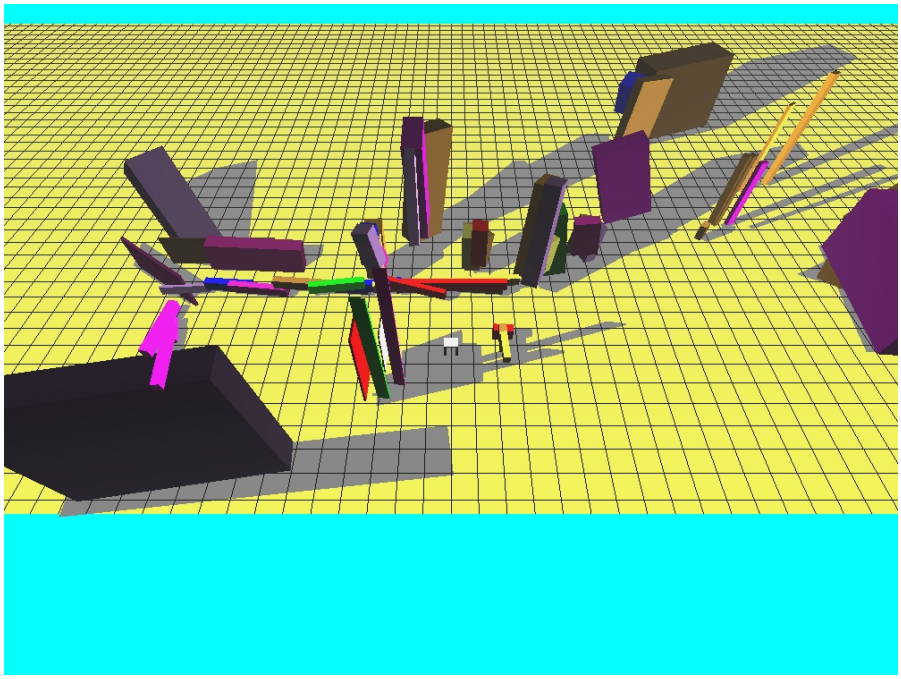

Figure 36: Behind & above the robot; facing the same direction as the robot in the overall cognitive map

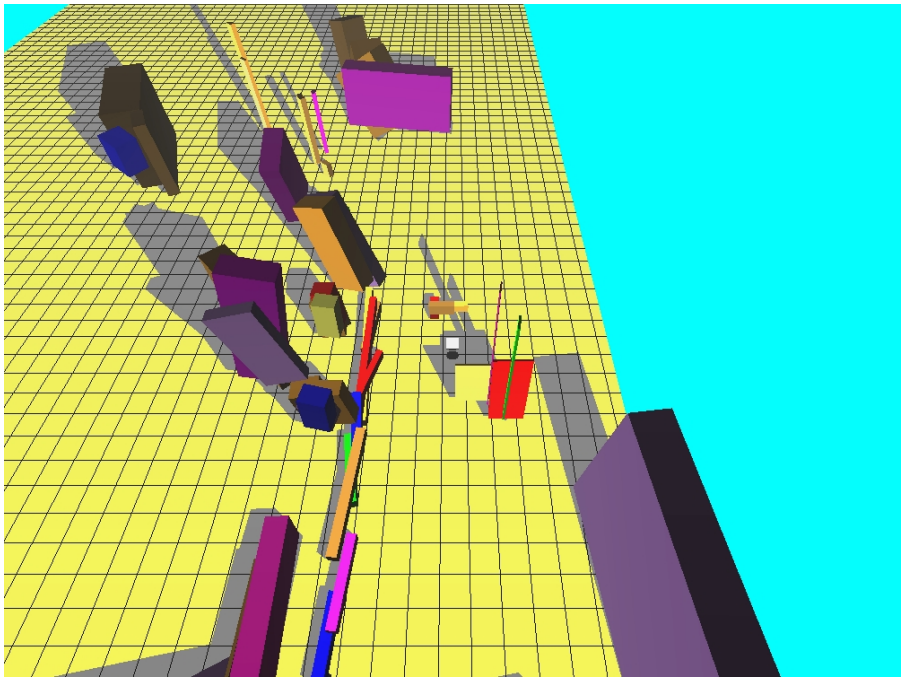

Figure 37: Left & above the robot in the overall cognitive map

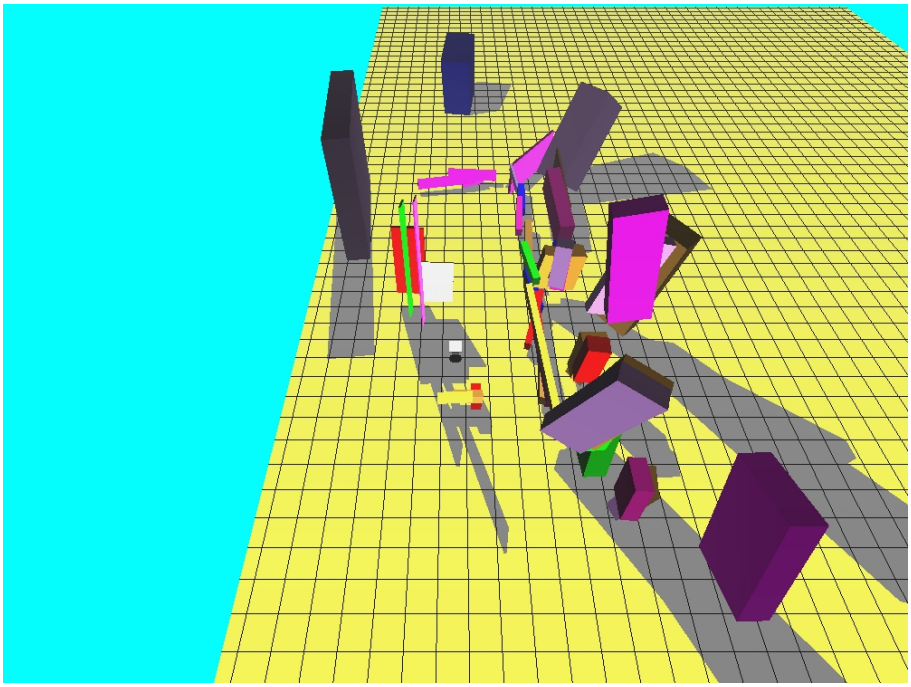

Figure 38: Right & above the robot in the overall cognitive map

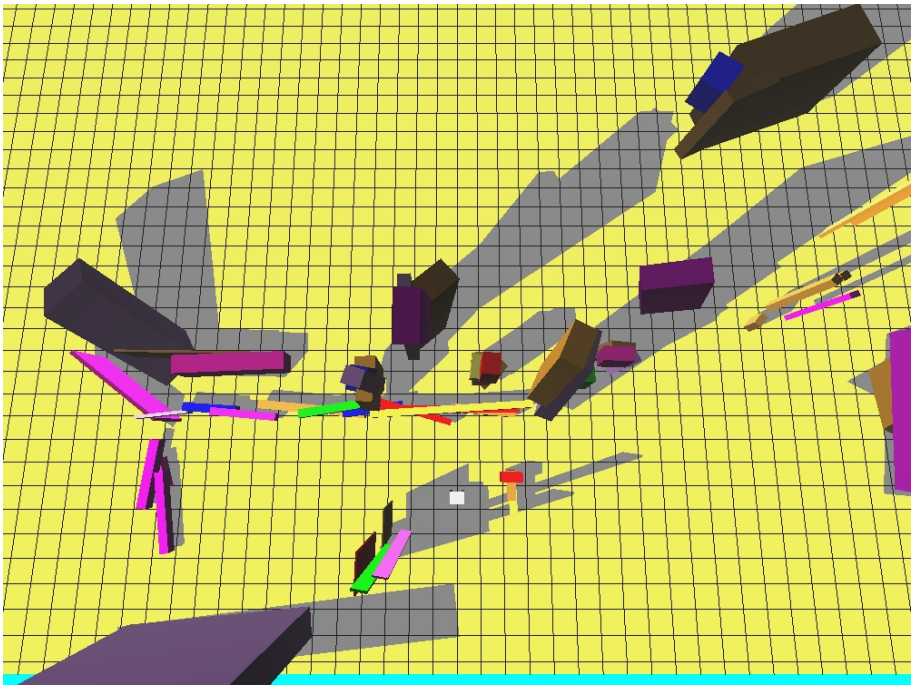

Figure 39: Above the robot facing down in the overall cognitive map

From these pictures a person not familiar with the training environment can deduce that there is a long low wall or bench in front of the robot as well as similar structures to the immediate right and a little ways off to the left of the robot. Behind this low wall appear to be several thin, tall objects. Behind and to the left of the robot there appears to be high walls enclosing the environment. Off to the right of the robot there are two open paths that are separated by tall, thin landmarks. Finally, the overall area immediately surrounding the robot appears to be open area mostly incased by long, low walls with openings at the corners.

It is worth mentioning at this point that some error does accrue in the system during training. Odometry readings are not exact and the laser system can jostle around during movement causing some slight errors in the calibration that will effect the overall placing and orientation of landmarks that are not near the robot. It will be shown in the next section that the system can handle these errors. It is important to remember that few, if any, cognitive maps are intended to be exact spatial replications of an environment.

CHAPTER VI

SYSTEM PERFORMANCE

Cognitive map generation is only the first step towards enabling the Segway RMP with the ability to localize itself within its environment and navigate from point to point. From the internal cognitive map, the RMP must be able to identify particular sets of landmarks and relate those to landmarks detected from a single 3-D laser scan taken at a random location. The RMP must also be able to utilize its cognitive map to plan a path through the environment that avoids obstacles and reaches a goal state.

Localization

Localization in mobile robots is a very old problem with many solutions. Possessing only knowledge about the placement of landmarks in the environment and the positioning of landmarks relative to the robot is the primary method of mobile robot localization. With this knowledge, two important questions arise:

1. How many landmarks does the robot detect?

The number of landmarks that the robot detects is vital for proper localization. Only detecting one landmark, at a distance $\boldsymbol{d_L}$, localizes the robot to any point on the sphere surrounding the robot with radius $\boldsymbol{d_L}$.

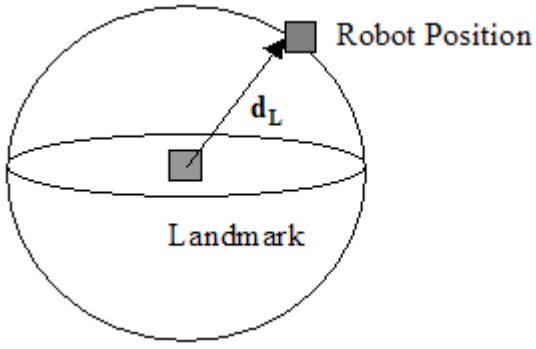

Figure 40: Localizing position from 1 landmark

Assuming that the robot can only move in two dimensions reduces the localization space to a circle around the landmark.

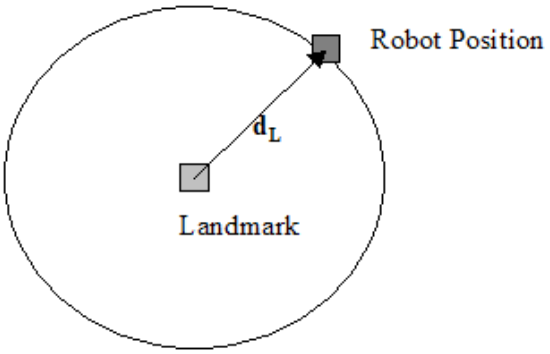

Figure 41: Localizing the robot on a 2-D plane from 1 landmark

Detecting two landmarks, at distances $\boldsymbol{d_{L1}}$ and $\boldsymbol{d_{L2}}$, localizes the robot to two points.

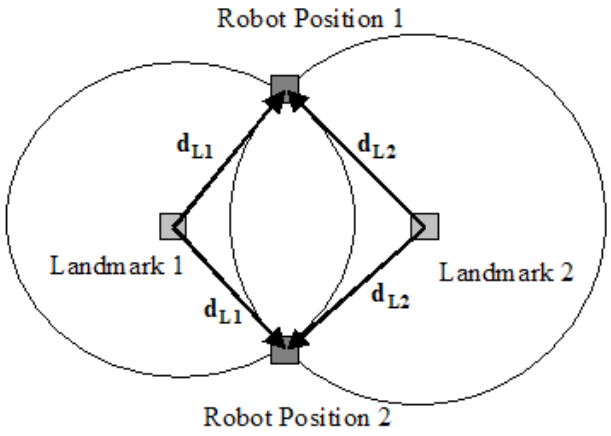

Figure 42: Localizing the robot from 2 landmarks

In order to localize to only one possible point requires the presence of at least three landmarks, at distances $\boldsymbol{d_{L1\text{-}3}}$.

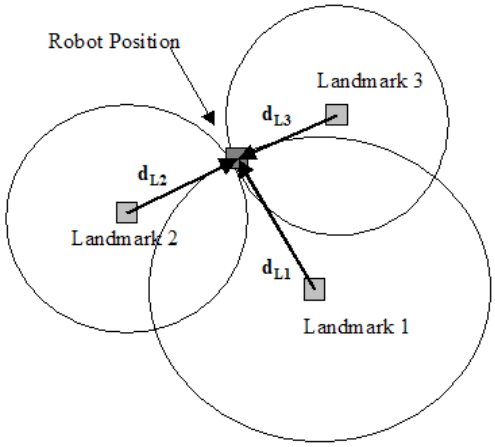

Figure 43: Localizing the robot from 3 landmarks

2. Are there features on these landmarks that make them discernable from different locations?

The number of landmarks required to localize can be reduced if more knowledge about particular landmarks is present. For example, if two landmarks can be determined to be different from each other, then the two-landmark case from above can be used for localization and the orientation of the landmarks from the robot's perspective can be used to determine which of the two possible locales the robot does actually occupy.

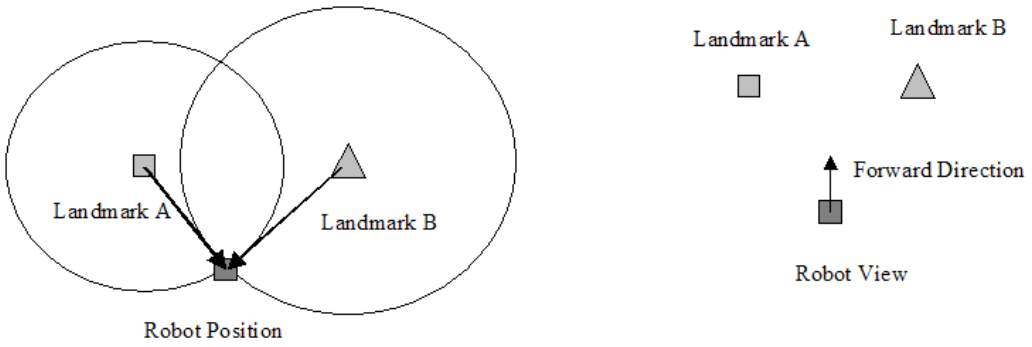

Figure 44: Knowing that landmark A is left of landmark B in the robot's view, the position of the robot can be determined

If the precise orientation of a landmark can be determined then only one landmark is necessary for localization because the robot can determine where on the circle (radius $\boldsymbol{d_L}$) around the landmark it is located by understanding exactly how the landmark appears to be oriented with respect to the robot.

For this research, landmarks were separately distinguishable through the knowledge of their 3-D location (center of mass) as well as their 3-D size and relative orientation. Due to this fact only two landmarks were required for accurate localization. Only utilizing two landmarks, though, causes another problem to arise. Since the robot has a large internal cognitive map filled with many landmarks there is a possibility that the robot can find a match for just two landmarks in a position that is not the correct position. To combat this, the robot calculates a set of possible locations and then systematically evaluates each location in that set by comparing what landmarks the robot DOES detect to the landmarks the robot SHOULD detect if the current location was the actual location. The location that matches best with the environment view the robot currently detects is chosen as the actual location of the robot.

The pseudo-code for this algorithm:

1. For each pair of landmarks detected by the robot DO:
    1. Find a matching pair of landmarks in the cognitive map
    2. If a match is found DO:
        1. Calculate the location the robot would need to be in to detect this match
        2. Store this location in a list
2. For each location in the location list DO:
    1. For each landmark detected by the robot DO:
        1. Find a possible match in the cognitive map
        2. If a match is found DO:
            1. Increase a *goodness* factor for this location
        3. If a match is not found DO:
            1. Decrease the *goodness* factor for this location
3. Return the location with the best *goodness* factor

This algorithm assumes the robot is in the environment for which it possesses the cognitive map. If the robot were to obtain a 3-D laser scan from a different environment the robot would still return the location (if one exists) with the best possible *goodness* factor, even though this number would be very low negative.

Navigation

Once the robot has localized itself within the environment the robot must also move itself around in the environment. There are many different navigation techniques for mobile robots, some of which are discussed by Murphy [10]. Because there are no sensors on the Segway RMP to detect moving obstacles, the RMP must navigate solely around stationary obstacles. Furthermore, the internal cognitive map created through the process described in this work identifies and locates stationary obstacles. Therefore, navigation of the environment can stem solely from the robots internal cognitive map. The technique applied in this thesis is a *sum of vectors* approach.

The robot calculates a path from a starting point (typically the location point determined above) to an ending point by repeatedly summing incremental vectors pointing from the starting point to the ending point with vectors pushing away from obstacles or landmarks in then environment.

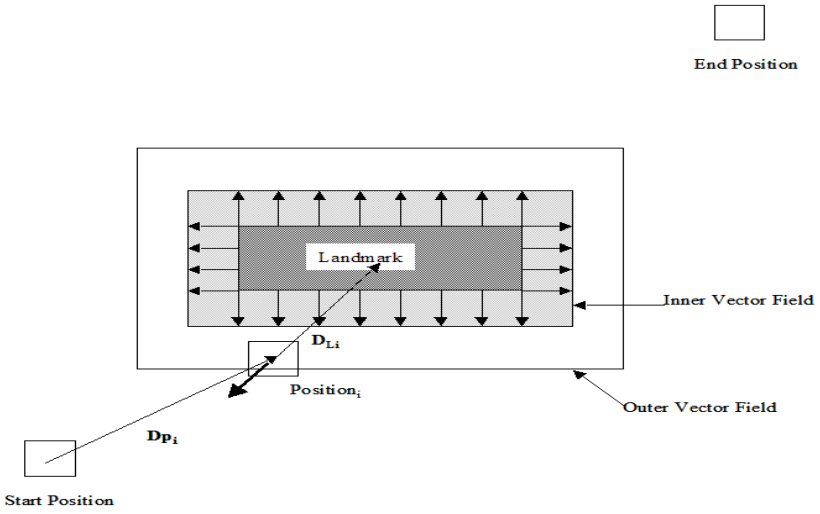

Figure 45: The robot moves towards an end position while being repelled by landmarks

This technique, for each landmark, creates two vector fields. The first vector field extends from the landmark two meters in each direction and pushes the robot back a small fraction of the distance $\boldsymbol{D_{Li}}$ in the opposite direction as $\boldsymbol{D_{Li}}$. The second vector field extends from the landmark one half meter in each direction and repels the robot back away from the landmark. The end result is that the robot moves directly to the end position until it reaches a landmark. Upon reaching a landmark the robot moves along the side of the landmark until it can resume moving towards the end position. The robot then resumes moving directly to the end position.

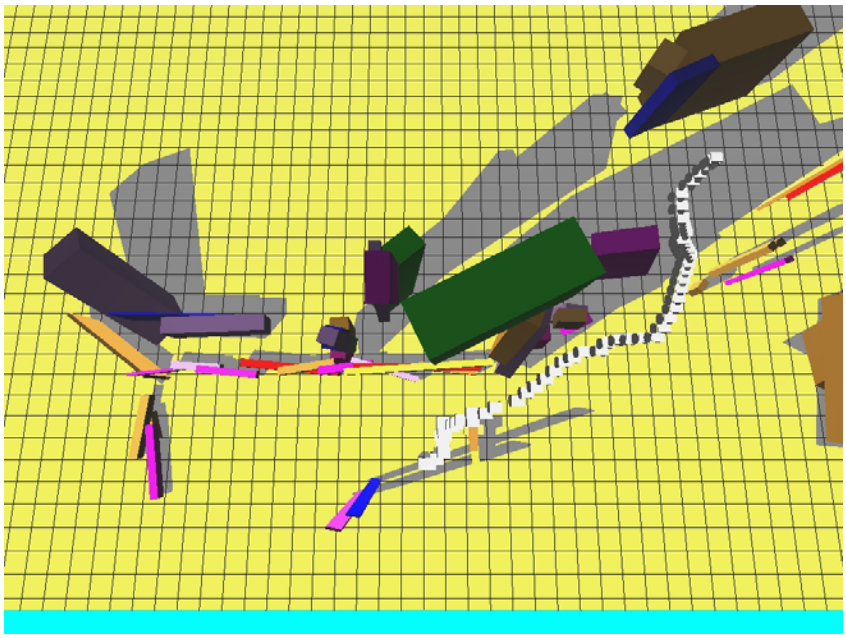

Figure 46: Each increment step the robot takes while navigating between two points and avoiding obstacles

Results

The results are presented and discussed as follows:

1. Generation of cognitive map
2. Localization from sample laser scans
3. Navigation through internal cognitive map

Generation of Cognitive Map

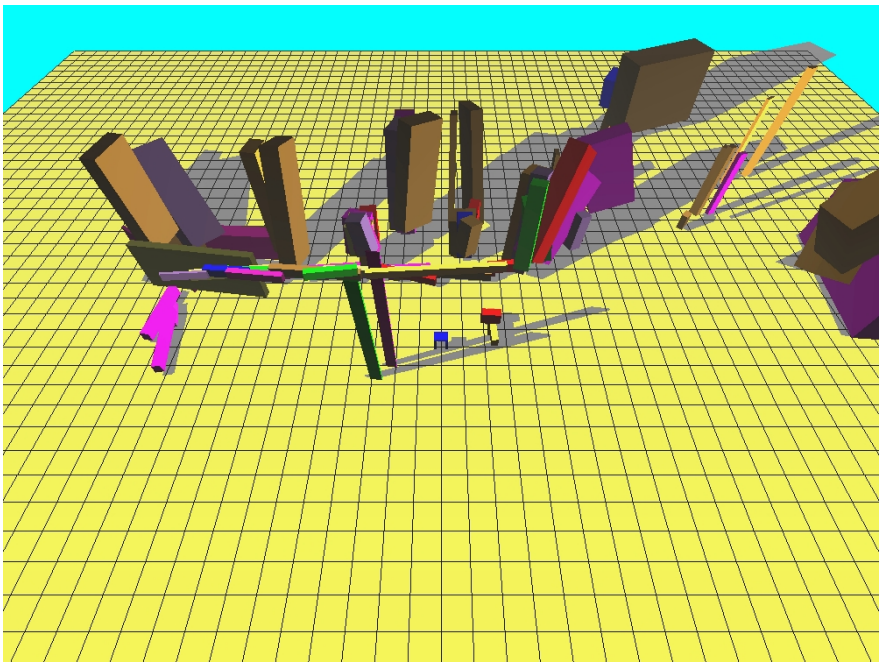

Figure 47: The entire cognitive map generated from all of the training data

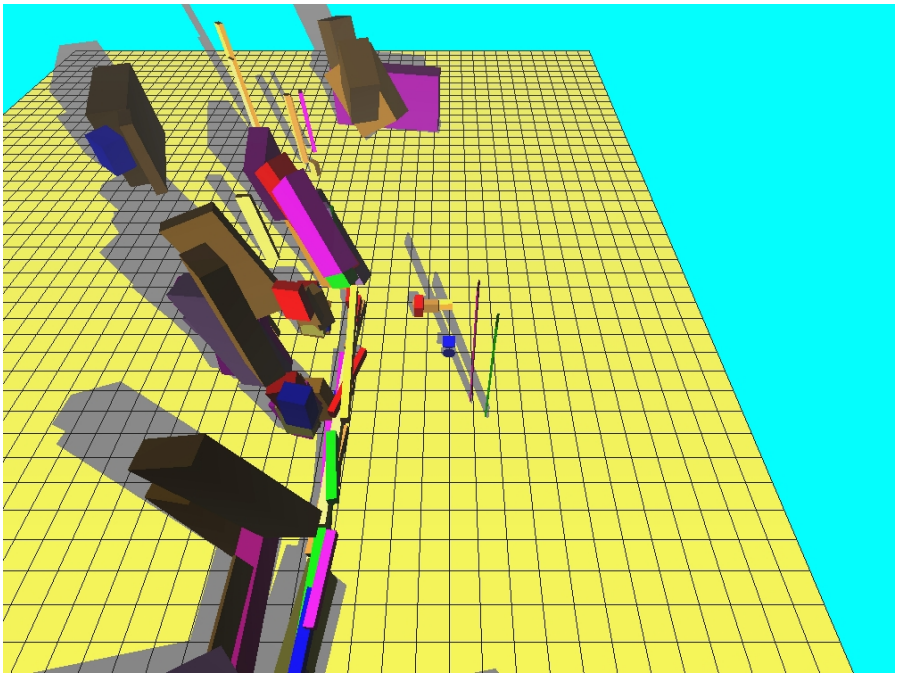

Figure 48: Side view of the entire cognitive map

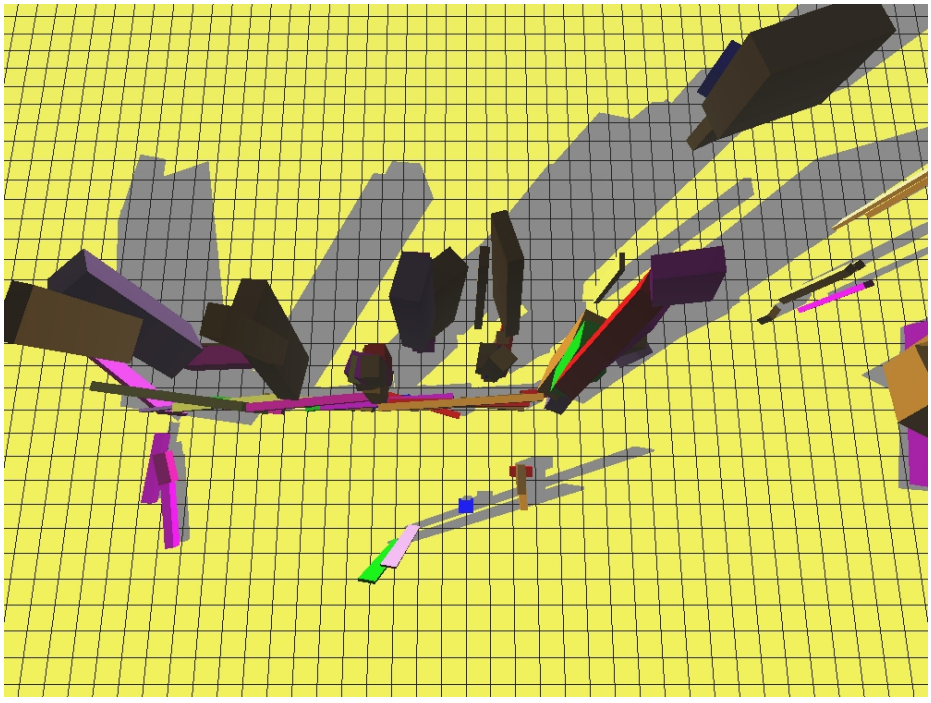

Figure 49: Aerial view of the entire cognitive map

The generation of the final cognitive map went very much according to design. However, there were some interesting occurrences that are worth noting.

1. Vertical landmarks were detected and placed very well
2. Landmarks that extend past the range of the SICK laser were detected well, but over successive scans, their center of mass tends to move due to the emergence of the sections that were previously out of range.
3. Objects could appear greater than they actually were! This is due to the fact that stray tree branches can occasionally connect two separate landmarks to make them appear, during processing, as one landmark.
4. Size and orientation values were quite consistent for vertical landmarks over successive scans.
5. Size and orientation values did vary slightly for horizontal landmarks, but only in the direction that extended past the range of the laser (typically the primary direction of the landmark).
6. The overall layout of the environment as well as the simple size and shape of most landmarks was well preserved.

Localization From Sample Laser Scans

Below are a few figures demonstrating the localization algorithm. To generate these figures a laser scan was taken at random from within the environment. After processing, the localization algorithm determined the position of the robot. This position is shown in the figures as the white square. The actual position that the laser scan was taken from is shown as the blue square. The difference is also represented in the tables that follow each figure.

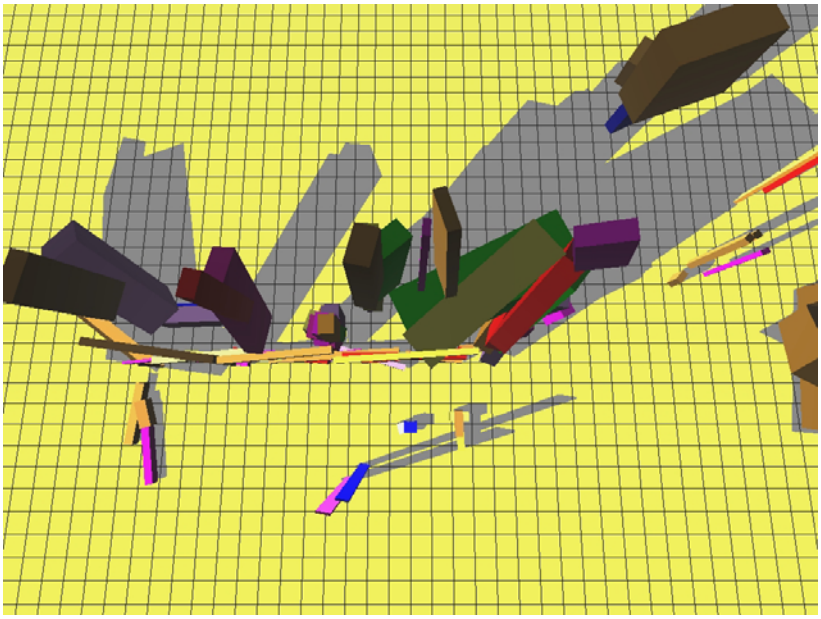

Figure 50: The blue square indicates the actual robot position. The white square indicates the position localized from the cognitive map

Table 3: Localized position versus actual position, including goodness factor, for figure 50

	Localized	**Actual**	*Goodness*
X Position	0.916 m	0.904 m	22.5
Z Position	0.296 m	0 m	
Angle	-13.8^{o}	0^{o}	

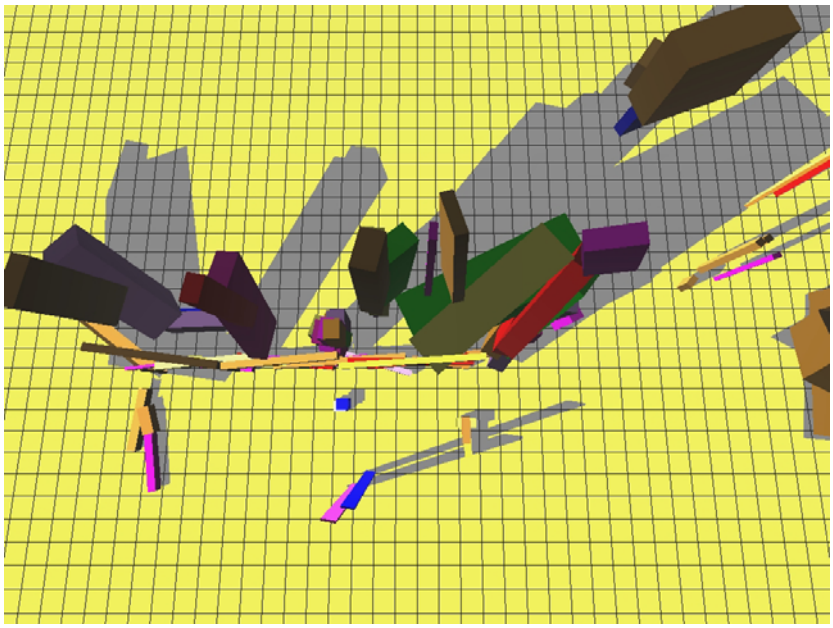

Figure 51: The white square almost perfectly coincides with the actual robot position (blue square)

Table 4: Localized position versus actual position, including goodness factor, for figure 51

	Localized	**Actual**	*Goodness*
X Position	2.917 m	2.29 m	16.0
Z Position	3.608 m	3.511 m	
Angle	-1.46^{o}	0^{o}	

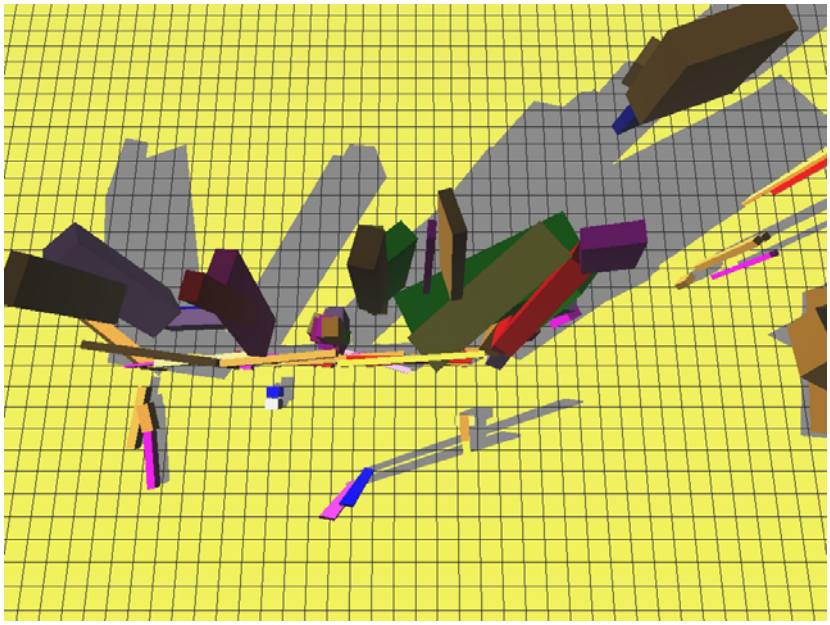

Figure 52: Sample localization towards the left of the courtyard

Table 5: Localized position versus actual position, including goodness factor, for figure 52

	Localized	**Actual**	*Goodness*
X Position	2.175 m	2.748 m	3.5
Z Position	6.779 m	6.743 m	
Angle	-0.69^{o}	0^{o}	

The localization algorithm works best when it can use two vertical landmarks for positioning. This is due to the fact that the placement of vertical landmarks does not change over successive laser scans, whereas the placement of horizontal landmarks may change over successive scans (see **Generation of Cognitive Map**). In addition, because of this fact, environments that work best for localization are environments that have vertical landmarks distributed throughout. Many outdoor environments do possess a useful dispersement of vertical landmarks. Combined with occasional horizontal landmarks this system could work well in many environments. It would be important for the trainer, however, to recognize the dispersement of vertical landmarks while training and make sure to utilize them as much as possible (i.e. do not avoid taking scans of areas with many vertical landmarks such as trees simply because the robot could not navigate those areas).

Navigation Through Internal Cognitive Map

Once localization had been accomplished, it was decided to simply demonstrate the robots navigation through its own internal cognitive map rather than through the actual environment. Since the cognitive map was an accurate depiction of the stationary obstacles in the local environment around the robot and since the robot had no further sensors to detect non-stationary obstacles while navigating, demonstrating proper navigation through the cognitive map would effectively show how the robot would navigate if extra sensors were installed.

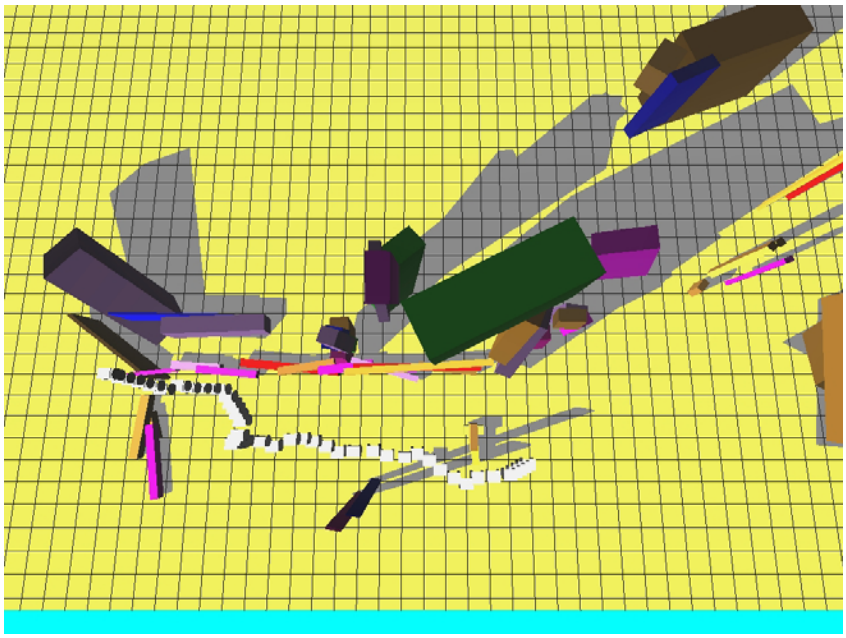

Figure 53: Robot navigating around obstacles within the cognitive map

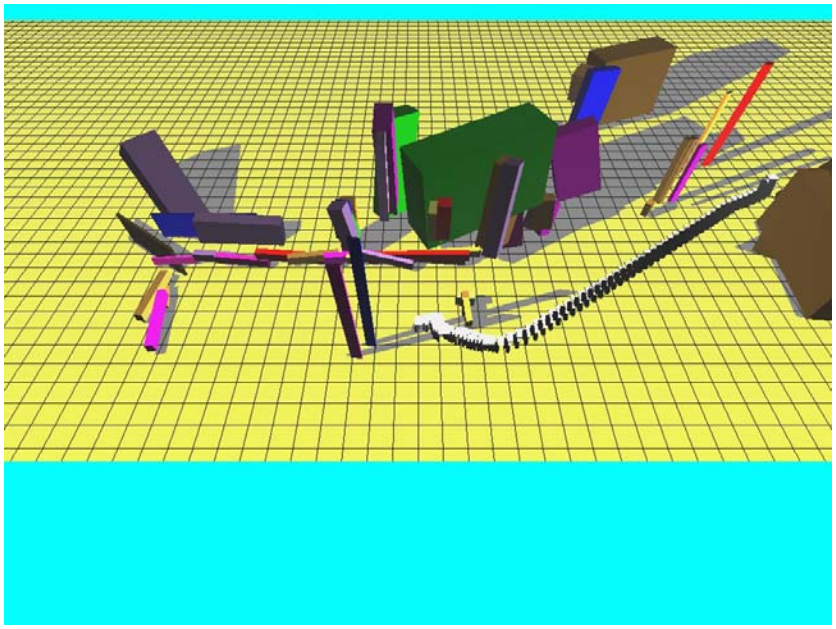

Figure 54: A second navigation demonstrating that once the robot overcomes the first obstacle it can move directly to the goal

It was discovered that occasionally the robot would come too close to obstacles while navigating. This is due to the fact that single objects in the actual environment are sometimes represented as a collection of objects in the cognitive map. The problem that arises is that the vector fields from these objects can cancel each other out. Obviously, if the robot were equipped with more sensors to find landmarks between successive steps in the navigation process, this problem could be avoided.

CHAPTER VII

CONCLUSIONS

Contributions

Much research has been conducted on the generation of maps for mobile robots. The approach in this thesis has outlined a method to generate a cognitive map for a mobile robot in an outdoor environment. Unlike most previous work in this area, this work is accomplished not by using two 2-D laser scanners but by using one rotating 2-D laser scanner. In addition, the methods used for processing these laser scans are both simple and robust. This work also discusses simple localization and navigation routines that the robot can use for the accomplishment of some task.

In order for the localization algorithm to work properly for a sample laser scan from the environment, it is necessary for two landmarks to be identified from that scan that can be matched with landmarks from the robots internal cognitive map. Often times it is the case that several possible locations are found to have a single matching pair of landmarks. At this point it becomes necessary to evaluate each possible location to determine whether this location could be the location of the original sample laser scan.

The navigation algorithm used in this work would require more sensors to be placed on the robot before it could be used in the real world. However, the algorithm used does work and if the step size in the path planning is kept to a minimum and the robot updates its current location as it travels with new laser scans the navigation algorithm could be applied to the real world.

Finally, this work spends much time talking about the hardware and software components that were put together to create an overall robotic system. When Vanderbilt University received the Segway RMP, there were no sensors on-board the robot, no means of controlling the robot, and no means of sharing data to and from the robot. Through the use of software packages such as NDDS, developed by RTI, and the Player robotic server, developed by researchers at www.playerstage.sourceforge.net, as well as through the integration of various components present in the Center for Intelligent Systems lab, a working robot with sensors, agents to handle control and data sharing, along with agents to represent information to a user was created.

Future Work

It is certainly hoped that the work in this thesis is extended. The idea of the cognitive map is an extremely useful idea for mobile robot navigation. This work has shown that a cognitive map can be generated easily from a single 2-D laser scanner equipped to scan 3-D. The next step would be to utilize more complex algorithms for the identification of landmarks as well as for the classification of landmarks. The information returned by a 3-D laser scan appears to be detailed enough to allow for different types of landmarks to be identified as such (i.e. trees, bushes, benches, etc are all identified as trees, bushes, benches respectively). In addition, close examination of the 3-D laser scan has revealed lines on the ground that correspond to the lines of a sidewalk (this will be discussed in Appendix A). Given the high number of points produced for a single laser scan and the current processing power of most laptop computers, specific research would need to be done to determine how best to quickly identify such features in the environment.

Following concepts used by Thrun [17], Montemerlo [9], and Surmann [15] the use of a cognitive map can be applied to indoor environments as well. In indoor environments individual landmarks would not be as useful as simple noting and recording the presence of hallways, doors, and openings. The technique used here to identify vertical and horizontal planes would be of great use in mapping 3-D hallways, doorways, and rooms.

Finally, extra sensors could be added to the Segway RMP. The landmark identification algorithm could be adjusted to incorporate input from these extra sensors while identifying landmarks. This extra information could aid in localization. The navigation algorithm could be equipped to utilize these sensors so that the robot can operate safely in the real world and avoid moving obstacles. The type of sensors added could be vision, sonar, or laser.

APPENDIX A

A CLOSER LOOK AT DATA PROCESSING

Taking a 3-D laser scan of an environment generates a lot of data. In addition, a lot of useful information about the environment is presented. Obviously not all of this information was utilized for this research. This appendix is going to discuss the data processing that did take place for the 3-D laser scans as well as present visual displays when parameters of the data processing algorithms were tweaked. This appendix will conclude by presenting some of the other forms of information present in a 3-D laser scan.

As mentioned in **Data Processing** the 3-D laser scans were initially processed to find vertical and horizontal planes in an environment. The parameters returned by this processing were an α value which related the angular offset the current plane was from horizontal and a λ value which related how smooth the current plane was. (Remember: for every point in the laser scan a plane that held that point was calculated using that point's nearest neighbors)

The graphing program used plotted points based on their α and λ values. Therefore, tweaking the tolerance at which a plane was considered horizontal or vertical and the tolerance at which a plane was considered smooth, gives some interesting results.

For the following figures the α and λ tolerances are given in the table and, as before, the different colors used are:

White = Horizontal plane

Green = Vertical plane

Blue = Does not match to plane

Red = Out of range

Table 6: α & λ values for figures 55 – 62

Figure #	α	λ
55	0.2	0.05
56	0.4	0.05
57	0.2	0.001
58	0.4	0.001
59	0.2	1.0
60	0.4	1.0
61	0.05	0.05
62	0.05	0.001

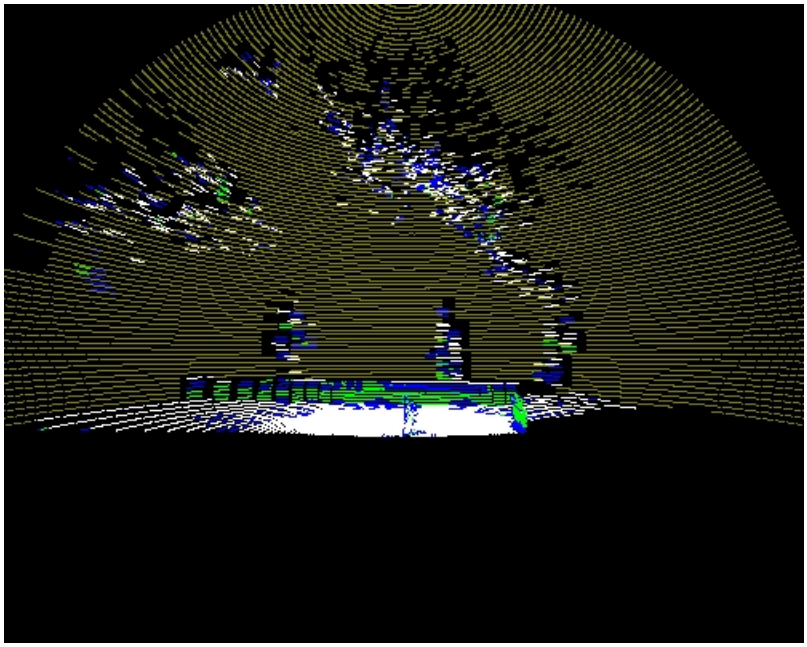

Figure 55: Planes detected when α = 0.2, λ = 0.05

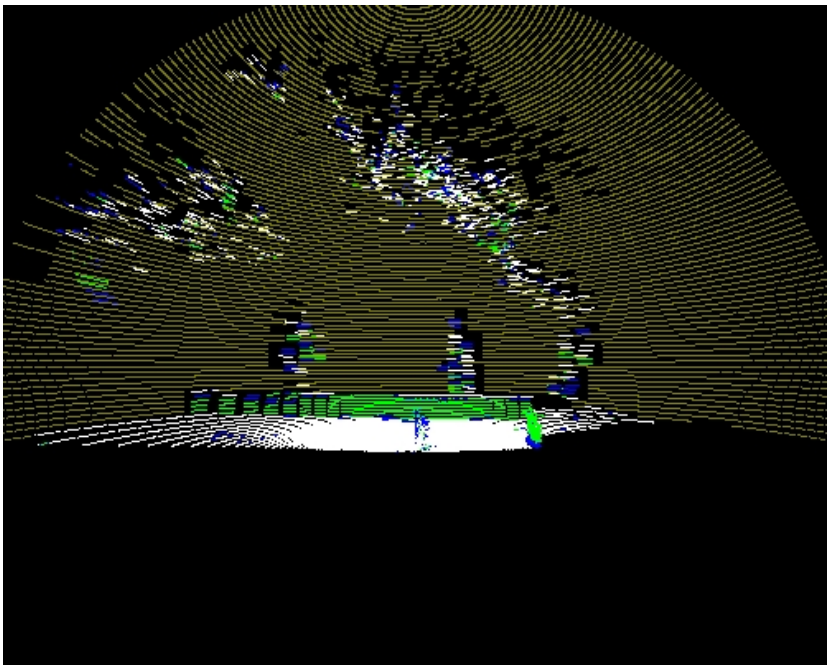

Figure 56: Planes detected when α = 0.4, λ = 0.05

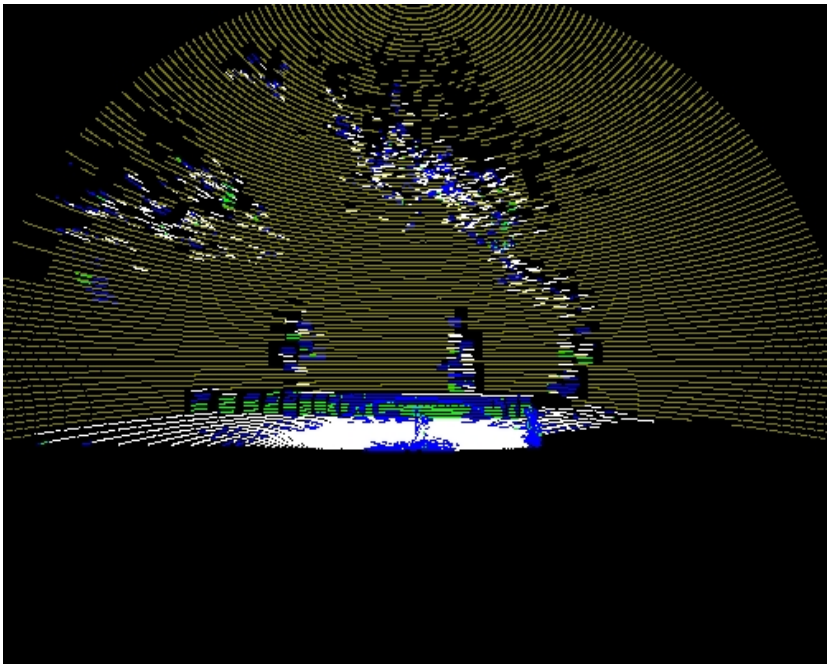

Figure 57: Planes detected when α = 0.2, λ = 0.001

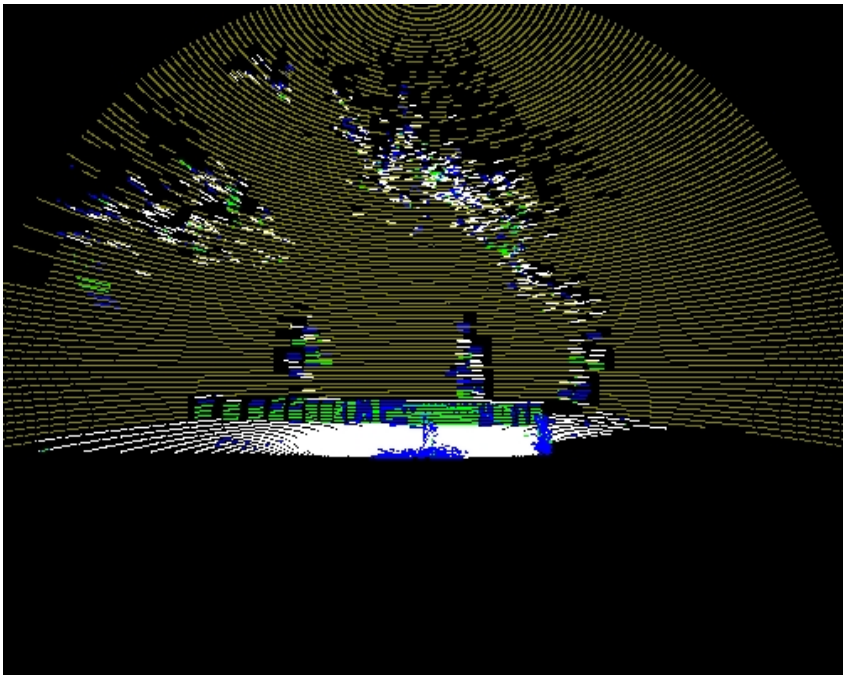

Figure 58: Planes detected when α = 0.4, λ = 0.001

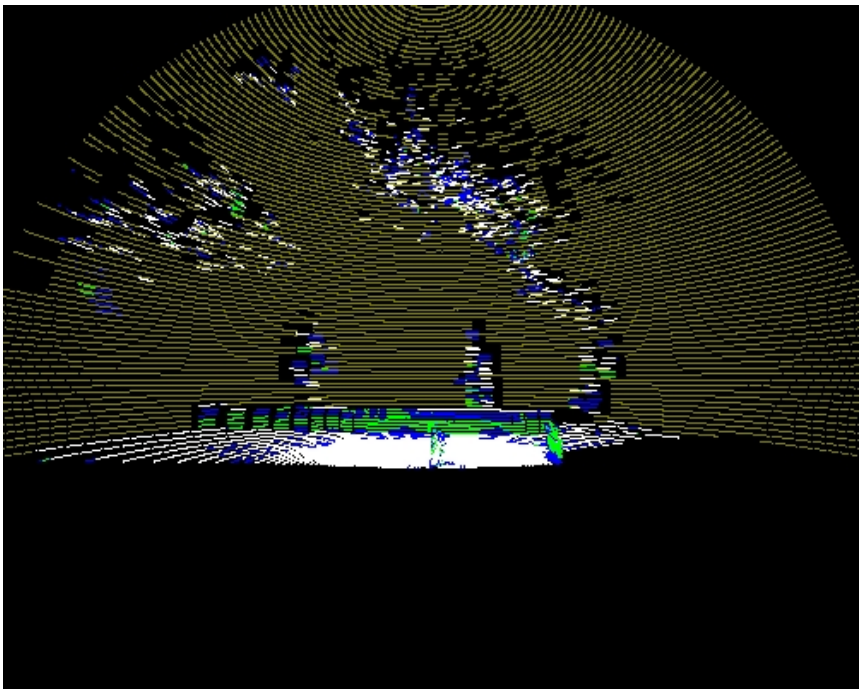

Figure 59: Planes detected when α = 0.2, λ = 1.0

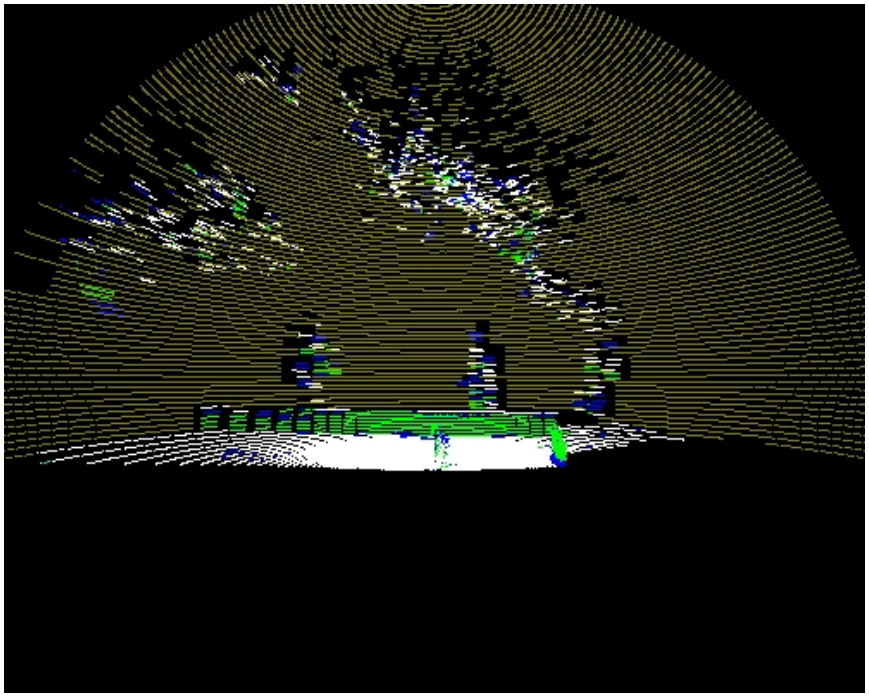

Figure 60: Planes detected when α = 0.4, λ = 1.0

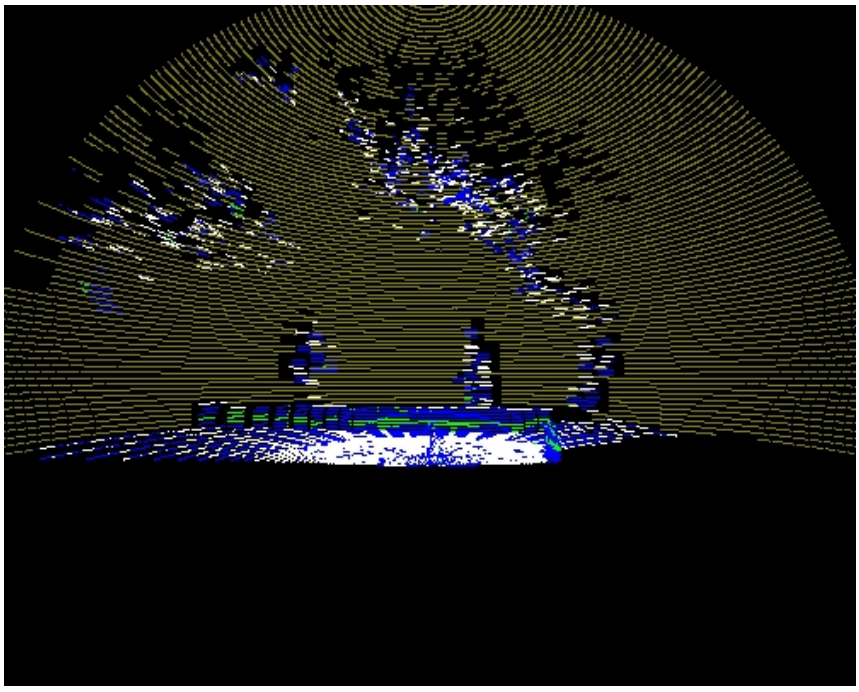

Figure 61: Planes detected when α = 0.05, λ = 0.05

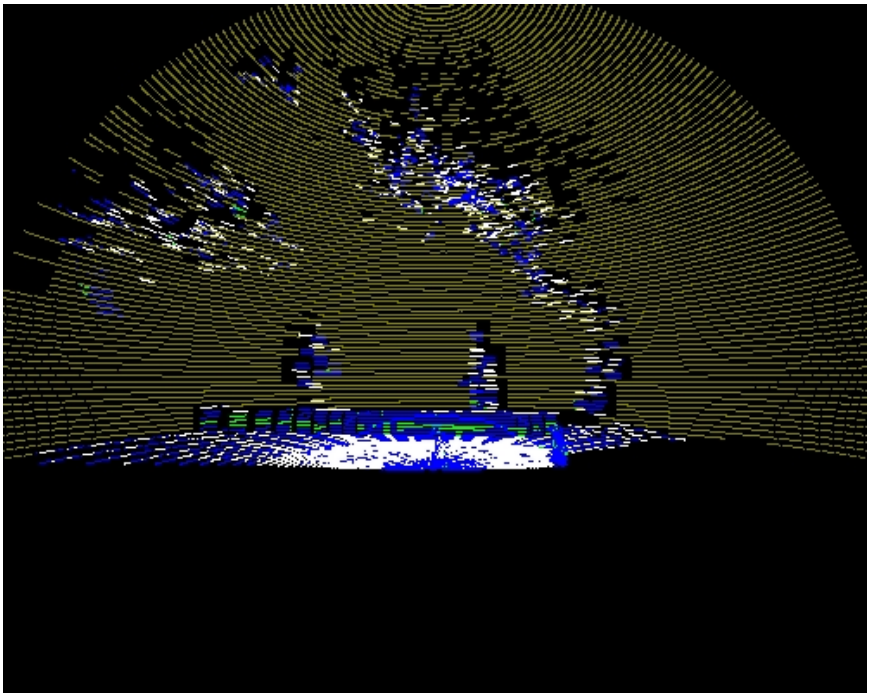

Figure 62: Planes detected when α = 0.05, λ = 0.001

It should be noted from these images that adjusting α has a greater effect on properly finding vertical and horizontal planes in the environment than adjusting λ.

The landmark identification algorithm used for this research was rather simple: it recursively found all points within a certain radius of a given home point and added these points to the current landmark. There were not many parameters in this algorithm that could be tweaked, but one parameter that did have an effect on the landmarks detected was the radius used when adding points to a landmark. Initially a very tight radius was used, approximately six inches. This value was chosen because it corresponds approximately to the linear distance between two individual laser values at maximum distance.

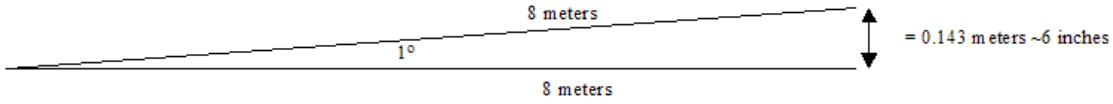

Figure 63: The linear distance between successive laser values reaches a maximum of approximately 6 inches

The algorithm, however, tended to miss landmarks, such as trees, whose points were spread out and were themselves located at least three to four meters away. The reason that this happened was that the points in landmarks such as trees are spread out over a wide area, and since the landmark was located at some distance from the laser scanner, the angular resolution caused connecting points to be missed.

Next, a larger radius was chosen that was approximately eighteen inches in length. As would be expected, though, the algorithm connected near-by landmarks.
The difference between having a radius of six inches and a radius of eighteen inches is shown below:

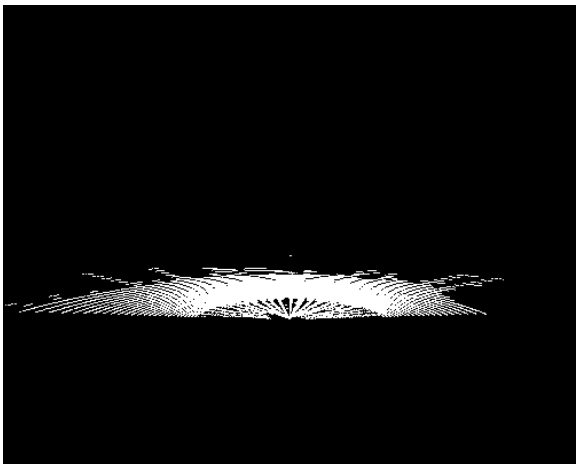

Figure 64: Floor detected with an 18 inch radius

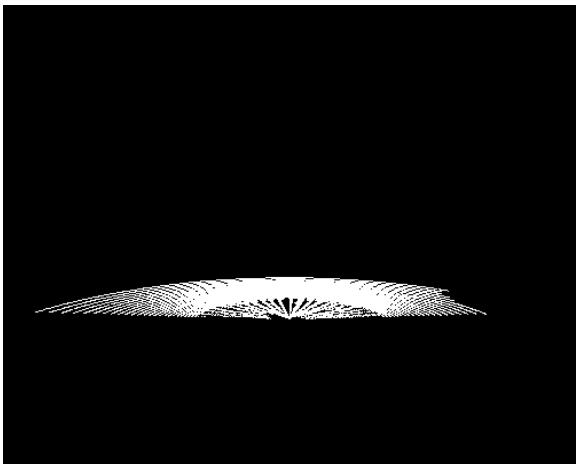

Figure 65: Floor detected with a 6 inch radius

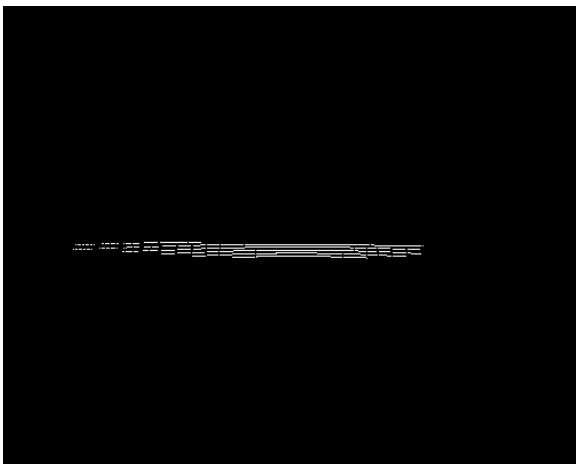

Figure 66: Bench detected with an 18 inch radius

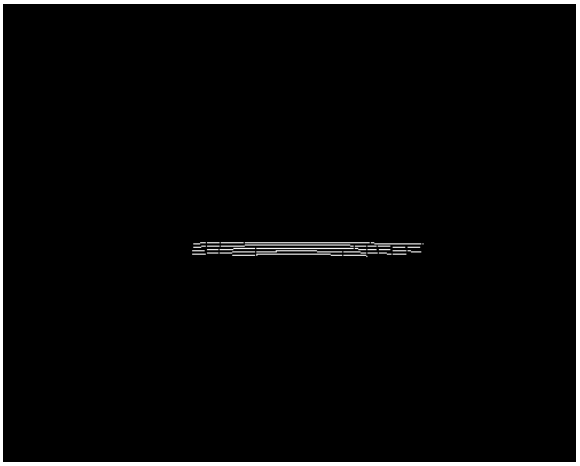

Figure 67: Bench detected with a 6 inch radius

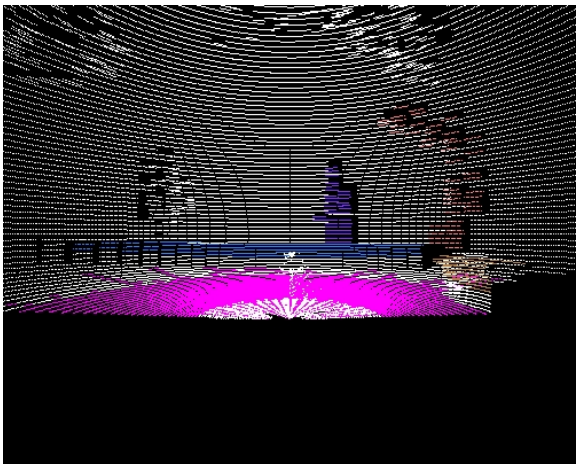

Figure 68: All of the landmarks detected with an 18 inch radius

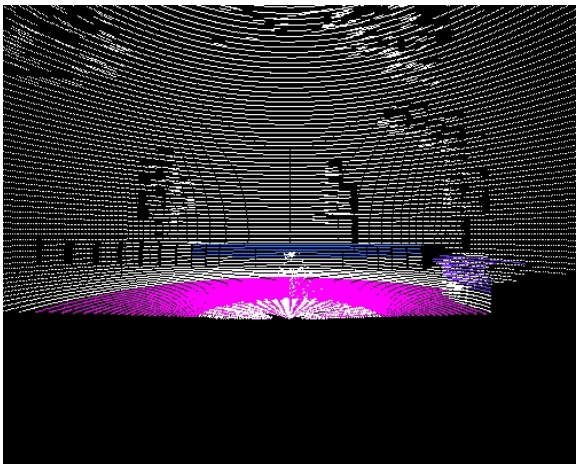

Figure 69: All of the landmarks detected with a 6 inch radius

In order to keep the landmark detection algorithm simple, it was decided to make the radius used for connecting points a linear function of the distance, from the origin, of those points. At maximum distance the radius was eighteen inches while at a distance of a half-meter the radius was six inches.

The research discussed in this thesis was only intended for outdoor environments. The research was also only intended to find usable, repeatably identifiable landmarks. From the data collected by these 3-D laser scans much more work could be done. For example, even though usable floor was identified from a laser scan, this research did not utilize this information in

navigation. In addition, it was noted that the laser scan did detect the edges of sidewalks. This information could be extremely useful in actual outdoor navigation.

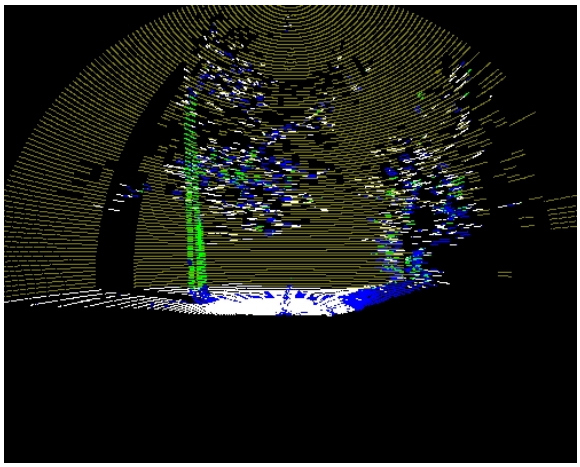

Figure 70: Normal view of an image

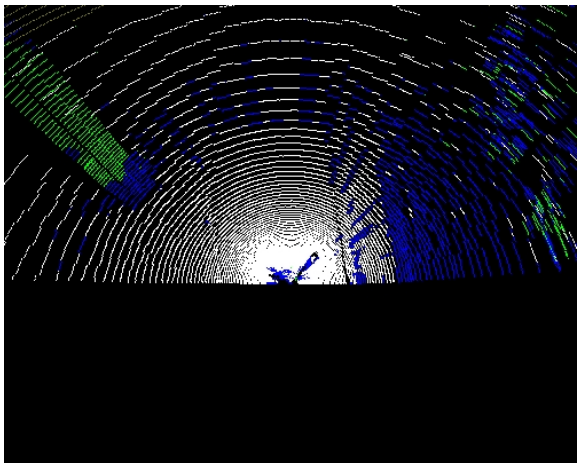

Figure 71: Aerial view of the same image showing sidewalk lines

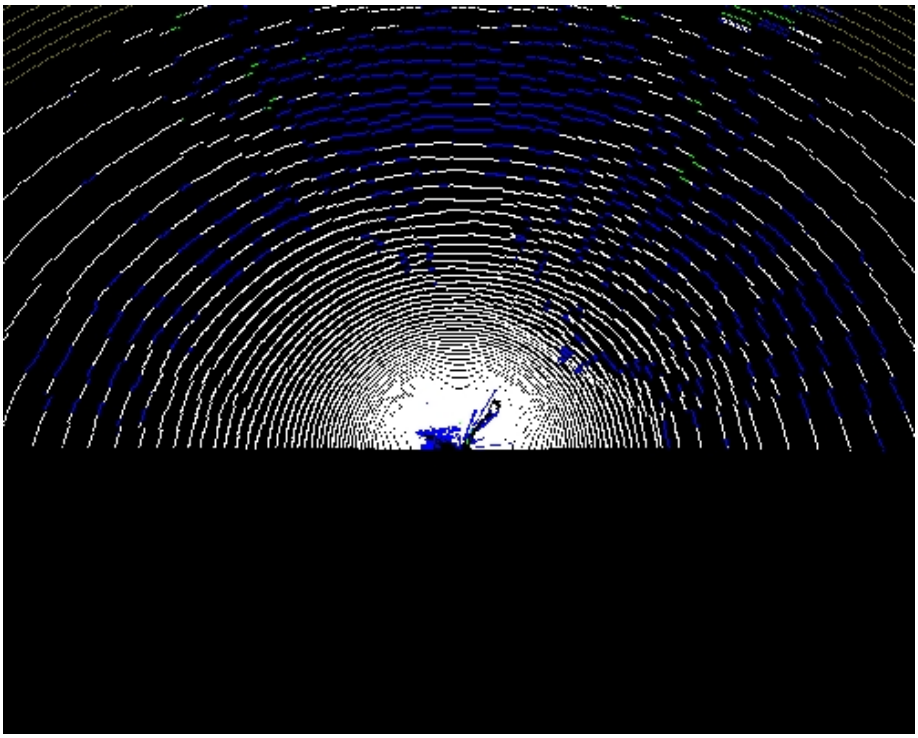

Figure 72: Aerial view showing sidewalk at an angle just greater than 45^{o}

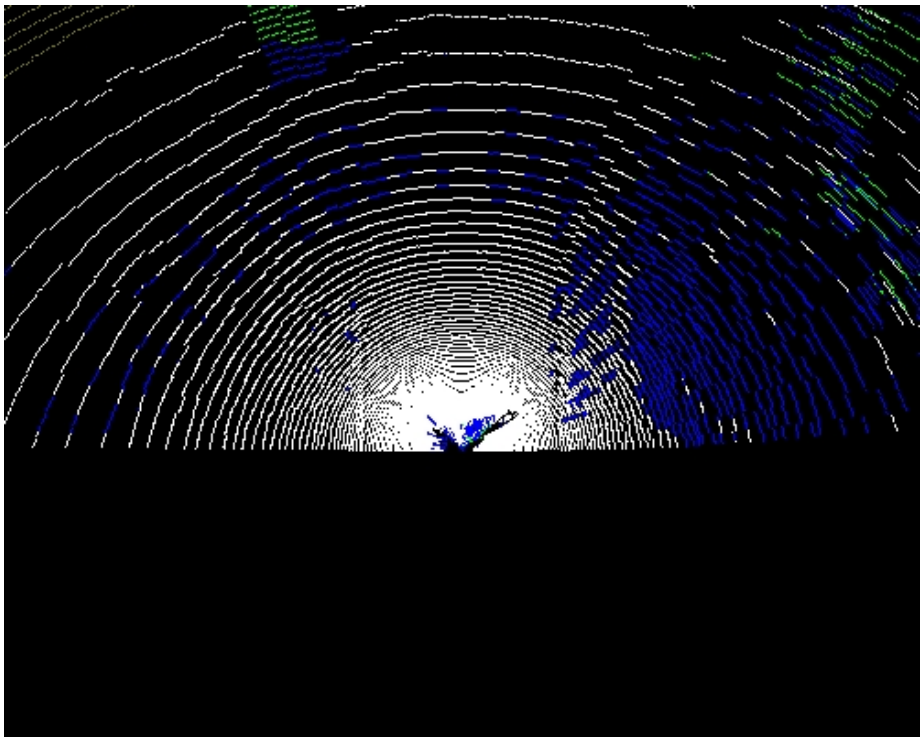

Figure 73: Aerial view showing a sidewalk that the robot is in-line with

The sidewalk lines in each figure can be seen running alongside of the robot until the density of the data is lost due to the angular resolution of the laser.

Finally, similar techniques to create block environments could be used with 3-D laser scanning to create virtual hallways and rooms from an indoor environment.

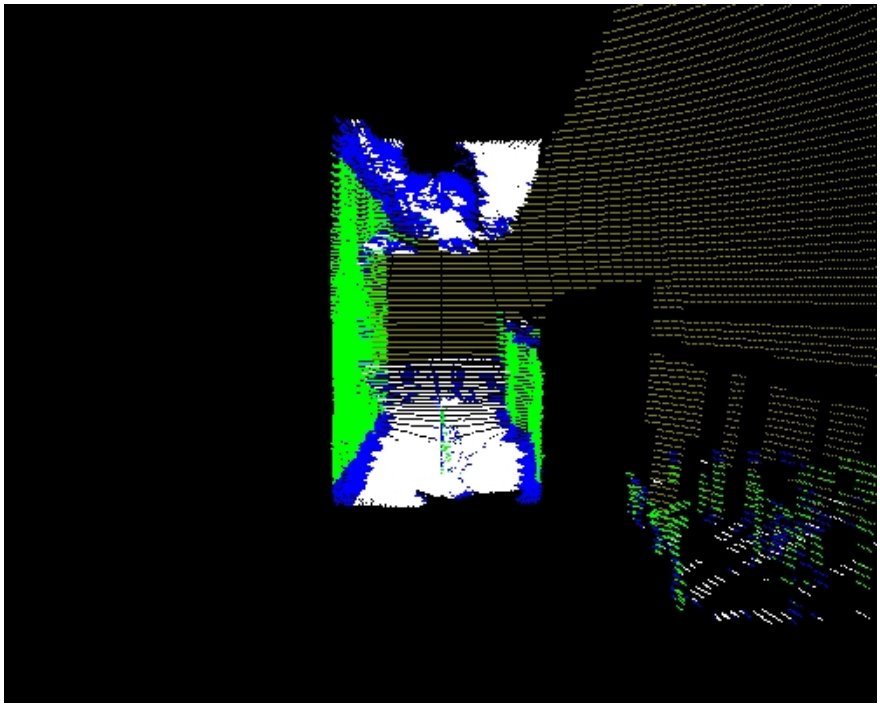

Figure 74: Hallway inside Featheringill Hall after the data was processed to identify vertical and horizontal planes

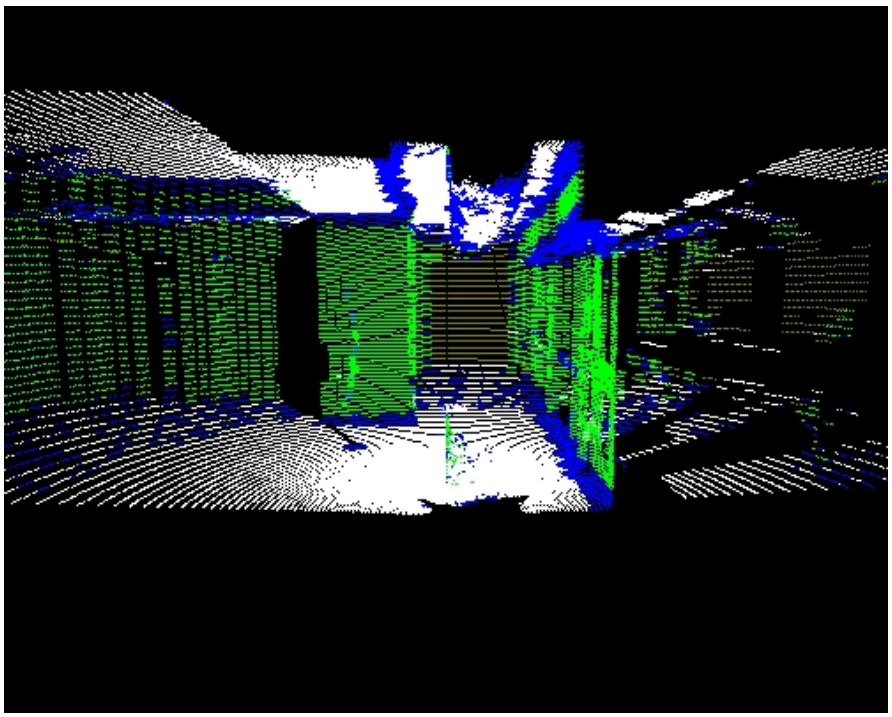

Figure 75: Hallway in Featheringill Hall leading to an open area in front of elevators

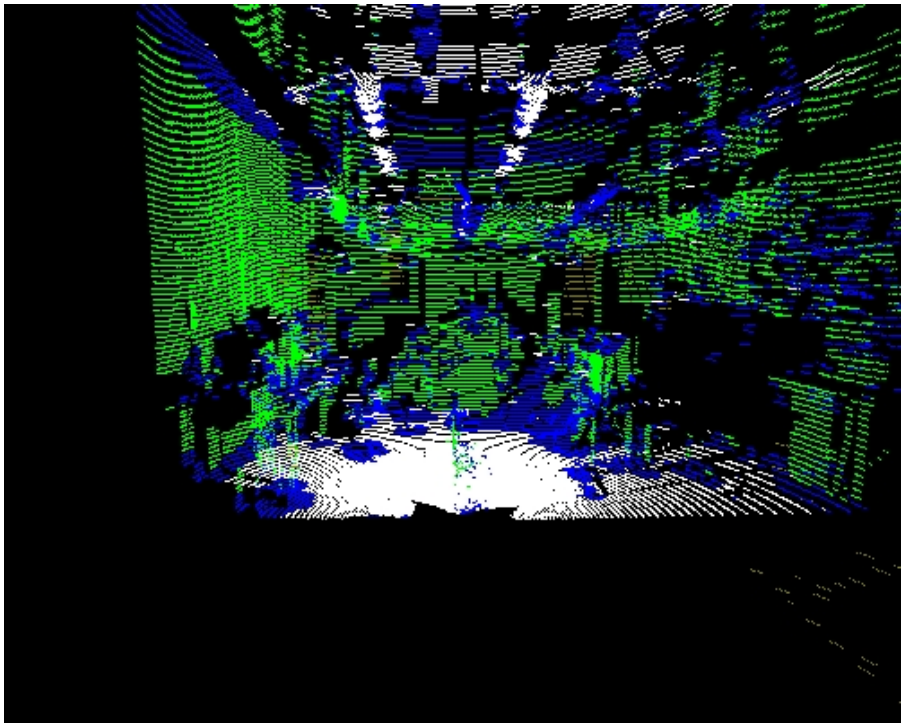

Figure 76: Inside the ISAC lab in Featheringill Hall. A lot of objects make the room seem cluttered, but the walls and floor are still well identified

The research presented in this work shows that this is possible and could be done as potential future work.

APPENDIX B

A CLOSER LOOK AT LOCALIZATION

The localization algorithm used in this work relies on the detection of at least two landmarks from a laser scan. Once detected the vector connecting the two landmarks is compared to similar vectors between landmarks in the cognitive map. From this comparison, a possible position is calculated. For each possible position a *goodness* factor is then calculated and the position with the best *goodness* factor is returned as the position of the robot.

This appendix will discuss in further depth the results returned by the localization algorithm as well as the *goodness* factor calculated for possible locations. Occasionally, there were not enough landmarks in a 3-D laser scan to accurately localize the robot. The results for these cases will also be discussed.

Once a pair of landmarks was identified with in a 3-D laser scan, the vector between them was calculated.

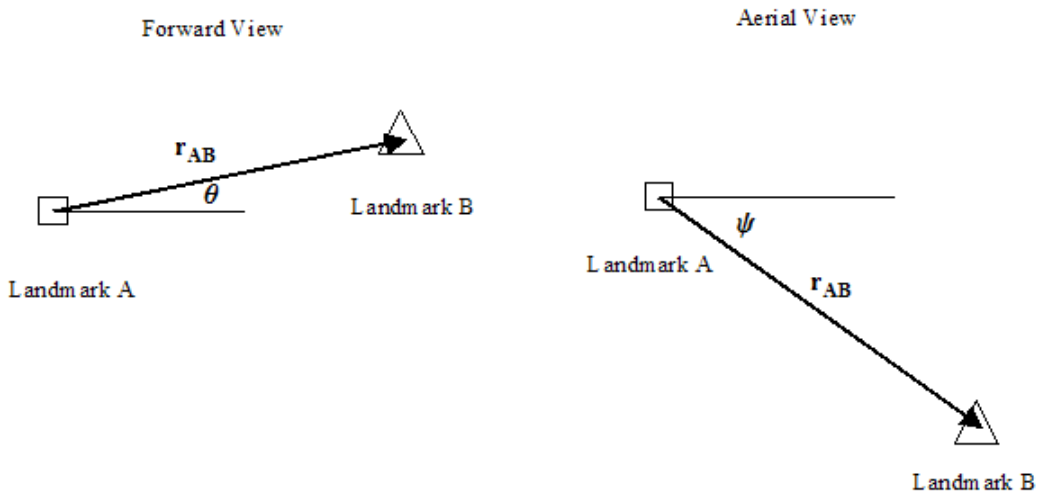

Figure 77: Vector $\mathbf{r}_{AB}$ with orientation angles θ, about the x-axis, and ψ, about the y-axis, connects the two landmarks

This vector was systematically compared to all vectors between pairs of landmarks in the cognitive map. For a match to exist two criteria had to be present:

1. The vectors had to be at least 85% similar
2. The landmarks had to be oriented in the same direction as their respective matches

The purpose for the second criteria mentioned was to ensure that landmark matches were consistent (i.e. tall and thin matched with tall and thin, short and long matched with short and long, etc). After a match was found the position and orientation in which the robot would need to be in order to detect the given landmark pair was calculated. For all matches found, a position and orientation of the robot was calculated.

The *goodness* factor calculated by this algorithm relates numerically how well a possible position as well as the landmarks detected at that position fit into the cognitive map. If a landmark appears in the laser scan that does not appear in the cognitive map, the *goodness* factor is decreased. In addition, if a landmark does not appear in the laser scan but does appear in the cognitive map, the *goodness* factor is decreased. For every landmark that does appear in the laser scan and is matched to a landmark, by position and orientation information, with in the cognitive map, the *goodness* factor is increased. Maintaining the *goodness* factor in this way allows for the possibility that features in the environment may change slightly, such as the addition or removal of landmarks, but the localization algorithm still has a likelihood of working properly. The *goodness* factor can also be considered a form of confidence factor. If the *goodness* factor is negative, the robot should consider not being very confident in its calculated location and should move slightly and take a second scan of the environment with which to localize from.

Below are several figures showing sample localizations. A table that presents the calculated localizations for the three best *goodness* factors follows each figure. Again, the blue robot indicates the actual position and the white robot indicates the localized position. NA means that further possible localizations were not found.

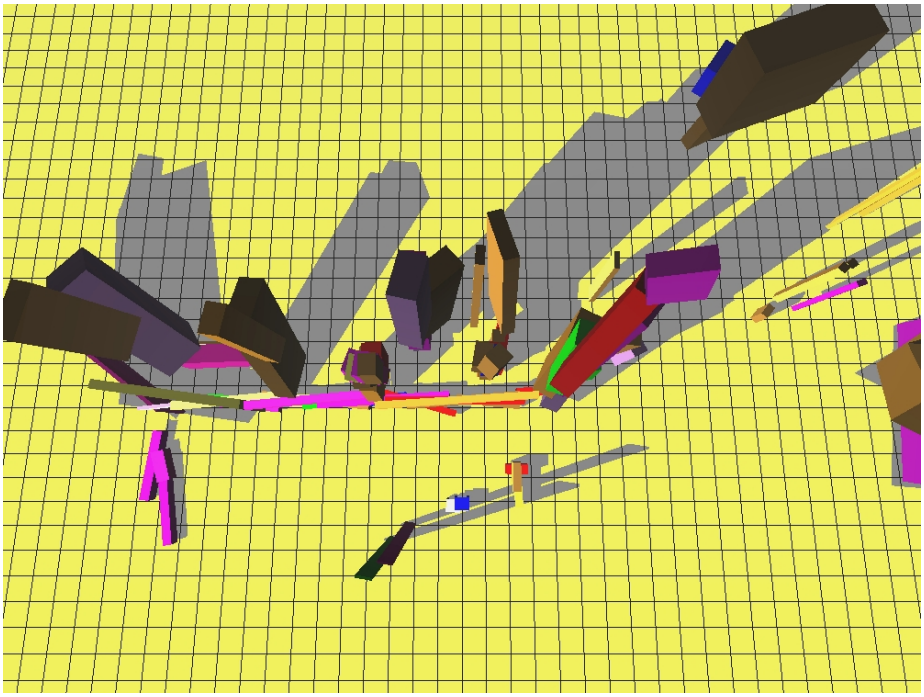

Figure 78: Localizing the robot with small error

Table 7: Three best localized positions, including goodness factor, versus actual position for figure 78

	X Position	**Z Position**	**Orientation**	*Goodness*
Actual Position	0 m	0 m	0^o	
Localized 1	-0.079 m	0.361 m	4.03^o	3
Localized 2	-0.107 m	0.697 m	-0.47^o	0.5
Localized 3	0.061 m	0.785 m	7.03^o	0

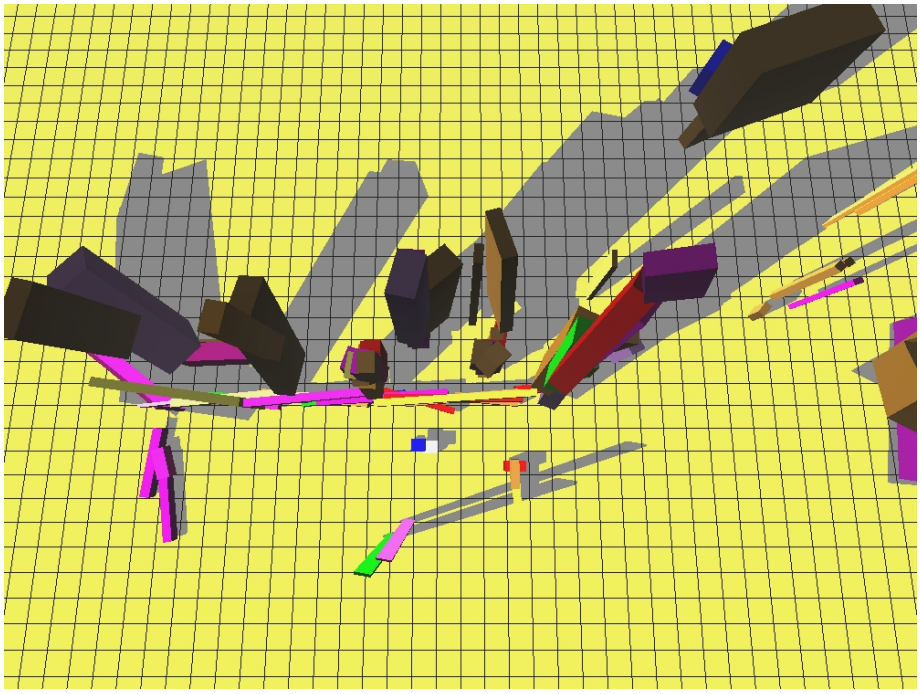

Figure 79: Again, some small error can arise in localization

Table 8: Three best localized positions, including goodness factors, versus actual position for figure 79

	X Position	**Z Position**	**Orientation**	*Goodness*
Actual Position	2.29 m	1.81 m	0^o	
Localized 1	2.21 m	1.29 m	0.24^o	23.5
Localized 2	2.18 m	1.11 m	-5.0^o	22.0
Localized 3	2.22 m	0.439 m	1.49^o	21.5

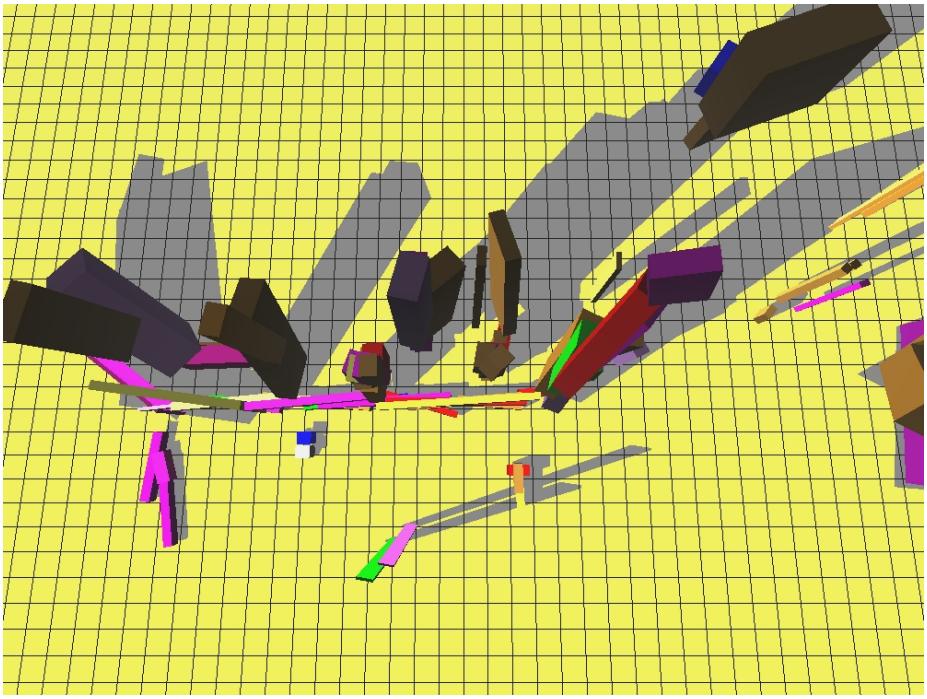

Figure 80: Localizing on the left side of the courtyard outside Featheringill Hall

Table 9: Three best localized positions, including goodness factors, versus actual position for figure 80

	X Position	**Z Position**	**Orientation**	*Goodness*
Actual Position	2.75 m	6.74 m	0^o	
Localized 1	2.18 m	6.78 m	3.5^o	3.5
Localized 2	2.74 m	6.79 m	1.4^o	1.0
Localized 3	2.19 m	6.95 m	0.5^o	0.5

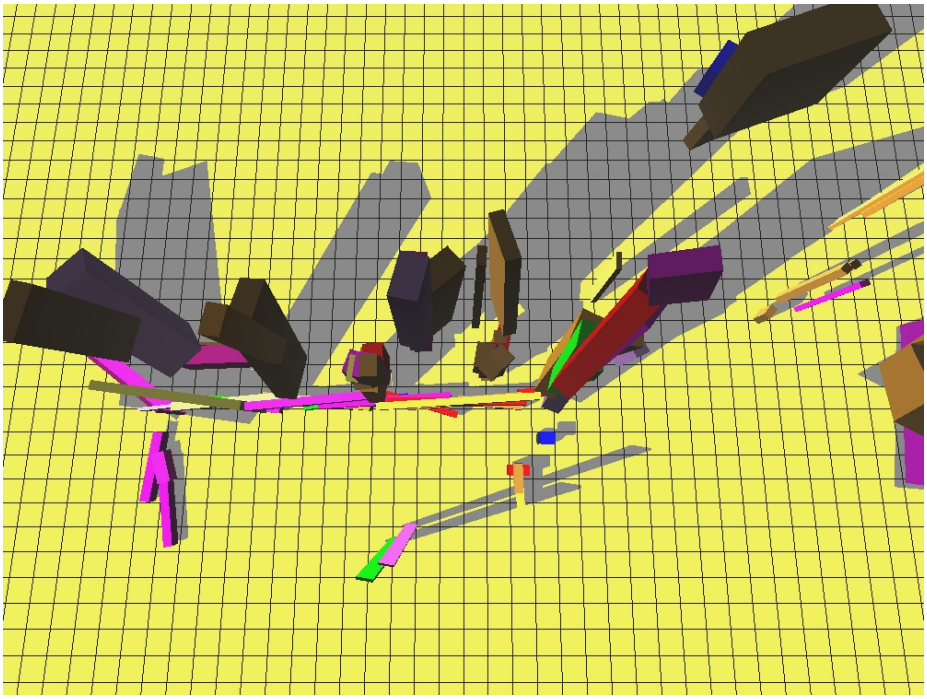

Figure 81: Localizing as the robot is moving out of the courtyard

Table 10: Three best localized positions, including goodness factors, versus actual position for figure 81

	X Position	**Z Position**	**Orientation**	*Goodness*
Actual Position	2.75 m	-3.58 m	0^o	
Localized 1	2.73 m	-3.50 m	2.3^o	14.5
Localized 2	2.65 m	-3.60 m	0.8^o	12.0
Localized 3	NA	NA	NA	NA

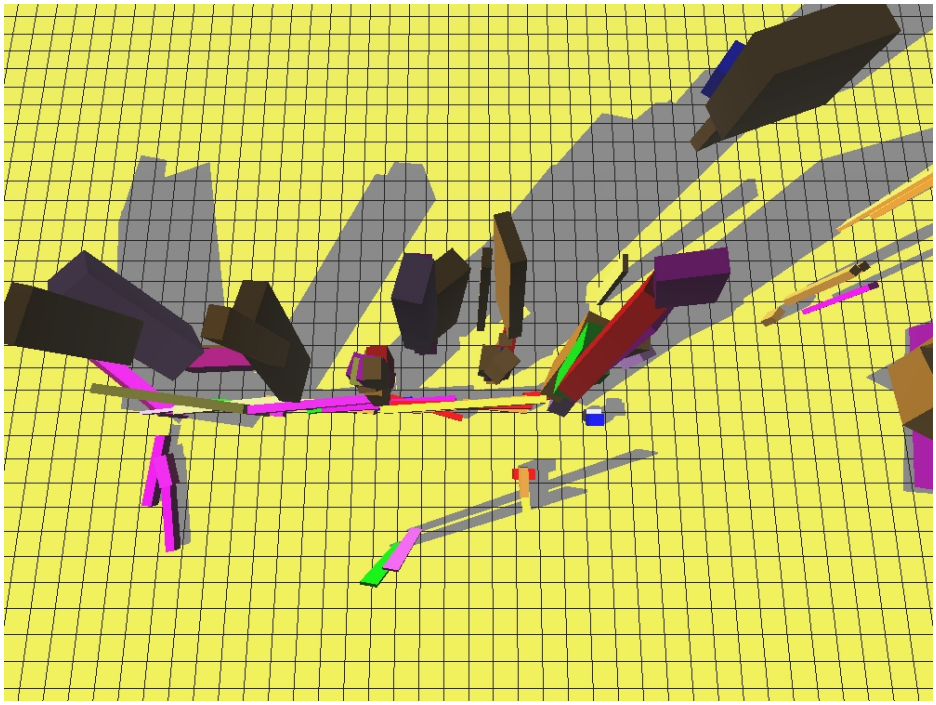

Figure 82: Localizing as the robot continues to move away from the courtyard

Table 11: Three best localized positions, including goodness factors, versus actual position for figure 82

	X Position	**Z Position**	**Orientation**	*Goodness*
Actual Position	3.66 m	-5.42 m	0^o	
Localized 1	3.85 m	-5.31 m	1.3^o	17.0
Localized 2	3.03 m	-4.82 m	-18.4^o	13.5
Localized 3	3.55 m	-5.28 m	12.1^o	12.0

It was possible for the robot to calculate a false localization. However, when using this algorithm it was hoped that knowledge gained from the calculation of the *goodness* factor would enable the robot to identify such false localizations. Knowing that a localization is false, or most likely false, the robot could then reposition itself in the environment in order to make a more accurate localization.

Below are figures that show false localizations and the corresponding *goodness* factors.

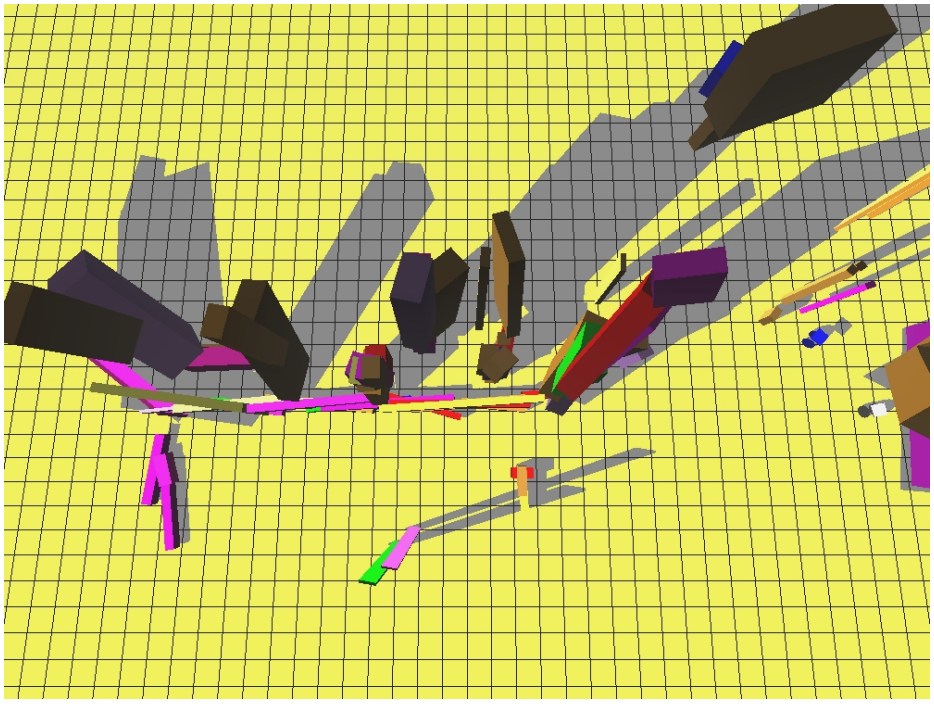

Figure 83: False localization. However, with very low goodness factors, these occurrences can be caught

Table 12: Three best localized positions, including goodness factors, versus actual position for the false localization in figure 83

	X Position	**Z Position**	**Orientation**	*Goodness*
Actual Position	7.18 m	-15.39 m	-45^{o}	
Localized 1	4.02 m	-17.52 m	-10.1^{o}	-3.5
Localized 2	NA	NA	NA	NA
Localized 3	NA	NA	NA	NA

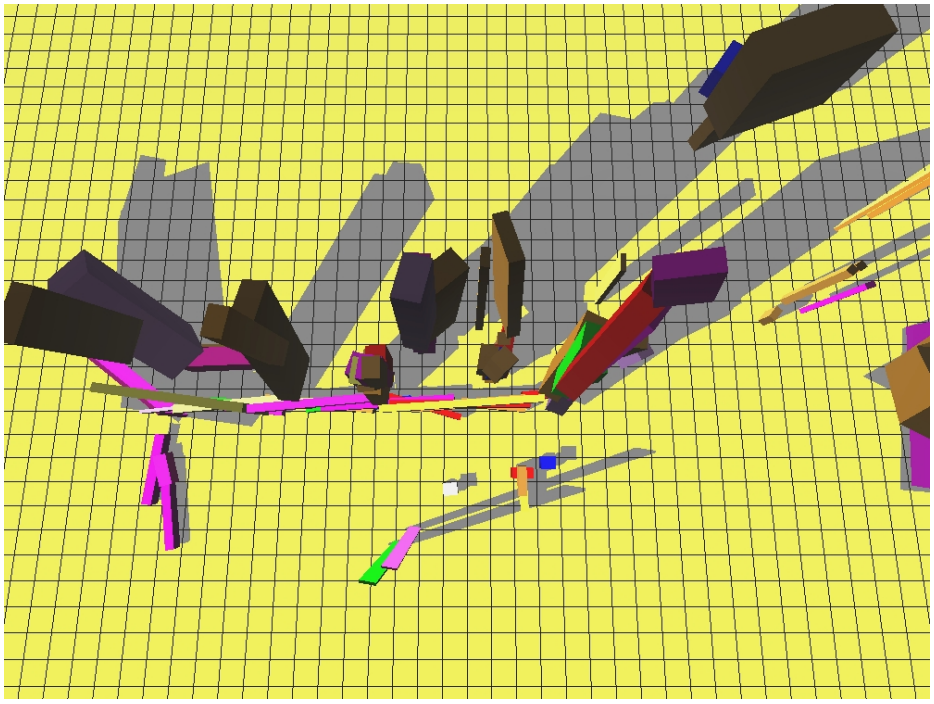

Figure 84: Another false localization

Table 13: Three best localized positions, including goodness factors, versus actual position for the false localization in figure 84

	X Position	**Z Position**	**Orientation**	*Goodness*
Actual Position	1.83 m	-3.41 m	0^o	
Localized 1	0.763 m	0.693 m	-8.5^o	-8.5
Localized 2	0.741 m	0.729 m	-7.1^o	-10.0
Localized 3	0.950 m	-1.35 m	4.4^o	-10.5

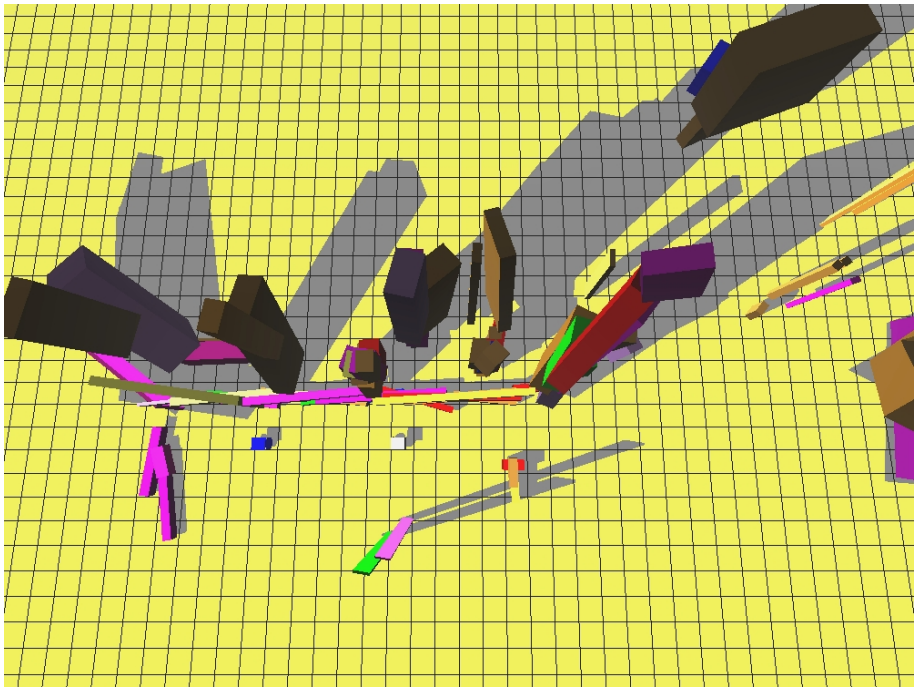

Figure 85: False localization can occur near places that had previously been localized correctly, but the goodness factor does not lie

Table 14: Three best localized positions, including goodness factors, versus actual position for the false localization in figure 85

	X Position	**Z Position**	**Orientation**	*Goodness*
Actual Position	2.29 m	8.55 m	0^o	
Localized 1	2.30 m	2.62 m	1.1^o	-11.5
Localized 2	2.30 m	2.62 m	12.3^o	-11.5
Localized 3	5.36 m	3.89 m	43.7^o	-15.0

REFERENCES

[1] Albus, J. S., *Outline for a Theory of Intelligence*, IEEE Transactions on Systems, Man and Cybernetics, 21, 743-509, 1991

[2] Dagan, R., "Cognitive Mapping" http://intraspec.ca.cogmap.php 2004

[3] Gerkey, B., Vaughan, R., and Howard, A., "Player/Stage Project", http://playerstage.sourceforge.net, 2003

[4] Hauser, M., Wild Minds: What Animals Really Think Henry Holt & Company, 2000

[5] Jefferies, M. E., Yeap, W.K., Smith, L., and Ferguson, D., *Building a Map for Robot Navigation Using a Theory of Cognitive Maps* in IASTED Proc. Artificial Intelligence and Applications, 2001

[6] Kamil, A., and Jones, J., "The seed-storing corvid Clark's nutcracker learns geometric relationships among landmarks", http://www.nature.com/cgi-taf/DynaPage.taf?file=/nature/journal/v390/n6657/abs/390276a0_r.html&dynoptions 1997

[7] Kawamura, K., Koku, A. B., Wilkes, D. M., Peters, R. A., and Sekmen, A., *Toward Egocentric Navigation*, International Journal of Robotics and Automation, Vol 17, No. 4, 2002

[8] Kortenkamp, D. M., *Cognitive maps for mobile robots: A representation for mapping and navigation* Ph.D. Dissertation, University of Michigan, 1993

[9] Montemerlo, M., Hahnel, D., Ferguson, D., Triebel, R., Burgard, W., Thayer, S., Whittaker, S., and Thrun, S*., A System for three-dimensional robotic mapping of underground mines* Comput. Sci. Dept., Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CS-02-185, 2002

[10] Murphy, R., Introduction to AI Robotics MIT Press, 2000

[11] Peters, R. A., Hambuchen, K., Kawamura, K., and Wilkes, D. M., *The Sensory Ego-Sphere as a Short-Term Memory for Humanoids*, Proc. IEEE-Ras Int'l Conf. On Humanoid Robots, Waseda University, Tokyo, Japan, 451-459, 2001

[12] Real-Time Innovations Inc. "Network Data Distribution Service – NDDS". http://www.rti.com

[13] Rosen, C., "Shakey the Robot" http://www.activrobots.com/HISTORY/ & http://www.sri.com

[14] Ruengcharungpong, A., *An Internet-Based Remote Manufacturing System*, Master's Thesis, Vanderbilt University, 1999

[15] Surmann, H., Lingemann, K., Nuchter, A., and Hertzberg, J., *A 3D laser range finder for autonomous mobile robots*, Proc. 32nd ISR, 153-158, 2001

[16] Thrun, S., *Learning metric-topological maps for indoor mobile robot navigation* Artificial Intelligence, 99(1), 1998, 21-71

[17] Thrun, S., Martin, C., and Liu, Y., *A Real-Time Expectation-Maximization Algorithm for Acquiring Multiplanar Maps of Indoor Environments With Mobile Robots*, IEEE Transactions on Robotics and Automation, Vol 20, No. 3, 2004

[18] Yeap, W. K., and Jefferies, M. E. *Computing a Representation of the Local Environment. Artificial Intelligence*, 107, 1999, 265-301