System Health Awareness in Total-Ionizing Dose Environments

By

Zachary Diggins

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

DOCTOR OF PHILOSOPHY
in

Electrical Engineering
December, 2016
Nashville, Tennessee

Approved:

Eric Barth, Ph.D.
Gabor Karsai, Ph.D.
Robert Reed, Ph.D.
Ron Schrimpf, Ph.D.
Arthur Witulski, Ph.D.

I would like to thank my committee, the many Vanderbilt faculty members, and other students that have mentored, taught me, and collaborated with me through my time at Vanderbilt starting as an undergraduate and through completing my PhD. You have guided me from the basics of how a transistor operates and introductory circuits to the work presented in this document. The ideas, curiosity, and academic rigor present in this great community has highly impacted my work.

I would also like to thank my parents and my brothers. Their continuous support was invaluable throughout this time. Finally, I would like to thank my wife, Kirsten, for her support during this work. Completed our PhDs together is a special memory that I greatly cherish, and her example continuously motivated me.

ACKNOWLEDGEMENTS

I would like to thank the Defense Threat Reduction Agency for their sponsorship of this work, without whose support this work would not have been possible. I would also like to thank Mike Freeman for accommodating our extensive use of his lab facilities.

Specifically, for the projects presented in this work, I would like to acknowledge my co-authors on this project: Nagabhushan Mahadevan, E. Bryn Pitt, Daniel Herbison, Gabor Karsai, Brian D. Sierawski, Eric J. Barth, Robert A. Reed, Ronald D. Schrimpf, Robert. A. Weller, Michael L. Alles, and Arthur Witulski. Their contributions are integrated throughout this work. I would like to especially acknowledge Dr. Witulski for his mentorship throughout my PhD work and Nag for always being available to dig deep into the ideas we were exploring.

I would also like to acknowledge Dr. Bhuva for his mentorship of my master's thesis work and for all the training that enabled me to successfully complete this work. Finally, I would like to acknowledge Dr. Fleetwood for his mentorship in general and specifically for his example of the attention to detail necessary for successful research.

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS iv

TABLE OF CONTENTS v

LIST OF FIGURES x

I. INTRODUCTION: PROVIDING AWARENESS 1

II. BACKGROUND: RADIATION EFFECTS FROM TRANSISTOR TO SYSTEM 5

- RADIATION AND ROBOTICS 5
- NUCLEAR POWER ROBOTICS 7
- FUKUSHIMA –DAIICHI 7
- TOTAL IONIZING DOSE AND ELECTRONICS 8
- HARDNESS ASSURANCE 10
- SYSTEM RELIABILITY AND BAYESIAN STATISTICS 10

III. TRACING RADIATION DEGRADATION 16

- INTRODUCTION 16
- SENSOR INPUT-OUTPUT RELATIONSHIP DEGRADATION 17
- EXPERIMENTAL DETAILS 19
- RESULTS 24
- DEGRADATION DIAGNOSIS 31
- IMPACT ON SYSTEM FUNCTIONAL PERFORMANCE 34
- CONCLUSIONS 35

IV. IDENTIFICATION OF HEALTH STATUS INDICATORS 39

- INTRODUCTION 39
- BACKGROUND 41
- TIMING WINDOW VIOLATIONS 42

PROPAGATION DELAYS AND TID 43
TIMING WINDOW HYPOTHESIS 44
EXPERIMENTAL DESIGN 46
EXPERIMENTAL RESULTS 48
ANALYSIS OF TID INDUCED TIMING WINDOW VIOLATIONS 51
RELATIONSHIP BETWEEN TID AND TIMING WINDOWS 52
OPERATING POINT TRADEOFFS 53
EMPIRICAL MODEL OF MAXIMUM OPERATING FREQUENCY 54
PROPAGATION OF DIGITAL "1" 56
CONCLUSIONS 57

V. DISCRETE BAYESIAN ANALYSIS FOR DIAGNOSIS AND PROGNOSIS 58

INTRODUCTION 58
BAYESIAN NETWORK CONSTRUCTION FOR TID AWARENESS 60
BUILDING A MODEL OF THE SYSTEM 61
Schematic/Block Diagram 61
Fault Propagation 62
Functional Dependencies 63
Designing the Bayesian Network 64
Populating Bayesian Network with Probabilities 68
ROBOTIC LINE FOLLOWER 70
Estimating Probability of Success of Key System Functions 70
Inference Using Observable Parameters 71
Sensitivity Analysis for Design Enhancement 73
Estimating TID Levels Including Different TID Levels for Different Components 75
DISCUSSION 76
Comparison with Hardness Assurance Methods 77
Working with Hybrid Data Sets 78
Design or Model Modification 79
Sensitivity to Discretization Process 79
CONCLUSIONS 80

VI. CONTINUOUS BAYESIAN MODELING OF TID DEGRADATION 82

INTRODUCTION 82
BACKGROUND 83
USING CONTINUOUS PARAMETER BAYESIAN NETWORKS FOR TID DEGRADATION MODELING 87
CASE STUDY: LINE TRACKING ROBOT 88
System Details 88
EXPERIMENTAL DETAILS 91
BAYESIAN NETWORK CONSTRUCTION 93
SABER MODEL OF ROBOT SYSTEM 95
CASE STUDY RESULTS 97
Verification of Parameter Estimation (Line sensor parameter estimation) 97
Prediction of System Response 100
DISCUSSION 104
Model Flexibility 104
Bayesian Network's Role in Hardness Assurance 104
Managing Model Complexity 105
Software Injection of TID Degradation 106
CONCLUSIONS 106

VII. SUMMARY AND CONCLUSIONS 107

APPENDIX

A. FUTURE DIRECTIONS 110

B. pyMC MODEL 113

C. MICROCONTROLLER TEST CODE 114

D. PYTHON ROBOT MODEL 119

E. ROBOT EXPERIMENTAL DETAILS 135

F. PyMC TUTORIAL 146

REFERENCES 164

LIST OF TABLES

Table Page

1. Possible Degradation Effects for Sonar Subcomponents.....32

LIST OF FIGURES

Figure Page

1. Functional diagram of abstracted robotic system. 2

2. Band diagram illustration from [10], showing how ionizing radiation leads to fixed charge trapped in SiO2 and the accumulation of fixed charge at the Si/SiO2 interface. 9

3. Four node Bayesian network from [21] Node Z is dependent on nodes X and Y, and node Y is dependent on node W..... 12

4. Notional example of degradation in a theoretical rangefinder. The degraded post-rad transfer function produces an incorrect calculated value when inferred from the system processor pre-rad look-up table. 18

5. Photograph of the three types of rangefinders tested on the mount used for measuring the transfer functions. From top left clockwise: IR rangefinder, sonar, laser rangefinder. 20

6. Test apparatus used to measure input-output transfer function for each sensor. The mounting points were designed for highly repeatable measurements (measurements repeatable to within 0.5%). 21

7. Measured pre-irradiation transfer function for the three rangefinders. Each sensor maps the physical phenomenon (distance), to a circuit level output (serial bit stream for laser rangefinder, digital pulse length for the sonar, and analog voltage for the IR sensor). 23

8. Transfer function for laser rangefinder for increasing TID. The plot shows the relationship between the serial output stream and distance for each TID value tested. Failure to respond is indicated as a sensor output value of 0 on the plot. Sensor shows no degradation until abrupt failure at 8 krad(SiO_2). 24

9. Transfer function for sonar rangefinder for increasing TID. The plot shows the relationship between the time for the echo to be detected and distance for each TID value tested. Failure to respond or the echo exceeding a maximum time is indicated as a sensor output value of 0 on the plot. 27

10. Transfer function for IR sensor for increasing TID. The plot shows the relationship between the sensor output voltage and distance for each dose level tested. Significant part-to-part variation was observed, and this plot serves as a representative case. 28

11. Ratio of post-rad calculated distance to actual physical distance for an IR sensor for increasing TID. The error from radiation is not distributed uniformly over the range of the sensor. 28

12. Significant part to part variability was observed for different instances of the same part number IR sensor. A slice in the dataset after 23 krad(SiO2) exposure for IR sensor. Sensor 1 corresponds to the sensor shown in Figure 10 and Figure 11. 29

13. Transfer function including 90 hour anneal. The sensor anneals to a different transfer function than the pre-irradiation transfer function, complicating efforts to recalibrate the device by tracking the received TID. 30

14. Supply current versus dose for the sonar and laser rangefinder. The degradation in supply current tracks strongly with the degradation in the input-output relationship of these sensors. The shifts are in different directions due to the supporting electronics. 30

15. Functional diagram of the sonar sensor. Test points were placed at each line of the figure to determine the source of the degradation. 33

16. Simulation of how the sonar input-output relationship would map a room after different irradiation steps. 35

17. Timing window illustration for microcontroller architecture where the entire instruction is executed in a single clock cycle. Nominal demonstrates correct operations, while the TID induced failure portion demonstrates incorrect data being latched due to increased propagation delays. 43

18. Illustration of timing window violation hypothesis, demonstrating the relationship between TID, operating frequency, and supply voltage (Vdd). 45

19. Block diagram from ATMEGA328P datasheet showing the microcontroller architecture. Arrows and text added to highlight the different regions of the microcontroller that the software functional unit tests exercise [50]. 47

20. Diagram showing the control flow for the master-slave microcontroller test configuration. Icons indicate test code written in C (top left), programmer board (top middle), test device board (Arduino Uno, top right), python script (bottom left), Keithley Sourcemeter and function generator (middle), and interrogator master processor (bottom right). 48

21. Results of test procedure for a representative part for the ATMEGA328P microcontroller. The maximum operating frequency for which the tests pass decreases with increasing TID. Error bars correspond to 1 MHz frequency measurement step size. 50

22. Results of test procedure for a representative part for ATMEGA328P microcontroller. Radiation steps were performed with all pins grounded, and then the test procedure to find the maximum operating frequency was performed for the three voltages. The data shown are for the opcode test. Error bars correspond to 1 MHz frequency measurement step size. 51

23. Results of the test procedure for a representative part for the PIC16F677 microcontroller with the part biased at 3.3 V during irradiation. Similar to the previous plots the maximum operating frequency for which the tests pass decrease with increasing TID. The leakage current is included on the right side axis. Error bars correspond to 1 MHz frequency measurement step size. 52

24. Data set for ATMEGA328P biased at 3.3 V and irradiated with all pins grounded for the ALU opcode test for “a” = 28.25, “b” =0.004225, “c” = 0.08827, and R2 = 0.9690. 55

25. Block diagram of case study system. Blocks represent key physical sections of the system and lines indicate signal or energy paths. Arrows indicate dependencies for functionality, but in Bayesian networks all arrows are bidirectional so direction is unimportant. 62

26. Failure modes for the line sensor. It can receive degraded power from the linear regulator and pass a degraded output to the microcontroller. 62

27. Model of functional dependencies for line follower robot. 64

28. Simplified Bayesian network for line-follower robot. Image shows the nodes, their interconnections, and the possible discrete states, separated into highlighted layers. 67

29. Conditional probability table showing the values used to populate the line tracking node in Figure 28. 70

30. Example of forward inference. The evidence of the TID node is set to the "Under 50 krad(SiO2)" state. The posterior distribution of the remaining nodes is calculated using Bayesian inference update. 72

31. Example of performing inference using observable parameters. Setting the White Surface Sense and Supply Current observational nodes to Degraded allows for the inference of the status of Line Tracking and Total Ionizing Dose distributions. 73

32. Automated sensitivity analysis for the Line Tracking node performed by the Bayesian network tool. Colors are coded by tracking node sensitivity. The Line Tracking performance is most influenced by Total Ionizing Dose, which is the logical outcome for this system topology. Notably, the observational nodes have the least direct impact on the Line Tracking distribution. 74

33. Modified network including a separate TID node for each of the components, accounting for part replacement, different shielding levels, or other factors that could produce different dose levels for different components. 76

34. Overview of continuous Bayesian network modeling process. The analysis is a multi-step process, and incorporates domain specific information in math models, statistical information in the Bayesian network, and system model information in the deterministic simulator. Additionally, failure modeling language can be used to inform the network architecture. 86

35. Image of line sensor over a track [66]. In operation the line sensor faces the track. 89

36. Figure shows individual sensor outputs (bottom) and combined output (top) for sensor board position relative to the center of the line. 90

37. Theory of degradation for line sensor optical couplers. The two primary sources of degradation are a decrease in the gain of the phototransistor and the decrease in output of the LED. For TID induced degradation, a decrease in gain of the phototransitor. 91

38. Example degradation of a reflectance sensor. The peak value of the sensor remains relatively constant with dose, but the low range output (over highly reflective surfaces) increases due to a decreased max gain in the phototransistor. 92

39. Degradation of combined sensor response for different TID levels. The curve is used as an input to a proportional-integral-derivative controller. As the curve becomes “flatter”, the effective control gain decreases. 93

40. Bayesian network for line-follower robot with degraded sensor. The model is broken up into three sections: Bayesian component model, environment model, and deterministic system model. 95

41. Completed system-level SABER model of Pololu robot. 96

42. Simulated position components for nominal parameters (left) and degraded parameters (right). 96

43. Summary of Bayesian sampling for the slope of one of the sensors. 98

44. Parameter estimation for the slope and intercept of the PyMC model. The slope and intercept values correspond to the log-linear white-level model parameters. The shape of the surface describes the part-to-part variability in radiation response. 99

45. Comparison between Bayesian generated samples and experimental data. Larger dots with error bars represent the data set mean and standard deviation. Smaller dots correspond to simulation results. 100

46. Comparison of Bayesian network and Saber result to actual full system test. Circles represent simulation results for maximum distance the robot deviates from the line (left axis) and diamonds represent experimental results for the ratio of robots that fail to track the line (right axis). 101

47. Cumulative probability distribution function of the robot’s ability to track the line versus dose for nominal operating conditions and an increased controller gain. Points represent experimental data from 5 tested systems. Lines represent model predictions. 103

48. Comparison of modeling the data together, or as two groups using one of the groups as an informed prior. The distribution shifts depending on how the priors are chosen. 112

49. Simulations of python robot model progressing around the track. 119

50. Calibration curve generated by sweeping the software model over the line, using the updated parabola model. 120

51. Degraded calibration curve at 50 krad. Note the decreased amplitudes of the parabolas and the increase from 0 in the offset. 121

52. Simulation of robot line-tracking performance versus the max achievable sensor output due to degradation in the linear regulator. 122

53. Simulation of the effect of the white level offset in the sensor on tracking performance. ... 123

54. Results of combined linear regulator model and sensor model. 124

55. Plot of probability of failed tracking of just the sensor degrading, just the regulator degrading, and both degrading. 125

56. Image of model robot and line sensor characterization setup. 135

57. Camera mount with track on the floor. Camera is placed in the middle of the mount and uses the four dots on the floor for image processing. 136

58. Image after video processing, showing the track the robot followed in orange. This is an example of successful tracking. 137

59. Image after video processing. This is an example of failed tracking, with the robot spiriling in circles near the bottom of the image. 137

60. Visualization of generated data. 149

61. Basic Bayesian network for normally distributed transistor Beta. 151

62. Comparison of step function prior versus std. dev. prior. 152

63. Distributions and chain plots for modeled parameters in the basic_model. 156

64. Lot-to-lot variability plot. 160

65. Lot-to-Lot variability model. 161

CHAPTER I

INTRODUCTION: PROVIDING AWARENESS

"Will it work?" This question, while seemingly very basic, is an abstraction of many concerns. Will the assumptions made during radiation testing prove to be consistent with the actual application? Will the environment degrade the components to the point where the system fails? Will any anomalies occur that stress the system to the point of failure? Knowing how the system will behave in the environment of interest allows for the appropriate mitigations to be implemented. All systems have limitations and circumstances under which the system will fail, and understanding these limitations and circumstances enables the use of the system in an intelligent manner. This work presents a framework for providing quantified, evidence-based answers to the likelihood of success for system functionalities using Bayesian analysis to incorporate multiple information sources. This work focuses on a specific type of system in a specific environment: a robotic system in a radiation disaster scenario, exploring the question of whether or not the robot will be able to complete the desired tasks in the application environment. However, many of the conclusions reached in this work are applicable to a variety of scenarios tracing subsystem changes to the system level.

Providing quantified reliability information spanning multiple levels of an electromechanical system hierarchy is a challenging problem because it stresses the abstractions made at each level of the hierarchy, such as the hierarchy shown in Figure 1. The design and modeling of complex systems heavily utilizes abstraction – breaking one large problem into smaller and smaller problems such that the necessary information for a particular level of the design is maintained and

the additional complexity of other levels is hidden. This approach has enabled the phenomenal progress of electronic design, with countless components available commercially, each solving a specific problem. In order for abstraction to work, assumptions must be made about the behavior of abstracted components. These assumptions can take many forms, such as datasheet requirements (e.g. the input voltage must be between 4.8 V and 5.2 V for a 5 V microcontroller). The stricter the requirements are around the individual subsystem components, the more difficult the design becomes.

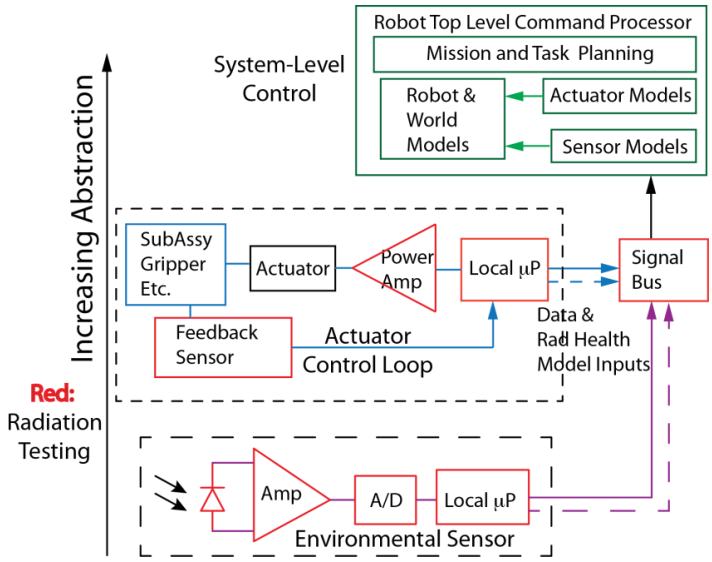

Figure 1. Functional diagram of abstracted robotic system.

The impact of radiation on electronics physically occurs at the device level, impacting each transistor, diode, or oxide independently. In the radiation effects literature, the focus is primarily on the impact on individual devices or components, because this type of insight is most widely applicable. However, the combined impact of this radiation can manifest itself at the component, sub-system, or system level, yielding complex responses that are potentially difficult to trace back to their source. The number of transistors in a modern robotic system can easily climb into the

billions, making the simulation of the entire system at the device level unrealistic even with modern computing resources. When modeling the full complexities of a system is not feasible, what is the next best that can be done? Deterministic models can be used where the physics is known and modeled in detail, and probabilistic modeling must be used when modelling processes that have significant variation and the models are driven by experimental data. This scenario defines the question addressed by this work: is it possible to include probabilistic degradation information in a model of the system that allows for meaningful action to be taken?

Rather than focusing on building highly detailed models of the radiation degradation of specific robots, or performing exhaustive component tests, this work focuses on using a mix of experiments and modeling to develop a framework for answer the following questions: will the system work, with what confidence, and what improvements would be most beneficial. These questions can be answered through tracing subsystem effects to the system level, measuring quantities of interest called system health indicators, and Bayesian analysis techniques to merge all available information.

The work is broken into four chapters, with the first chapter presenting a case study of characterizing the degradation of specific components, tracing those effects to the system level, and identifying key parameters that provide information about the health of the components. The second chapter presents a novel technique for measuring and forecasting the health of a complex integrated circuit, focusing on microcontrollers. The third chapter presents a framework based on Bayesian statistics for designing experiments, quantifying the probability of successful operation for key functions, and identifying areas where the most design improvement can be made using discrete categories. The fourth chapter presents implementation of the Bayesian analysis techniques using continuous distributions for improved modeling attributes. Each chapter

corresponds to a published journal article, and contains expanded detail not included in the journal publication.

CHAPTER II

BACKGROUND: RADIATION EFFECTS FROM TRANSISTOR TO SYSTEM

This work weaves together topics ranging from device physics to robotics. In the following background sections, the goal is to draw out key information that is unique to this work. General review of relevant topics in extensive detail is cited where appropriate.

Radiation and Robotics

Robots are desirable in situations where the environment or task is too dangerous or difficult for human workers. An overview of the major deployments of robots for disaster scenarios, including radiation disasters such as Fukushima-Daiichi is found in [1]. The book reaches several general conclusions about disaster robotic systems. First, the time between the disaster and the arrival of disaster robots is critical for effective mitigation, with the first 24-48 hours playing an especially important role in limiting the effects of a disaster. This suggests that commercially available robotic systems designed for a wide variety of circumstances will play an important role, rather than custom designed robots for a specific disaster (although custom designs can come into play during the longer cleanup phase). Second, user error is the primary source of malfunction/failure, with the mean-time-between-failure on the range of 20-100 hours. The book suggests that additional electronics and automated control termed assistive autonomy is necessary to reduce the likelihood of user error. Robots will continue to expand their use of electronics.

A comprehensive analysis of the impact of radiation on robots was completed in 1997 by Sandia National Laboratories [2]. The Sandia efforts provide an important foundation for work at the

intersection of robotics and radiation. The report provides an analysis of the different radiation environments robots may encounter, the impact of different types of radiation, the test results for key robotic components, and includes case studies for multiple robotic systems.

The report identifies total ionizing dose (TID) from gamma radiation as the main source of radiation degradation. Neutron, alpha, or other types of radiation are less prevalent with the exception of interaction directly with nuclear fuel. The general focus of the analysis is on identifying a level to which the component operates successfully (e.g., the sensor was within specification until 100 krad(SiO_2)).

The general conclusions that the report identified are that the electronics are the most sensitive components, degrading significantly before most other materials (Teflon and similar polymers are also very sensitive). This conclusion guides the focus of this proposals focus on the electronic components as the critical components to be modelled in the robotic system.

Significant emphasis is placed on testing sensors in the Sandia report, because while other component can be shielded from radiation, the sensors must be exposed to the environment to perform their function.

A variety of post-radiation-disaster environments are possible, but they all share several characteristics: the radiation field can be variable over space and time, the physical environment can be highly unstructured and unpredictable (rubble, debris, etc.), other extreme environment elements may exist (dust, temperature, smoke), and communication bandwidth may be limited. These factors pose significant challenges to accurate sensing and interpretation of the robot environment - a key capability in order for robots to be able to perform useful work at a post-nuclear-disaster site.

Nuclear Power Robotics

Other work involving robots and radiation focuses on the maintenance of nuclear power plants and the handling of nuclear waste, with a detailed analysis of the approach to radiation and robotics available in [3]. These specific applications are highly structured and can support custom hardware and regular maintenance using systems designed with the radiation environment from the beginning. Significant work exists in developing these types of robotic systems, with an emphasis on shielding the sensitive components from the radiation sources. The "split" technique, separating out all the components that can be separated behind shielding, is commonly used in the nuclear power industry. Other techniques include replacing parts on a scheduled basis. Custom designs using radiation hardened components require significant design effort and can be prohibitively expensive.

Fukushima –Daiichi

The Fukushima-Daiichi natural disaster and subsequent nuclear power incident in March 2011 created a dangerous work environment for human operators, including possible high radiation exposure levels and the requirement to wear protective suits resulting in heat and mobility issues [4]. Robotic intervention was used where possible, including surveying by terrestrial and aerial based robots or drones [4]. The ground-based robots encountered a radiation environment dominated by gamma radiation, potentially producing significant TID [5]. In the early stages of disaster survey and remediation, the robots used for intervention primarily consisted of COTS components, which went through some TID screening but not a stringent hardness assurance process [5], a development process similar to low-cost small satellites. Currently, custom-designed robots for remediation tasks are at work at Fukushima in what is one of the largest engineering

tasks in the world, scrubbing and decommissioning the contaminated areas from the disaster [6][7]. In high dose areas, the TID response is critical to the number and quality of missions the robot can perform before being decommissioned or reconditioned. Awareness of the TID response of the system can guide the choice of missions performed based on available functionalities or identify high value improvements to the system. Recent efforts at exploring the inside of the reactor rooms has led to unspecified failure by a custom-designed robotic system [8]. It is inconclusive if radiation-induced degradation is the source of failure for this instance, but radiation degradation is a potential candidate for causing the failure.

The radiation environment in a reactor accident or post-explosion scenario can consist of a mixture of alpha, beta, gamma, and neutron emitters [8-10]. The contribution of these particles to the TID deposited in the oxides of the sensor electronics depends on the distribution of radioactive material in the environment, the penetration of the particles and the capture cross-sections of the materials. Gamma rays are of particular concern because of their occurrence in relevant nuclear decay chains, high penetration range, high dose rate, and persistence over long time scales [8-10]. It is difficult to track the effects of the accumulated dose on the electronics in real time in the field, due to the unstructured environment's highly variable dose rate and the temperature-dependent recovery of damage due to annealing processes.

Total Ionizing Dose and Electronics

Gamma-induced TID affects CMOS and bipolar electronics, as well as a variety of other materials used in this environment [2]. Fundamentally, photon or particle radiation deposits energy that creates electron-hole pairs in the electronic materials. The most significant effects occur in the insulators accompanying the semiconductors, such as gate or field oxides in MOS devices.

Electrons exit the insulators quickly, while holes do not, because of their low mobility relative to the electrons, leading to the build-up of positive charge with time and accumulated dose [9], as illustrated in Figure 2.

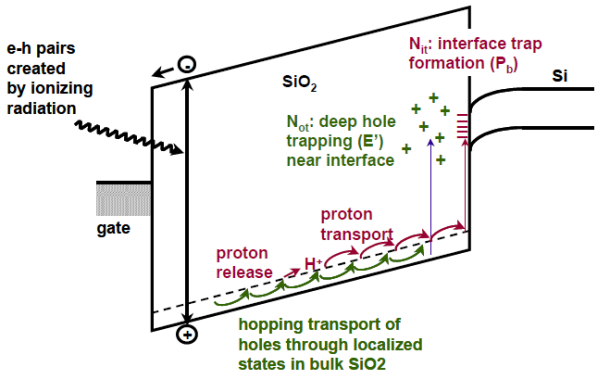

Figure 2. Band diagram illustration from [10], showing how ionizing radiation leads to fixed charge trapped in SiO_2 and the accumulation of fixed charge at the Si/SiO_2 interface.

This oxide-trapped charge results in changes in the electrical characteristics of the semiconductor devices, for example, changes in threshold voltage and increased leakage current in CMOS transistors. Fabrication-level hardening techniques for integrated circuits exist, but most robotic sensors are designed using commercial off the shelf (COTS) electronics for cost, availability, and schedule reasons, which are guaranteed to meet only a range of electrical parameters prior to any exposure to radiation. In robot applications, the term "sensor" often refers to an entire subsystem, including signal processing electronics, voltage regulation, and communication ports, rather than just the actual transducer itself. The effects of radiation on a COTS sensor assembly in a gamma TID environment are a combination of the radiation effects on the physical sensing element and the radiation effects on the supporting electronics performing the signal processing. Dose rates for nuclear disaster environments are inherently highly variable, due to the different types and

severities of accidents possible and the inverse-square law relationship between dose rate and distance from the radiation source. Field reports from facilities surrounding the Fukushima-Daiichi nuclear plant showed dose rates as high as 6.5 rad/hour, with much higher dose rates anticipated as the distance from the reactor decreases [11].

Testing electronics to evaluate the radiation response is a complicated practice, with several industry and military standards in place [11][12][13]. In general, it involves simulating the radiation environment using a source that can produce similar radiation damage in a much shorter time period. This can be accomplished by using high flux sources, but for certain applications dose rate effects must be considered. For all experiments performed in this work, to simulate a nuclear disaster radiation environment, a 660 keV Gamma Cs-137 source is used.

Hardness Assurance

Traditional hardness assurance is centered around assurance, guaranteeing that the electronic components and subassemblies will remain within specification for the desired mission. This is a necessary and appropriate goal for many applications. The primary challenge to this approach is the difficulty in incorporating existing commercial solutions and subassemblies whose radiation tolerance is unknown. The commercial electronics market is extremely large and continuously investing significantly in research and development. A detailed review of hardness assurance approaches can be found in [14].

System Reliability and Bayesian Statistics

System reliability is a broad field, encompassing many areas of specialization and approaches. A general overview of the approach to system design and system reliability can be found in the

NASA Systems Engineering Handbook [15]. The document covers a variety of topics, with some of the key points included understanding which sub-systems are most vulnerable in the system and taking steps to minimize and tolerate variability in sub-systems.

Understanding the rate of occurrence and the propagation of discrete faults throughout the system is one area of focus in systems reliability. An in-depth analysis of the approach to modeling discrete faults in a qualitative manner can be found in [16]. This approach is fairly mature, but does not apply directly in TID environments because there are potentially large amounts of degradation rather than binary success/failure.

Bayesian networks provide a method for modeling continuous quantities, such as a system parameter that degrades with TID, by using probability distribution which can be binary, discrete, or continuous. A Bayesian network is a graphical representation of the relationships between different probabilistic quantities. Bayesian networks draw their name from the application of Bayes' theorem (stated in Eq. (1)) which gives a method for calculating the posterior probability of occurrence of an event A given that event B has occurred (P(A|B)), from the likelihood of the occurrence of event B given that event A has occurred (P(B|A)), and the prior probability of the events A (P(A)) and B (P(B)).

$$P(A|B) = {P(B|A)P(A)}\Big/{P(B)} \tag{1}$$

Bayesian statistics have been applied to total-ionizing dose and single-event effects previously in [17][18][19][20], with a focus on the statistics of individual components. The papers cited contain

a detailed discussion of the origin of Bayesian statistics and the benefits of applying Bayesian statistics in the radiation effects context. This work builds on the Bayesian statistics approach, applying the techniques to hierarchical systems consisting of multiple interacting components.

A simple four node example Bayesian network is shown in Figure 3,adapted from [21], where each node represents some random variable. The graphical representation is a directed acyclical graph (DAG). While the arrows have a direction, there can be no cycles (or feedback loops). The arrows represent a correlation or causal link, and the nodes have a distribution of conditional probabilities for being in the possible nodal states, with the probabilities for all the states summing to 1.

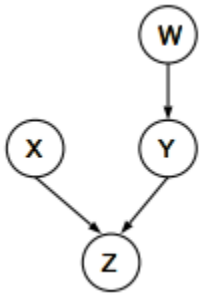

Figure 3. Four node Bayesian network from [21] Node Z is dependent on nodes X and Y, and node Y is dependent on node W.

Bayesian networks have the advantage that each node is only dependent on the nodes directly linked to it. In other words, each node is conditionally independent from its non-descendants, given its parents [21]. Without the network structure, the joint probability can be calculated as the product of conditional probabilities using the same notation as example (1):

$$P(W,X,Y,Z) = P(W)P(X|W)P(Y|W,X)P(Z|W,X,Y) \quad (2)$$

With the network structure the calculations can be simplified, e.g.:

$$P(W,Z|X,Y) = P(W|Y)P(Z|X,Y) \tag{3}$$

Similar equations can be written for the other nodes. By chaining the node equations together in a network and solving for the target value, the desired variable can be estimated, a process called Bayesian inference [21][22]. Bayesian statistical inference is a broad and expanding field, with appealing qualities for many disciplines. Fundamentally, it performs the same type of statistical inference as traditional techniques such as those found in [23], but Bayesian statistics also have many appealing properties for simulation and interpreting results. Notable properties include that the simulation results demonstrate ergodicity, meaning subsections of results are statistically representative of the results as a whole. Additionally, the Markov chain Monte Carlo techniques produce results for parameters that can be incorporated into other simulation tools. Detailed discussions of these properties can be found in [24][25][26].

Software packages exist for performing Bayesian inference on an arbitrarily large network [27]. Discrete Bayesian network computation packages such as GENIE [28] provide an easy to use and interpret environment and efficient computation. Continuous modeling packages such as PyMC [29] or BUGS (OpenBUGS [30], WinBUGS [31]) provide great flexibility but require additional effort in terms of computation and evaluating the goodness o

f fit for the model and computational approach. Additional approaches such as dynamic Bayesian networks have been implemented for systems that undergo discrete state changes [32].

An extensive review of the historical development and popular algorithms for computing with continuous Bayesian networks is available in [33]. The general approach is to utilize properties of Markov chain theory and Monte Carlo techniques to calculate the posterior probabilities given the

evidence and the prior. Examples of Bayesian analysis for reliability applications are available in an excellent NASA primer for the use of Bayesian networks for reliability found in [33].

The notation up to this point is used to describe discrete events. For continuous random variables, Bayes' theorem takes a similar form, using continuous functions that describe probability distributions:

$$f(a|b) = \frac{f(a)f(b|a)}{\int f(b|a)f(a)da} \tag{7}$$

Normal distributions are commonly used to describe part-to-part variability. The equation for the probability density function of a normal distribution is:

$$N(\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{8}$$

Combining multiple normal distributions using Bayes theorem leads to analytically complex integrals such as Eq. 9 for only 2 continuous variables.

$$f(a|b) = \frac{\frac{1}{\sigma_a\sqrt{2\pi}} e^{-\frac{(x-\mu_a)^2}{2{\sigma_a}^2}} * \frac{1}{\sigma_{b|a}\sqrt{2\pi}} e^{-\frac{\left(x-\mu_{b|a}\right)^2}{2{\sigma_{b|a}}^2}}}{\int \left(\frac{1}{\sigma_{b|a}\sqrt{2\pi}} e^{-\frac{\left(x-\mu_{b|a}\right)^2}{2{\sigma_{b|a}}^2}} * \frac{1}{\sigma_a\sqrt{2\pi}} e^{-\frac{(x-\mu_a)^2}{2{\sigma_a}^2}} \right) da} \tag{9}$$

Past work explores the use of the above equations in more detail for component hardness assurance [19]. This work will focus on the use of continuous variables and Bayes' theorem for networks to predict system performance. Historically, analytically difficult integrals such as Eq. 9 have slowed

the adoption of Bayesian statistics for larger networks, where the number of variables can be significantly larger than two. Inference algorithms have been developed to solve these types of complex integrals efficiently, meaning the user only needs to design the network and interpret the results. Monte Carlo methods are used to sample the desired distributions, specifically Markov Chain Monte Carlo (MCMC). The methods are available in multiple software packages, such as OpenBUGS [15] or PyMC (Python Markov Chain Monte Carlo) [29]. For this work, PyMC, a Python based implementation of the MCMC approach, is used. PyMC allows for straightforward integration with other simulation tools. The integration and sampling techniques use several properties of the Bayesian network to perform efficient computation on a multidimensional parameter space.

CHAPTER III

TRACING RADIATION DEGRADATION

Z. J. Diggins, N. Mahadevan, D. Herbison, G. Karsai, E. Barth, R. A. Reed, R. D. Schrimpf, R. A. Weller, M. L. Alles, and A. Witulski, "Range-Finding Sensor Degradation in Gamma Radiation Environments," *IEEE Sensors Journal*, vol. 15, no. 3, pp. 1864–1871, Mar. 2015.

Introduction

In a post-nuclear-disaster environment such as that of the 2011 Fukushima-Daiichi accident, robots are ideal to survey the environment and perform remediation tasks in areas where radiation levels are either unknown or unsafe for human occupancy. While full custom designs are necessary for operation near the core or handling nuclear fuel [34], commercial disaster robots are useful for rapid deployment in the surrounding facility [35]. The robots may need to be equipped with a variety of range finding sensors to combat different sensing challenges (dust, smoke, water vapor, etc.). Radioactive materials in the environment have the potential to affect the performance of the robot sub-systems. Range-finders and other sensors are especially vulnerable because at least a portion of the sensor must be exposed unshielded to the environment. Extensive research on the effects of total ionizing dose (TID) on semiconductor devices exist [9], [36], [37], showing that the TID that the sensor receives from the radiation environment has the potential to change the sensor transfer function, which is defined in this work as the mapping between the physical

distance being measured and the sensor output signal. Range-finder sensor failure due to total ionizing dose has been reported previously [38], [39], but the impact of TID on the transfer function while the range-finder sensor is still operational has not been investigated. In this work, three sensors that exploit different physical laws for range finding were irradiated, and their transfer functions were measured: the Parallax Laser Range Finder, the PING Ultrasonic Distance Sensor, and the Sharp GP2Y0A21YK0F Infrared Rangefinder. The sensors were irradiated using a 660 keV ^{137}Cesium gamma source, and the sensor transfer functions were then evaluated across their specified distance ranges. The key contribution of this work is to determine not only a sensor failure point due to radiation, but to measure and analyze the degraded sensor signal before failure is reached.

This work evaluates the degradation of rangefinders due to radiation typical of a nuclear disaster environment. A measurement technique for measuring the input-output transfer functions of each sensor is presented, and the results of the degradation for each sensor are shown. Furthermore, a diagnosis methodology is presented, allowing identification of which individual components in the sensor assembly produce the measured degradation. Additionally, implications of the degradation are evaluated, exploring how different types of degradation will affect system level performance of a robot performing mapping tasks in a nuclear disaster environment.

Sensor Input-Output Relationship Degradation

In a typical robotic application, a processor has a look-up table or similar algorithm to map a sensor output to a calculated physical value that will be used by the system for decision making. If the sensing element or supporting electronics are impacted significantly by radiation, the relationship between the physical distance being measured and the sensor output will change. This scenario is

the motivation for not just finding the total failure point but also measuring the degradation before failure. The goal of this work is to test this hypothesis and explore its implications. Figure 4 illustrates the hypothesis for a theoretical range finding sensor, which relates the physical distance to the sensor output through a constant of proportionality divided by the distance. The left half of the figure shows the sensor's mapping between the physical phenomenon (distance) and the sensor output (voltage). The black curve is the pre-rad value, which would be stored in a system processor and used to generate a calculated physical value, as shown by the right half of the figure. The orange curve is a hypothetical mapping after the sensor has received a given amount of TID (the sensor is still operating but in a degraded state), which would result in a different calculated physical value.

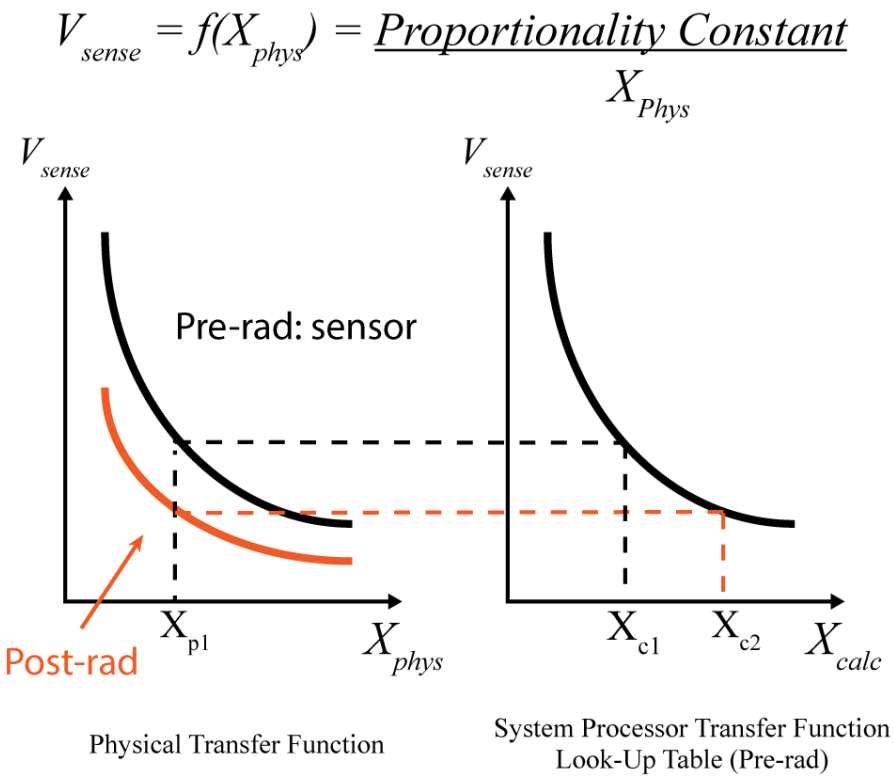

Figure 4. Notional example of degradation in a theoretical rangefinder. The degraded post-rad transfer function produces an incorrect calculated value when inferred from the system processor pre-rad look-up table.

Before irradiation the calculated distance (X_{c1}) will be close in value to the actual physical value (X_{p1}), but after irradiation the calculated value (X_{c2}) will potentially differ significantly from the actual physical value (X_{p1}). Different distortions in the mapping are possible due to the interaction between the sensor sensing element and signal processing electronics. The following sections present a method to measure the shape and severity of the degradation of environmental sensors before complete failure in post-radiation-disaster environments and explore the implications of the degradation.

Experimental Details

Three rangefinders using different physical principles for sensing were investigated. Although all the rangefinders measure distance, the physical output variable is different in each case. The Parallax Laser Range Finder uses a visible laser and a CMOS camera array to make a triangulation measurement - the output is the camera array column that is the center of the reflected illumination. The PING Ultrasonic Distance Sensor uses sonar time of flight - the output is a digital pulse whose length is the time it takes for the sound to reach the target and return. The Sharp GP2Y0A21YK0F Infrared Rangefinder uses a position-sensitive detector (PSD) and an infrared (IR) light emitting diode (LED). The distance between the sensor and the target changes the position of the reflected IR spot on the PSD. The PSD signal is processed through analog circuitry – the output is an analog voltage inversely proportional to the distance to the target.

Figure 5. Photograph of the three types of rangefinders tested on the mount used for measuring the transfer functions. From top left clockwise: IR rangefinder, sonar, laser rangefinder.

Figure 5 is an image of the sensors used. These three devices were chosen because of their different levels of complexity, different modes of output, and different physical sensing mechanisms. The IR sensor consists of a single integrated circuit and the infrared emitter/receiver pair, functioning as an entirely analog device. The sonar is more complex, using a microcontroller to supervise the start and stop of a sensing pulse, consisting of a mixture of analog and digital components. The laser rangefinder is entirely digital, making discrete measurements and communicating through a serial bit stream. The three sensors represent a large class of sensors that could be used for robotic range-finding, and the architectures are similar to those used in more complex rangefinders such as scanning laser rangefinders.

For this work, each sensor was tested as it would be deployed on a disaster-mitigation robot. The entire sensor assembly including the supporting electronics was irradiated with no shielding. For robotic applications, weight and ease of integration are important, making commercial unshielded

sensors the logical choice. If the sensor designer is making a custom radiation hardened sensor, potentially the radiation response of only the transducers is of primary interest such as in [40]. In this case, the transducer can be irradiated to dose steps by itself and then inserted into a measurement system for characterization.

A test mount for measuring distance with all three sensors at the same time was designed to make variations in sensor and target positioning minimal (less than 0.3% of angle error and less than 1 mm of target positioning error were achieved). A ^{137}Cs source, was used for all irradiations; the 660 keV gamma rays are representative of the environment and will penetrate the sensors without significant attenuation [38].

All parts were exposed to specified TID intervals and then removed from the irradiation chamber and placed on the mount to measure the transfer functions as illustrated by Figure 6. The sensors were powered on during irradiation and placed in a standby mode. The time between irradiation and measurement was minimized to reduce any impact of room-temperature annealing on the measurement.

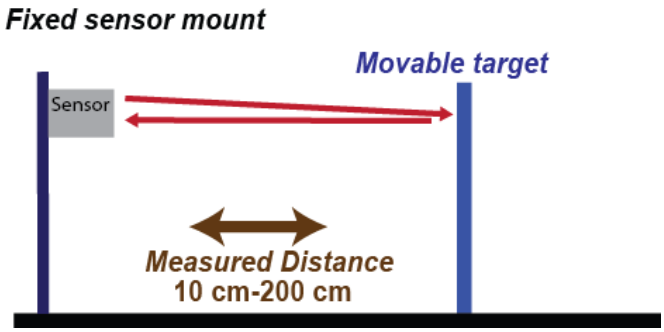

Figure 6. Test apparatus used to measure input-output transfer function for each sensor. The mounting points were designed for highly repeatable measurements (measurements repeatable to within 0.5%).

The sensors' measured pre-irradiation transfer functions are shown in Figure 7. These pre-rad values are used in a lookup table as an inverse function to calculate measured distance versus actual distance for some of the following plots, as they would be used in a typical robotic control system.

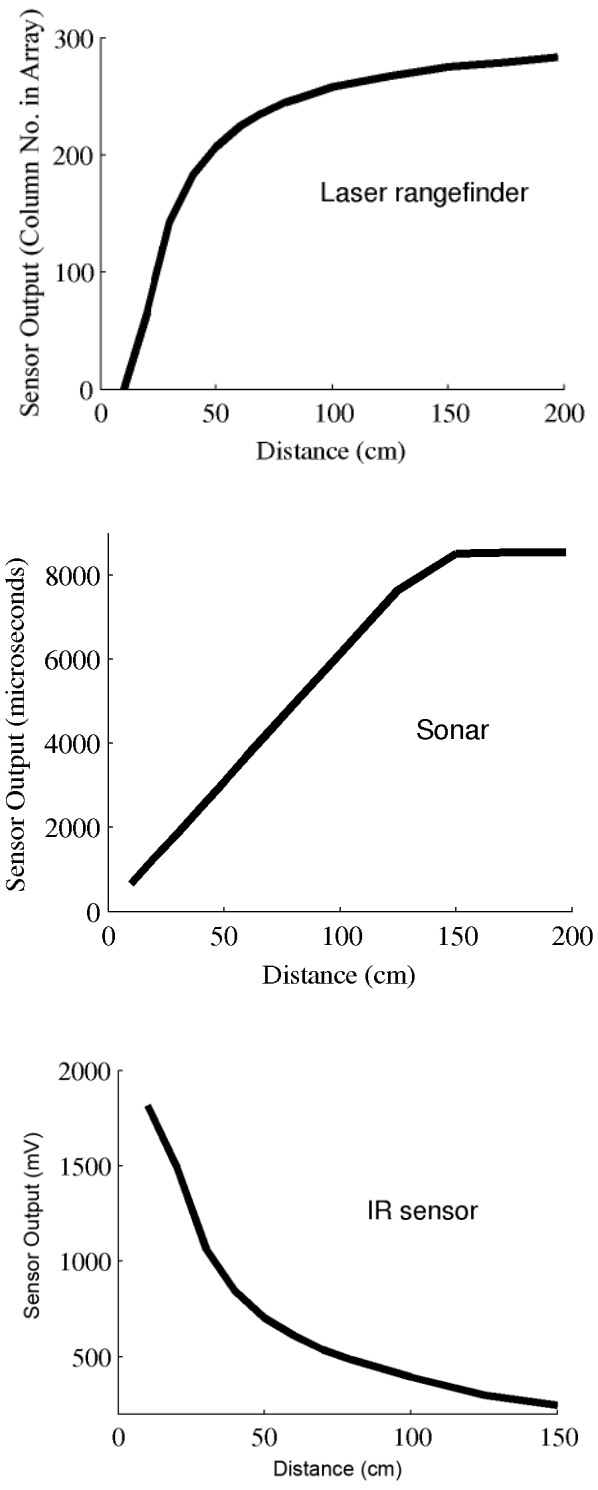

Figure 7. Measured pre-irradiation transfer function for the three rangefinders. Each sensor maps the physical phenomenon (distance), to a circuit level output (serial bit stream for laser rangefinder, digital pulse length for the sonar, and analog voltage for the IR sensor).

Results

Figure 8 shows the transfer function of the laser rangefinder versus dose, with the pre-irradiation curve being placed at TID = 0 krad(SiO_2) (far right side of the 3-D plot). Three instances of the sensor were tested, all showing the same results. The laser rangefinder showed no degradation until an abrupt failure at 8 krad(SiO_2). Once the sensor failed, it did not respond to serial communication requests, which is displayed as a sensor output of zero on the plot. Follow-up tests performed on the sensor components indicated that the failure was due to the microcontroller that processed the camera image. This type of failure mode occurs at a low TID level with no indication of upcoming failure in the pre-failure transfer functions.

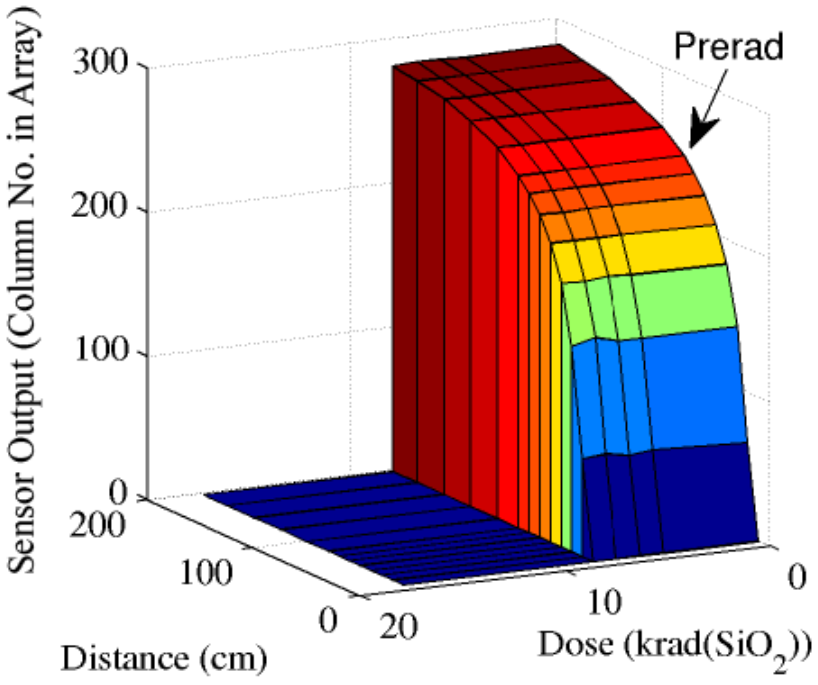

Figure 8. Transfer function for laser rangefinder for increasing TID. The plot shows the relationship between the serial output stream and distance for each TID value tested. Failure to respond is indicated as a sensor output value of 0 on the plot. Sensor shows no degradation until abrupt failure at 8 krad(SiO_2).

Figure 9 shows the transfer function versus dose for the sonar rangefinder. The sonar rangefinder showed similar abrupt degradation, but with a region of variation in the transfer function before complete failure. This region, which begins with mild variation around 10 and ends with severe degradation around 15 krad(SiO_2), presents a significant reliability concern, because the sensor's output is different than the pre-rad curve for the same physical distance. Follow-up tests identified degradation in a charge pump power integrated circuit as the source of the initial degradation, and microcontroller failure as the source of the final failure.

A version of this sensor from a different date-code batch had a different microcontroller subcomponent, and it showed total failure at 3 krad(SiO_2). This difference was marked as a version number on the part, (but was not marked on distribution websites). This difference in radiation sensitivity of failure at 3 and 15 krad(SiO_2) between two lots marked with the same part number highlights how much impact one subcomponent can have on the radiation hardness of a sensor assembly. Robot designers concerned about radiation hardness must take extra care when working with COTS sensor assemblies.

The infrared (IR) rangefinder input-output transfer function versus dose shown in Figure 10 exhibits non-monotonic degradation, and the sensor continues to be operational at 92 krad(SiO_2). The shape of the surface is attributed to the combination of radiation effects in the IR lens, the position-sensitive detector, and the analog circuitry between the PSD and the analog output. Figure 11 shows the ratio of the calculated distance to the actual physical distance using the pre-rad inverse transfer function and the post-rad sensor transfer function. This relationship is a measure of the error that the sensor variation causes at the robot system level. Each IR rangefinder tested showed degradation that was similar in its characteristics (non-monotonic) but had a different variation in its response surface.

Figure 12 shows the percent change in sensor output after 23 krad(SiO_2) exposure for three instances of the IR sensor. Significant part-to-part variation in the degradation is observed.

Figure 13 shows the same data as Figure 10, the IR sensor's transfer function, including the transfer function after a 90 hour room temperature anneal, in a two-dimensional format. The room temperature anneal allows time for the generated holes to migrate, impacting the device's response. The IR sensor anneals to a different calibration curve (compared to original pre-rad calibration curve) following 92 krad(SiO_2) exposure. The figure clearly indicates the non-monotonic transfer function change with dose. The characteristics of the degradation make recalibrating the sensor with a new look-up table at the new dose rate challenging, because if the accumulated dose and annealing are not accurately calculated, the estimate could be off significantly.

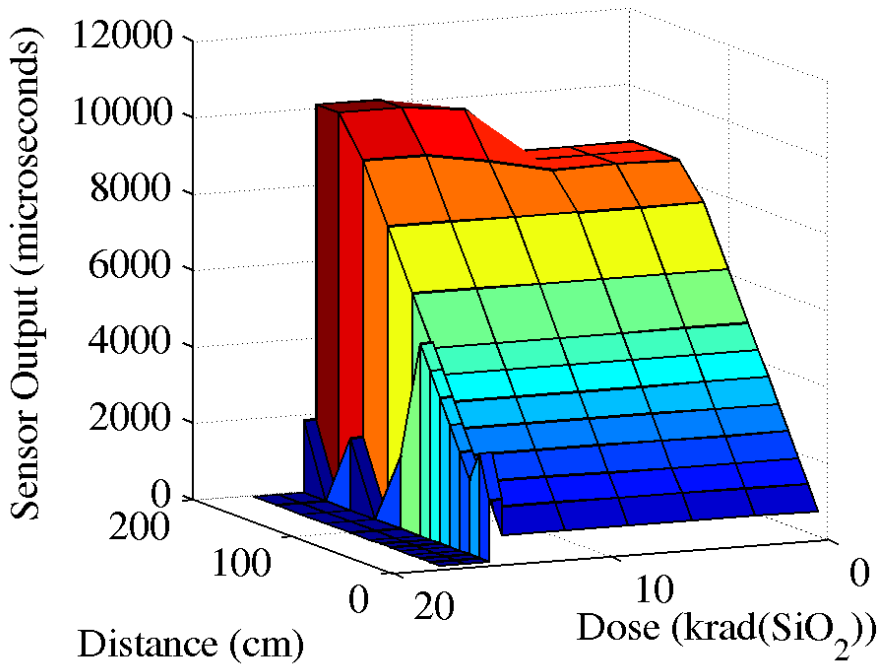

Figure 9. Transfer function for sonar rangefinder for increasing TID. The plot shows the relationship between the time for the echo to be detected and distance for each TID value tested. Failure to respond or the echo exceeding a maximum time is indicated as a sensor output value of 0 on the plot.

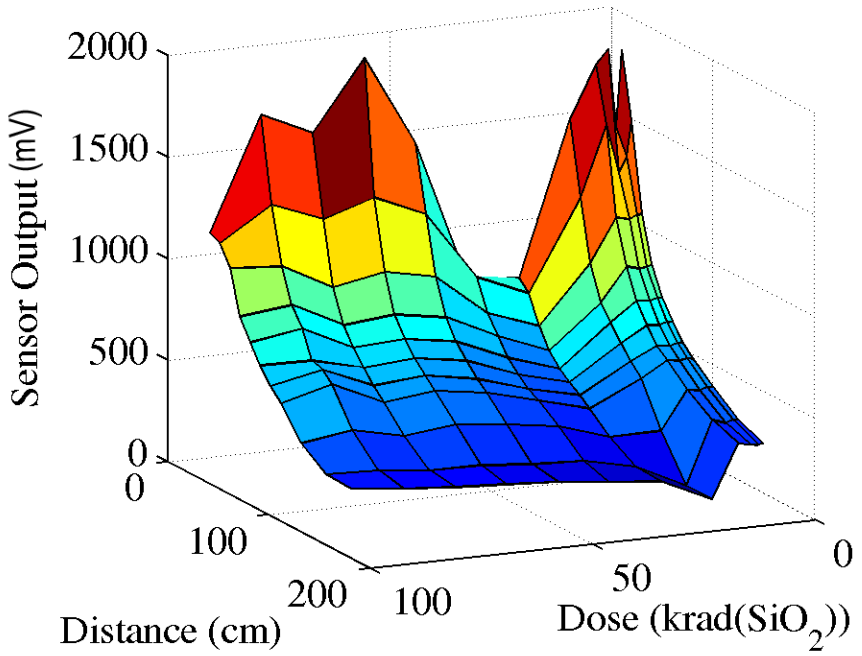

Figure 10. Transfer function for IR sensor for increasing TID. The plot shows the relationship between the sensor output voltage and distance for each dose level tested. Significant part-to-part variation was observed, and this plot serves as a representative case.

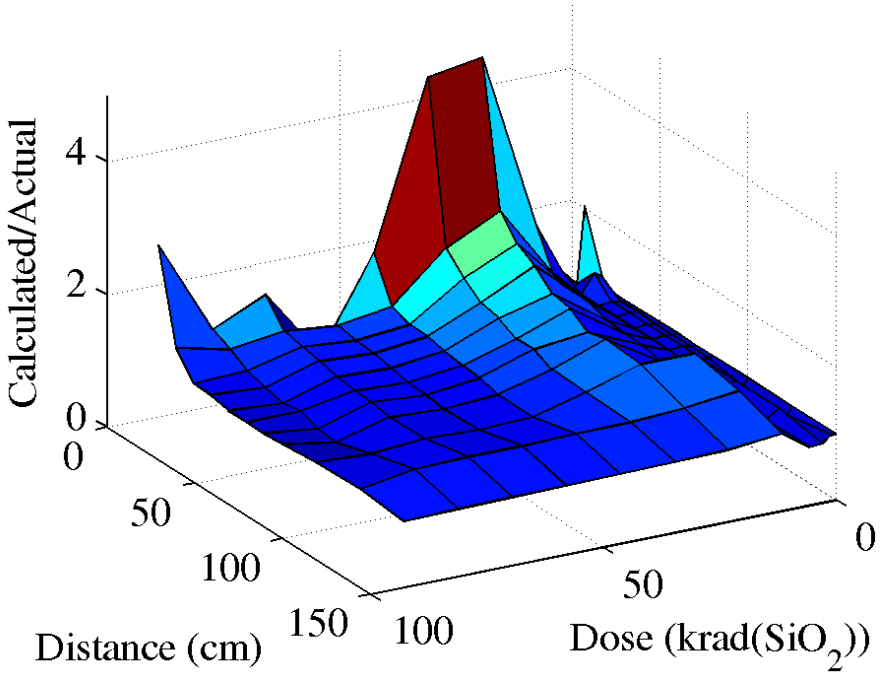

Figure 11. Ratio of post-rad calculated distance to actual physical distance for an IR sensor for increasing TID. The error from radiation is not distributed uniformly over the range of the sensor.

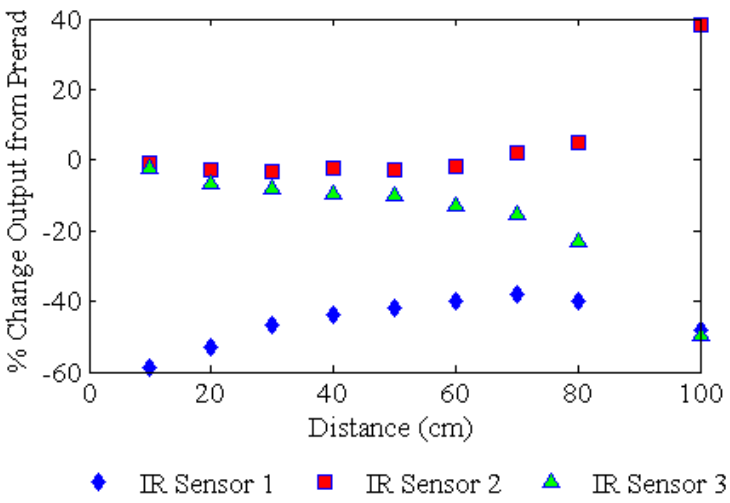

Figure 12. Significant part to part variability was observed for different instances of the same part number IR sensor. A slice in the dataset after 23 krad(SiO2) exposure for IR sensor. Sensor 1 corresponds to the sensor shown in Figure 10 and Figure 11.

Figure 14 shows the change in power supply current for increasing TID. Supply current was monitored because it is known to vary with TID [41]. Significant changes in supply current for the laser range finder and sonar indicate TID effects in the supporting electronics. Additionally, the two highest dose levels for the sonar (blue curve) indicate the transition to severe degradation and total failure as shown in Figure 9. The transition in the laser range finder curve that occurs around 8 krad(SiO_2) corresponds to the rangefinder becoming non-operational. These correlations indicate that supply current has the potential to be used as a health status indicator for the operational integrity of some (but not all, i.e., the IR rangefinder) of the sensors considered here.

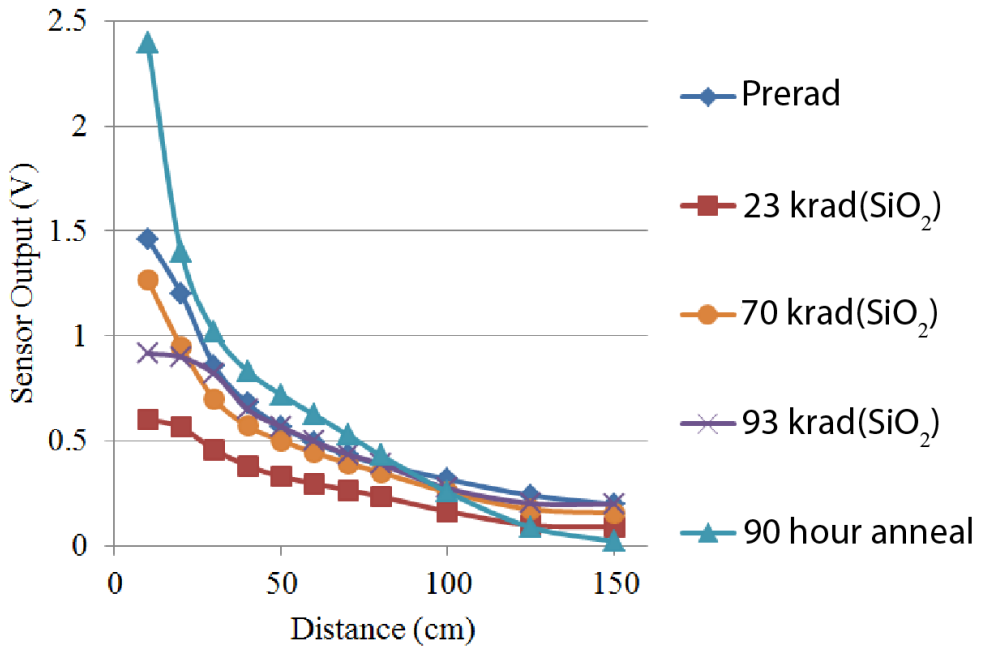

Figure 13. Transfer function including 90 hour anneal. The sensor anneals to a different transfer function than the pre-irradiation transfer function, complicating efforts to recalibrate the device by tracking the received TID.

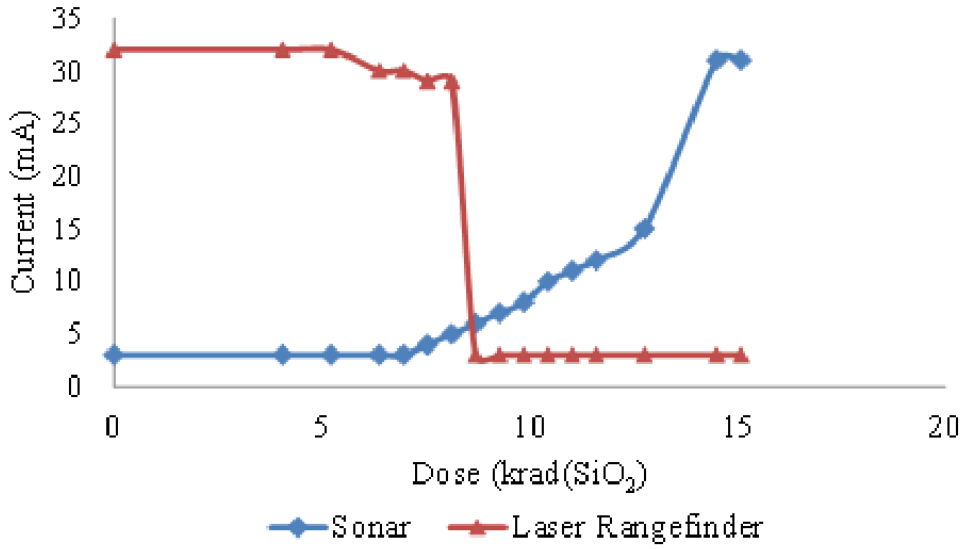

Figure 14. Supply current versus dose for the sonar and laser rangefinder. The degradation in supply current tracks strongly with the degradation in the input-output relationship of these sensors. The shifts are in different directions due to the supporting electronics.

Degradation Diagnosis

It is desirable to diagnose the source of the TID-induced degradation in order to predict expected operation of a sensor in the field and to serve as a starting point for a radiation hardening by design approach. The sonar sensor was analyzed in detail and follow-up tests were performed to investigate the shape of the surface plotted in Figure 9. The IR sensor was not a good candidate for detailed investigation because the circuitry existed in a single integrated package, and the laser rangefinder degradation could be attributed to a single component (the microcontroller responsible for communication). However, the sonar sensor was an ideal candidate because it showed degradation before abrupt failure and internal circuit nodes were accessible for monitoring.

Additional test points combined with a functional model of the sensor's operation can be used to diagnose the source of the features in Figure 9. The subcomponents of the sonar that were susceptible to radiation degradation are listed in Table 1. Possible degradation effects were assigned to each subcomponent based on experience with radiation effects on similar parts. Two op amps are used in the assembly (one for transmitting and one for receiving), but since they are two instances of the same component they share an entry in the table. The functional model for the sonar sensor, shown in Figure 15, was developed from analyzing the structure of the sonar sensor. For this sensor, the functional distinctions corresponded with the discrete components on the circuit board, but for a more involved sensor consisting of multiple subassemblies the functional distinctions could contain multiple discrete components.

An appropriate measurement was made on the sonar rangefinder board at each input/output transition in the functional diagram, for each dose step. For the voltage regulator and charge pump,

DC voltages and currents were measured. For the output of the op amp and piezo element, an oscilloscope was used.

The test point information was analyzed, revealing a dip in the charge pump output voltage from -5 V to -4.3 V at the dose level corresponding to the shift in far distance sensing at about 10 krad(SiO_2). This voltage shift explains the first feature of Figure 9. The transmitting power of the amplifiers was reduced due to the degradation in the charge pump, resulting in a signal that was weaker, changing the characteristics of the reflection at distances approaching the limit of the sensor's operation. At these dose levels all other subcomponents were operating with nominal characteristics.

TABLE I
POSSIBLE DEGRADATION EFFECTS FOR SONAR SUBCOMPONENTS

SUBCOMPONENT	DEGRADATION EFFECT
Voltage Regulator	Not provide correct voltage, consume excess power, not provide enough current
Charge Pump	Propagates regulator effects, does not provide correct negative voltage rail
Op Amp	Carries forward regulator/charge pump effect, reduced amplification, input offsets increase error
Piezo Element	Generates degraded signal (as result of degraded input signal), sensitivity of system is reduced
Latch	Change in latching threshold, sensitivity of system is affected
Microcontroller	Digital failure leading to non-responsive system, pulse generation characteristics change

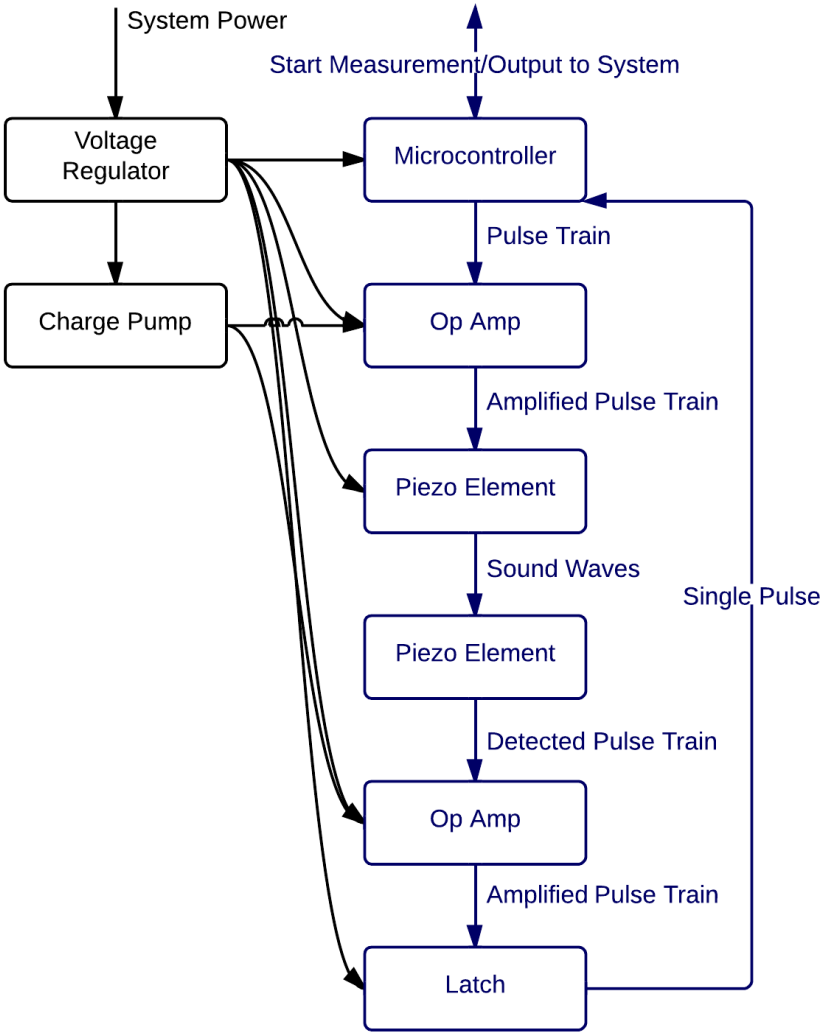

Figure 15. Functional diagram of the sonar sensor. Test points were placed at each line of the figure to determine the source of the degradation.

The second feature in Figure 9, the unstable behavior that begins after 14.5 krad(SiO_2) corresponds to the combined degradation of the charge pump and op-amps. The latch, microcontroller, and piezo element all tested nominally, but the test points for the charge-pump showed a significant voltage deviation and the op amp produced an unstable output. Finally, the total failure of the

device corresponds to the microcontroller no longer starting the pulse train when prompted, indicating that the first stage in the sensor has failed.

The approach of defining a functional diagram for the sensor, estimating possible failure modes, adding and monitoring test points to gain additional data, and then looking for values outside normal ranges at each test point would allow the system using the sensor to determine the health of the sensor in detail.

Impact on System Functional Performance

To evaluate the impact of degradation before abrupt failure, the measured pre- and post-rad rangefinder input-output transfer functions were used to create a map of a rectangular virtual room of the dimensions measurable by this sonar rangefinder. The measured sensor transfer functions were fed a data set representing the sonar scanning a room at a fixed elevation with a one-degree rotation between measurements. Figure 16 shows several images of the virtual room, one as it would be measured by the pre-irradiated sonar and others as the room would be measured by the sonar after receiving increasing doses of TID.

Pre-irradiation, the sonar data constructs an accurate representation of the room. However, the degraded sensor drastically misrepresents the room, becoming progressively worse with increasing TID. At 11.6 krad(SiO_2) the degradation at the long distances produces distortion of the far side of the room. At 14.5 krad(SiO_2) the sensor is no longer monotonic and highly degraded, producing a distorted shape. At 15.1 krad(SiO_2) the sensor was non-functional, reporting the same value for any distance measured. The sonar's degradation significantly distorts the room's shape, passing incorrect information upstream to the rest of the robotic system (the discontinuities are the result of the one-degree step size for the measurements). If this sensor was used alone for robot

navigation or mission planning the system would make incorrect assessments in both mapping and localization. The dose levels in Figure 16 were chosen to illustrate the key changes in output upstream to the system leading up to the sensor's failure.

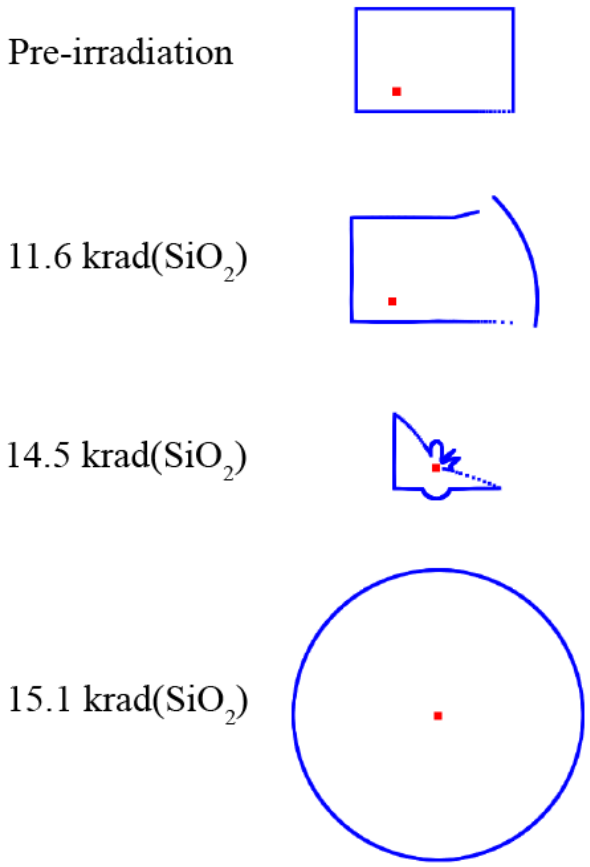

Figure 16. Simulation of how the sonar input-output relationship would map a room after different irradiation steps.

Conclusions

Dose levels realistic for a post-radiation-disaster environment induced faults in all three range-finding sensors investigated as robot radiated-sensor exemplars in this study. Both abrupt failure (laser rangefinder) and inaccurate sensing due to significant changes in sensor transfer function

(sonar and IR rangefinder) present significant challenges to a robot attempting to do useful work for long periods of time in a gamma radiation field. The degradation and failure in the sensors tested were primarily due to the supporting electronics rather than the transducer. This finding indicates that sensors that measure physical quantities other than distance, but which use similar supporting signal processing electronics, will likely show similar vulnerabilities and failure modes.

If the system relies on a pre-irradiation look-up table to interpret sensor signals, severe impairment of important robot functions such as mapping and localization can occur. This impairment can come in the form of incorrect signals propagating upstream, such as in the sonar room example. Mitigation strategies such as measuring radiation dose during field operation and adjusting the sensor models in the robot control system according to a pre-programmed pattern are demonstrated to be ineffective because of sensor part-to-part radiation response variability and room temperature annealing. High variability of transfer functions with radiation for COTS parts with the same part number present an additional obstacle to reliable operation and recalibration.

Measureable sensor characteristics that can be correlated to sensor performance/degradation, such as power supply current, could potentially be effective in making the robot aware of a sensor's current state of degradation before complete failure occurs, but supply current does not appear to be a universal proxy for radiation damage in all sensors. The diagnosis methodology used to analyze the degradation mechanisms of the sonar sensor can be extended to more complex sensor assemblies as a first step in developing a radiation-hardened-by-design approach for redesigning a sensor assembly, with the caveat that this approach is limited to sensors where internal circuit nodes are accessible.

The characteristics of a radiation-robust sensor will vary with the exact mission requirements, but the experimental data support several conclusions. First, the sensor should clearly indicate that it is functioning properly or improperly (through an increase in standby current or other reporting mechanism), allowing the system to know when to stop trusting the data from the sensor. This capability is essential for a robust sensor to prevent information that is no longer representative of the physical quantity being sensed being passed upstream to the robot control system. A sensor assembly may need to be altered from its standard commercial form so that internal circuit nodes can be monitored in order to obtain this characteristic. Second, a robust sensor would limit the impact of variations in a single component on the system performance. The sensor should be designed in such a way that variations in radiation-sensitive quantities such as supply current or amplifier gain will not impact the sensor transfer function unless the degradation is severe. This can be accomplished by sensing a differential quantity that can use thresholding to produce a high level of immunity to parametric drift in the internal components. An example of this is the laser triangulation approach sensing which column on the camera is brightest – as long as the pixels degrade similarly the same column will remain brighter comparatively. Sensors with these characteristics should be reliable in robotic systems intended for radiation environments.

The benefit of using commercial rangefinders with integrated supporting electronics are the low deployment cost and ease of system integration, and the conclusions above provide guidance on the types of commercial sensors that should be targeted for inclusion. Commercial sensors with operational principles similar to those studied in this work would be appropriate for surveying the disaster site and basic remediation tasks, providing multiple hours of reliable operation at anticipated dose levels, but extended operation in worst-case dose scenarios would require a part replacement strategy or TID hardened sensors. Further investigations into the impact of other

radiation sources (neutrons, alphas) as well as the combined TID-temperature degradation are necessary for extended operation in worst case scenarios, with the more challenging environmental requirements suggesting the need for customized rangefinder designs, such as isolating the transducer element from the supporting electronics and introducing additional radiation shielding materials, trading cost and ease of integration for radiation robustness.

While this chapter focused on rangefinders and sensors, the key outcomes are applicable to many types of uncharacterized electronic systems that may be operated in radiation environments. Visibility into the degradation and root causes is essential for any mitigations to be implemented. This prompted the following chapter, focusing on how to diagnose degradation in complex COTS systems.

CHAPTER IV

IDENTIFICATION OF HEALTH STATUS INDICATORS

Z. J. Diggins, N. Mahadevan, D. Herbison, G. Karsai, B. D. Sierawski, E. Barth, E. B. Pitt, R. A. Reed, R. D. Schrimpf, R. A. Weller, and others, "Total-ionizing-dose induced timing window violations in CMOS microcontrollers," *IEEE Transactions on Nuclear Science*, vol. 61, no. 6, pp. 2979–2984, 2014.

Introduction

The previous chapter investigated the TID degradation of several Commercial-off-the-shelf (COTS) systems. The ability to instrument the system by monitoring voltage rails, current profiles, and other expected outputs is invaluable to understanding the impact TID has on the system. For some systems, the ability to instrument with such fidelity proves difficult. This prompted the search for universal "Health Status Indicators", or parameters that vary with TID and can be used for diagnosing the status of the system or predicting its future performance. This chapter introduces a TID degradation measurement technique and health status indicator for observing and forecasting the degradation of digitally clocked systems.

Commercial-off-the-shelf (COTS) microcontrollers enable low-power, low-cost, and high-capability design solutions for electronic systems, and are embedded in complex mechatronic systems such as robots. Due to the highly integrated "black box" nature of microcontrollers,

previous reports of total ionizing dose (TID) induced failure in COTS microcontrollers are limited to abrupt failure at a measured dose level [42][43], [43], [44]. Because of their complexity and many integrated functions, microcontrollers present a hardness assurance challenge. Software and configuration changes also have potential impacts on TID hardness. Additional information about degradation and TID robustness would allow systems containing COTS microcontrollers to be deployed in radiation environments with increased reliability and provide microcontroller designers with information on how to increase the reliability of the design.

TID has been shown to impact propagation delays in individual transistors [45], the timing characteristics of integrated circuits [46], memory access times [47], and propagation delays in CMOS Flash-Based FPGAs [48]. In this work, timing window violations are experimentally demonstrated to be the primary source of failure in response to TID for a class of low power microcontrollers, and a model for the degradation and hardening implications are presented. The conclusion that timing window violations are the primary source of failure is supported by a measurement technique that allows insight into the internal degradation of the device during the radiation exposure. The technique is necessary when information regarding individual transistor degradation, the microcontroller technology, or the fabrication process is not available, such as incorporating a COTS mechatronic subassembly including an integrated microcontroller. Timing window violations are identified by performing a series of software tests on the microcontroller at various frequencies and supply voltages. The technique to determine the maximum and minimum frequencies and voltages for the microcontroller is more commonly used in electrical characterization [49]. Results are presented for a commercially available microcontroller, the Atmel ATMEGA328P [50]. Analysis of the experimental results shows that the degradation is consistent with the expected degradation of logic gate switching time based on charge build-up in

the gate oxide and shallow trench isolation [51]. The results are further verified using a subset of the software tests on a Microchip PIC16F677 [52]. The clock frequency measurement is non-destructive and can be monitored in a field deployment scenario to evaluate the health of the device, allowing for mission planning as well as life extension through reducing clock operating frequency or increasing supply voltage.

Background

Microcontrollers are integrated into a variety of systems that may be introduced to radiation environments, including medical devices, satellites, and robots. While implanted medical devices and satellites are often specialized designs where radiation robustness may be considered from the beginning of the design process, robots used for disaster mitigation (e.g., the Fukushima-Daiichi nuclear incident) usually consist of many integrated subassemblies. Previous work has shown that microcontrollers are the primary source of TID induced failure in several range-finding sensors for robotic applications [53]. Information during field deployment about the remaining useful life or TID margin would improve robotic mission planning.

Microcontrollers are often fabricated in high voltage processes in order to easily integrate with a variety of other systems, such as motors or sensors operating at 5 V or higher. The high voltage process necessitates thicker oxides than a modern microprocessor used solely for computation would, which can increase the vulnerability of the microcontroller to TID [54]. This potentially increased TID vulnerability makes the microcontroller a key component to analyze and a potential "weak link" for a subassembly in an ionizing radiation environment.

Timing Window Violations

TID generates trapped charge in oxides and at interfaces [37], potentially changing the threshold voltage and increasing leakage currents that can result in various circuit level impacts including increases in propagation delays [46]. The effects of TID are present before degradation or failure under nominal operating conditions can be observed. For microcontrollers, timing windows and propagation delays are of particular interest because the device operates using one central clock. As opposed to microprocessors with highly pipelined architectures, most microcontroller execute all instructions in only a few pipeline stages, with the entire ALU operation often being completed in one clock cycle. Figure 17 shows a simplified illustration of the timing characteristics of a single-cycle microcontroller. If TID increases the propagation delay of a data path past its allotted timing window, then incorrect data may be latched and propagated onwards.

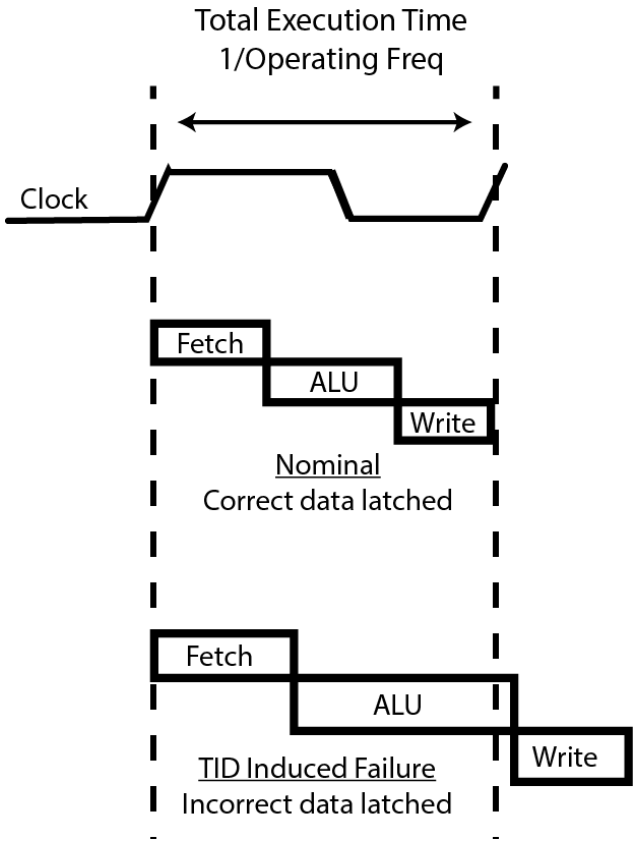

Figure 17. Timing window illustration for microcontroller architecture where the entire instruction is executed in a single clock cycle. Nominal demonstrates correct operations, while the TID induced failure portion demonstrates incorrect data being latched due to increased propagation delays.

Propagation Delays and TID

Timing effects have been established as a significant source of IC degradation for CMOS devices, with substantial increases in propagation delay resulting in timing failures [47]. Significant increases in propagation delay have been demonstrated for FPGAs, including >2x changes [48] and between 4 and 11X [56] in test circuits. These FPGA test circuits represent simplified versions of the circuits that make up a microcontroller arithmetic logic core, indicating that significant increases in propagation delay for microcontrollers can be expected. Additionally, published results with ring oscillators in a high voltage process revealed significant changes in the

propagation delay characteristics of the inverters, resulting in significant changes in the frequency of the ring oscillator [55]. While the ring oscillator experiment had transistor-level information available, COTS microcontrollers are complex systems with many thousands of transistors making the supply voltage and timing window technique demonstrated in this paper necessary.

Timing Window Hypothesis

Two parameters are controllable in the investigation of timing windows in microcontrollers: supply voltage and operating frequency. Figure 18 is an illustration of the relationship between operating frequency, supply voltage, and TID for a device irradiated at a fixed bias voltage and frequency, showing the maximum operating frequency that a given software routine passes successfully. The equations governing propagation delay and their relationship with TID are analyzed in [55]. The analysis contained in [55] is discussed below in the context of not only inverter chains but also generalized clocked logic such as microcontrollers. The drive strength of the transistors in the chain of logic elements is proportional to the supply voltage. The supply voltage exhibits an inverse relationship with propagation delay, as shown in (5), resulting in a proportional relationship with maximum operating frequency, where t_d (4) is the propagation delay for a logic stage, t_{pHL} and t_{pLH} are the high-to-low and low-to-high propagation times, μ_n is the NMOS mobility, C_{ox} is the oxide capacitance, $(\frac{W}{L})_n$ is the width-to-length ratio for the NMOS, V_{DD} is the supply voltage, α_n is defined by (6), and V_{Tn} is the NMOS threshold voltage:

$$t_d = \frac{1}{2}(t_{pHL} + t_{pLH}) \tag{4}$$

$$t_{pHL} = \frac{C}{\mu_n C_{ox}(\frac{W}{L})_n V_{DD}} \alpha_n \tag{5}$$

$$\alpha_n = \frac{8V_{DD}^2}{7V_{DD}^2 - 12V_{DD}V_{Tn} + 4V_{Tn}^2} \quad (6)$$

Similar equations exist for the low-to-high propagation time. The operating frequency directly controls the length of the timing window. Running software tests at different frequencies allows observation of timing windows, with passing tests for sufficiently long timing windows (low frequencies) and failing tests for timing windows that are too short (high frequencies). Importantly, the total propagation delay can increase even if one of the terms (t_{pHL} or t_{pLH}) decreases if the two terms are not balanced optimally.

With increasing TID the maximum operating frequency at which the device is functional decreases. By overclocking the device past its nominal frequency, degradation is observable before the nominal failure point. Finally, the device remains operational past its nominal failure point if operated at a reduced frequency.

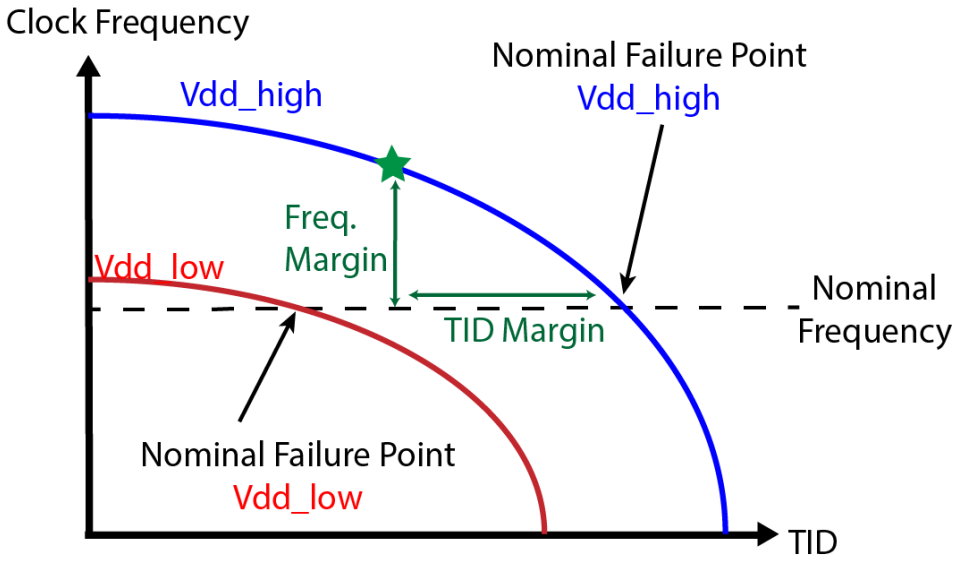

Figure 18. Illustration of timing window violation hypothesis, demonstrating the relationship between TID, operating frequency, and supply voltage (Vdd).

Experimental Design

The Atmel ATMEGA328P microcontroller was chosen to test the frequency/voltage timing window hypothesis described in the previous section. This device was chosen because it has a low power, single execution cycle architecture (indicating many logic stages in each pipeline stage), and has a variable core operational voltage of 1.8 V – 5.5 V. In order to validate the results for the ATMEGA328P and extend the results to a broader range of microcontrollers, a Microchip PIC16F677 was tested with a subset of the software tests using the same test procedure.

In order to determine if timing windows are being violated, test code must be run on the microcontroller. Functional unit tests were designed to target individual sections of the microcontroller [50], as shown in Figure 19. Industry standards for start-up tests and self-tests including the IEC60730 Class B compliance [56, p. 998] guidelines governing functionality tests for microcontrollers used in controlling machinery were consulted when developing the software functional unit tests. The test software routines were designed to be decoupled but dependencies on the same hardware exist between tests. For example, the digital I/O is necessary for communication even if the device passes the SRAM tests. The seven tests were: digital I/O, SRAM, FLASH, opcode, counter, comparator, and ADC.

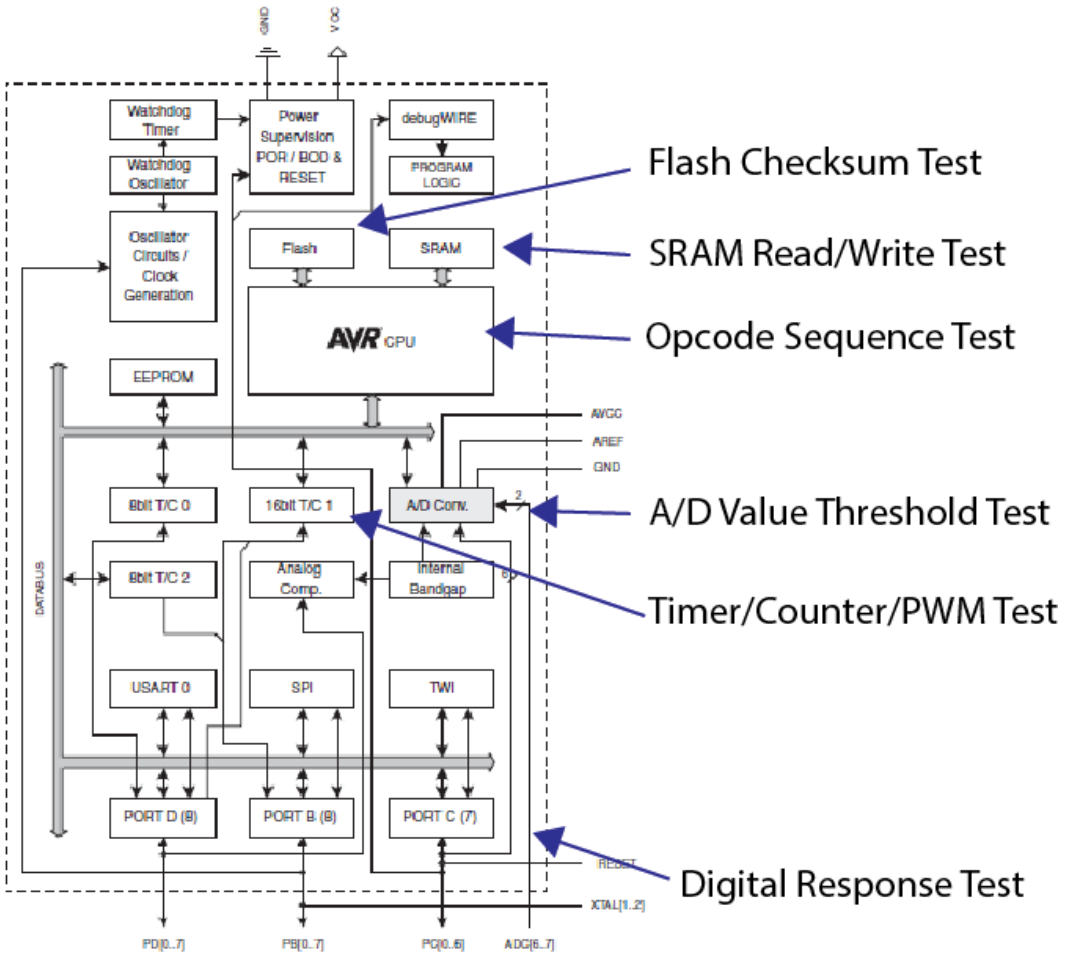

Figure 19. Block diagram from ATMEGA328P datasheet showing the microcontroller architecture. Arrows and text added to highlight the different regions of the microcontroller that the software functional unit tests exercise [50].

The overall test procedure was automated, in order to test a wide range of frequencies, supply voltages, and software functional unit tests after each irradiation dose step. A master/slave configuration was used to test the microcontrollers, where each slave Device Under Test (DUT) received commands and responded, while the master (an Arduino Due board) issued commands and evaluated the slave's responses for correctness. A Tektronix AFG3252 was used to vary the clock frequency in 1 MHz steps, and a Keithley 2410 Sourcemeter was used to vary the voltage and measure the supply current. The system was controlled through a MATLAB script using serial and GPIB communication. All TID experiments were performed using a Cesium-137 gamma

source at a dose rate of 600 rad(SiO_2)/min. After each dose step the DUT was removed from the irradiator and tested.

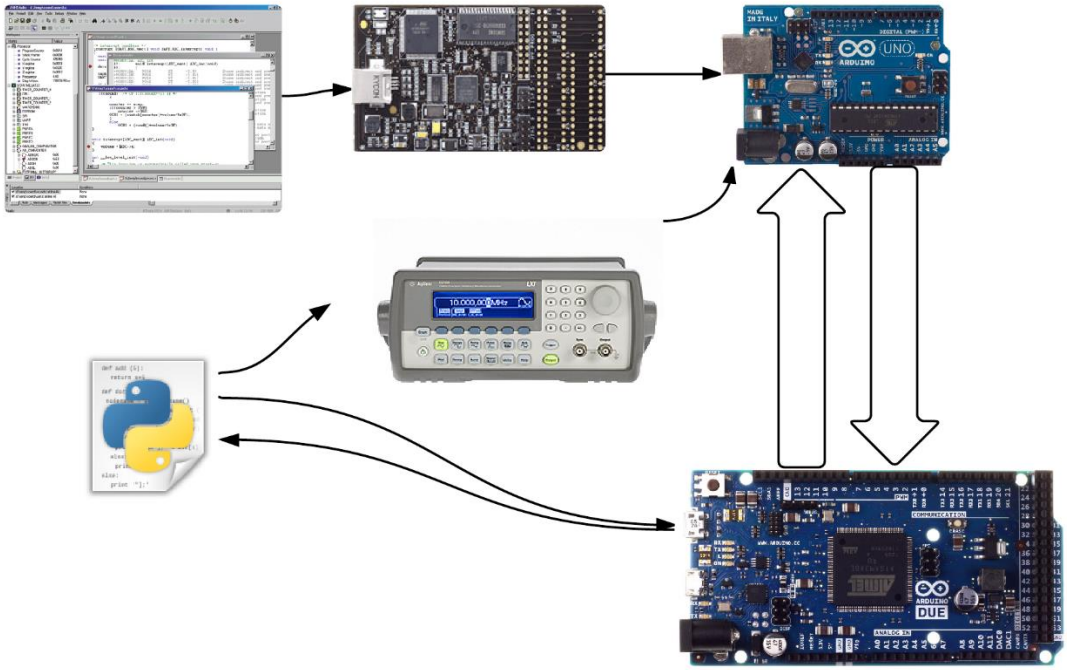

Figure 20. Diagram showing the control flow for the master-slave microcontroller test configuration. Icons indicate test code written in C (top left), programmer board (top middle), test device board (Arduino Uno, top right), python script (bottom left), Keithley Sourcemeter and function generator (middle), and interrogator master processor (bottom right).

Experimental Results

The ATMEGA328P was irradiated while biased at 0 V and the timing window test procedure was performed for seven software functional tests operating at 3.3 V, repeating the process for each dose step. The bias condition under irradiation of 0 V does not represent the worst-case bias condition but does place the device in a known state (biased and clocked operation would produce software dependent duty cycles for the transistors, and biased and un-clocked operation would produce different logic states for the transistors depending on when the clock was removed or the power up status). The data for each test are shown in Figure 21. The steps in the graph are a result

of the 1 MHz step size in frequency changes, with the data point indicating the highest frequency at which a test passes and the error bars indicating the distance to the next 1 MHz resolution step. The functional tests for the counters and comparators overlap with the digital I/O test results therefore they have been omitted for clarity. The results show a decreasing maximum operating frequency for increasing TID consistent among all tests. Reducing the clock frequency allowed the device to remain functional past the TID failure level observed at nominal frequency. The results are consistent with the hypothesis that timing windows are the source of the failure because all software functional tests demonstrated the same trend with TID. The slight difference in the curves for the different software tests is likely a result of the different critical paths involved in the distinct operations. These experiments were repeated for multiple bias voltages and repeated for three parts. The datasets in Figure 21 were chosen for illustration because they have the highest step resolution in TID, producing the most detailed curves.

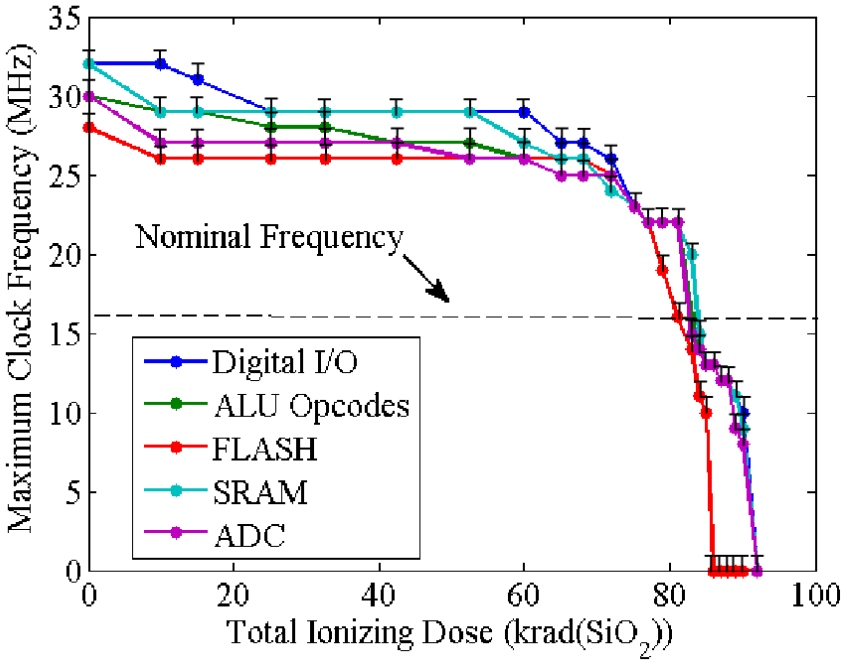

Figure 21. Results of test procedure for a representative part for the ATMEGA328P microcontroller. The maximum operating frequency for which the tests pass decreases with increasing TID. Error bars correspond to 1 MHz frequency measurement step size.

Additional irradiations were performed on the ATMEGA328P while biased at 0 V. The device was then biased at 1.8 V, 2.7 V, and 3.6 V to determine the maximum frequency that the tests passed for each bias voltage. The data, shown in Figure 22, exhibit the voltage dependence described above, yielding increasing maximum operating frequency with increasing supply voltage.

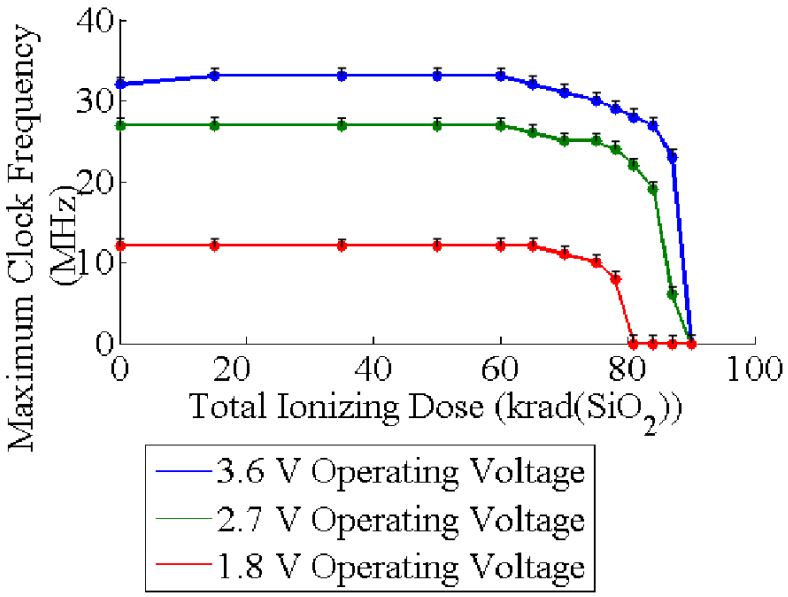

Figure 22. Results of test procedure for a representative part for ATMEGA328P microcontroller. Radiation steps were performed with all pins grounded, and then the test procedure to find the maximum operating frequency was performed for the three voltages. The data shown are for the opcode test. Error bars correspond to 1 MHz frequency measurement step size.

Figure 23 shows the maximum operating frequency for the PIC16F677 versus TID on the left axis with the solid line and the leakage current, measured by stopping the clock to the device, on the right axis. The PIC16F677 was biased at 3.3 V during both irradiation and testing. The software functional tests used to determine the maximum operating frequency was the lowest frequency of the digital I/O, flash memory, and opcode tests. The significant leakage current increase supports the conclusion that leakage current significantly impacts the timing windows and contributes to the device's failure.

Analysis of TID Induced Timing Window Violations

The results support the hypothesis illustrated in Fig. 2, that supply voltage and clock frequency can be used to detect and quantify timing window violations in microprocessors. Both the ATMEGA328P and the PIC16F677 exhibited similar decreases in maximum operating frequency

with increasing TID. The voltage relationship with maximum operating frequency is consistent with the interaction between timing windows and drive strength. The leakage current increase measurement for the PIC16F677 suggests that TID induced leakage paths play a significant role in the degradation.

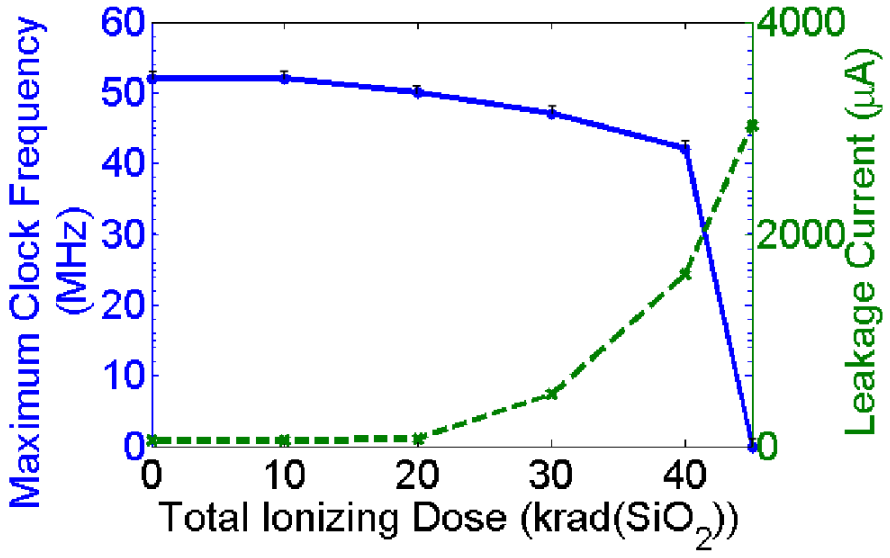

Figure 23. Results of the test procedure for a representative part for the PIC16F677 microcontroller with the part biased at 3.3 V during irradiation. Similar to the previous plots the maximum operating frequency for which the tests pass decrease with increasing TID. The leakage current is included on the right side axis. Error bars correspond to 1 MHz frequency measurement step size.

Relationship Between TID and Timing Windows

When the microcontroller is exposed to TID, charge trapping in gate and field oxides will produce three primary effects in Equations 1-3:

1. Charge build up in the gate oxide will impact the logic drive strength and subthreshold characteristics of the NMOS and PMOS devices, unbalancing the low-to-high and high-to-low propagation times, resulting in a greater overall propagation time.

2. Leakage in the STI of the NMOSFET will impact the PMOS to NMOS drive strength ratio, increasing low-to-high propagation time and decreasing high-to-low propagation time.
3. Leakage in the STI of the NMOSFET will prevent the output from reaching the true V_{DD}, impacting the drive strength of subsequent stages and potentially preventing signal propagation.

Without transistor level degradation information and detailed information about the microcontrollers' architecture, weighting the impact of these three effects is not possible, but all contribute to produce the degradation characteristics exhibited in Figure 18.

Operating Point Tradeoffs

The experimental technique allows for tradeoffs between operating lifetime in a radiation environment, supply voltage, and operating frequency. Within the system's electrical limitation, the operating point with the longest lifetime is when the supply voltage is at its maximum and the system operating frequency is at its minimum. This point provides the greatest lifetime for a given amount of TID damage because it has the highest drive strength and the largest timing windows of the four corners bounded by supply voltage and operating frequency. It is possible that another operating point should be chosen due to power concerns, and reduced supply voltages could potentially reduce the amount of charge that becomes trapped. A potentially attractive option is to place the microcontroller in a low power sleep mode to help reduce TID damage, and then operate the device at the maximum supply voltage during active operation.

Empirical Model of Maximum Operating Frequency

The ability to model a multi-thousand gate device such as a microcontroller with simple metrics facilitates a comparison between devices and allows the implementation of health tracking in-situ. An empirical fit of the data sidesteps the limitation of not having transistor level information available. Fundamentally, the condition for timing windows to be satisfied is, where t_{d_total} is the total propagation time for the critical timing path:

$$\frac{1}{clock\ freq.} - t_{d\_total} > 0 \tag{7}$$

Equation (7) can be redefined in terms of maximum operating frequency:

$$\max freq. = 1/t_{d\_total} \tag{8}$$

The results presented in the previous sections suggest that TID decreases the maximum frequency, which can be written as:

$$\max freq. = \max freq_{initial} - \frac{1}{t_{delay\_TID}} \tag{9}$$

Where the t_{delay_TID} can be further broken down into average delay increase per gate times the number of gates in the critical path in a single clock cycle:

$$t_{delay\_TID} = (logic\ depth) * (TID\_delay\ /\ gate) \tag{10}$$

This model could be extended to include voltage dependence as well if more voltages were tested, resulting in a new term that interacts with the $TID\_delay\ /\ gate$ term. The TID delay per gate

increases exponentially with TID, which is consistent with the exponential increase in leakage current shown for the PIC16 part in Fig. 6 and other studies of STI leakage [18]. In Figure 24, the exponential model is shown to fit the data, allowing comparisons between different microcontrollers and forecasts of remaining useful life in the field. The "a" coefficient corresponds to the pre-irradiation maximum operating frequency, the "b" coefficient is proportional to the number of gates in the critical timing path, and the exponential term corresponds to the impact of the TID on the gate's propagation characteristics.

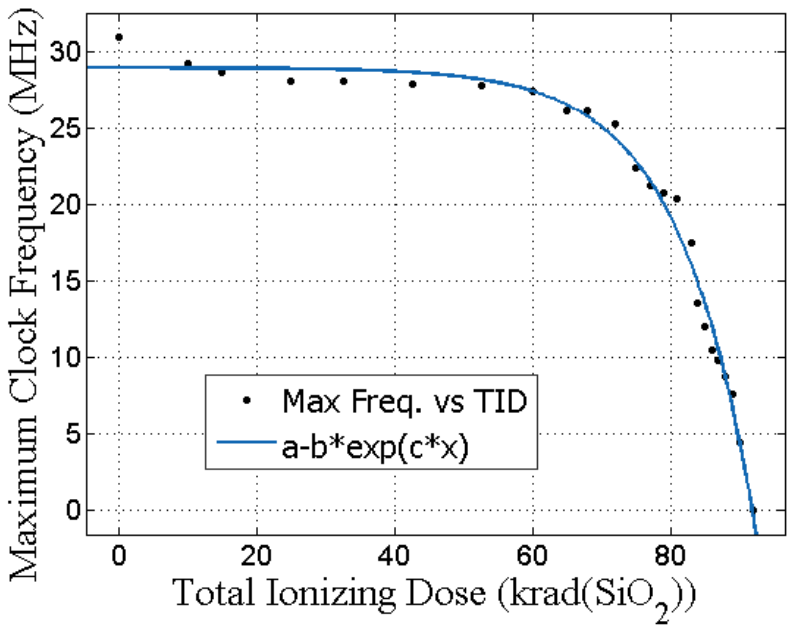

Figure 24. Data set for ATMEGA328P biased at 3.3 V and irradiated with all pins grounded for the ALU opcode test for "a" = 28.25, "b" =0.004225, "c" = 0.08827, and R2 = 0.9690.

Values for the fit coefficients for the tests shown in Fig. 4 are detailed in Table I. Since the fit is determined by a single exponential term, it suggests that one mechanism dominates the TID response, supporting the conclusion that timing window violations are occurring for the microcontrollers tested instead of a combination of other potential sources of TID-induced failure. Without detailed processor architecture information, it is not possible to determine definitively if

TABLE II
MODEL FIT COEFFICIENTS

Test Name	"a" Max. Freq.	"b" Pipeline Depth	"c" Impact of TID	R^2
Digital I/O	30.31	0.008195	0.08827	0.9697
ALU Opcodes	28.25	0.004225	0.09474	0.9690
FLASH	27.47	0.006952	0.09246	0.9260
SRAM	29.27	0.005214	0.09285	0.9740
ADC	27.35	0.001732	0.1045	0.9718

Coefficients correspond to a-b*exp(c*x) model and data from Figure 21 and Figure 23.

the trends in the coefficients match the proposed model, but the results behave as expected, showing more variation in the pipeline depth term "b" (which is expected to vary between tests) than the radiation response term "c" (which is expected to be the same between tests). Similar timing models could be useful for other clocked digital systems, such as FPGAs or ASICs.

Propagation of Digital "1"

When the leakage in the shallow trench isolation of the NMOSFET approaches the drive current of the PMOSFET in an inverter, the ability to output logic "1" degrades. In a microprocessor, there

may be a chain of logic gates in which logic "1" must be propagated for several gates in a row, but the leakage prevents this propagation because the magnitude of the leakage is on the order of the drive strength of the pull up network. This effect can occur at an arbitrarily low system clock frequency. The ultimate failure of the devices tested is potentially due to this effect. The impact of logic "1" degradation can be mitigated through the use of pipelining. By implementing registers between a chosen number of logic gates the longest path of all logic "1" outputs without a register stage will be limited.

Conclusions

Clock frequency and supply voltage are effective diagnostic tools to evaluate the influence of TID on microcontroller timing windows. Varying clock frequency and supply voltages provides a dynamic non-destructive technique for evaluating TID performance margin of microcontrollers in the field. Designers should choose the minimum frequency that fits the design specifications and increase the supply voltage as necessary for maximum TID hardness for microcontrollers and other components with architectures and characteristics comparable to the microcontrollers tested.

CHAPTER V

DISCRETE BAYESIAN ANALYSIS FOR DIAGNOSIS AND PROGNOSIS

Z. J. Diggins, N. Mahadevan, E. B. Pitt, D. Herbison, G. Karsai, B. D. Sierawski, E. J. Barth, R. A. Reed, R. D. Schrimpf, R. A. Weller, and others, "System Health Awareness in Total-Ionizing Dose Environments," *IEEE Transactions on Nuclear Science*, vol. 62, no. 4, pp. 1674–1681, 2015.

Introduction

The fundamental question of interest when using electronic components in radiation environments is whether the components operate successfully together as a system for the desired application and environment (thus completing the desired mission). The primary factors in system missions' success are the nature of the radiation environment and the degradation characteristics of the components. In addition, a high degree of uncertainty in system radiation tolerance estimation is introduced because of uncertainty in the radiation environment, part-to-part variability in radiation sensitivity, and the changes in operating conditions, amongst other factors. For high reliability applications, the uncertainty in operational outcome must be eliminated or reduced. This high reliability is currently and successfully accomplished by designing systems using radiation-hardened components, component- and system-level testing, maintaining significant design margins, and procurement practices that ensure that the components used are similar to the

components tested [14]. These measures have proven successful for many high reliability applications such as space electronics.

A complementary concept to assurance, or guaranteeing successful operation, is awareness, understanding how and why the system performs as it does while operating in a TID environment. Awareness is useful during the field deployment to evaluate which functions can be performed, and during design to guide improvements to the system. Using systems engineering principles and Bayesian networks, this work presents a framework for quantitatively estimating the health of a system in a radiation environment, including the many interactions between the components. This approach has two primary applications. First, systems built with mostly commercial-off-the-shelf (COTS) components that must operate in a radiation environment, including small, inexpensive satellites (Cubesats, satellite cluster formations) and robots for application in the nuclear power industry, cannot afford the costs of a full custom design including rad-hard parts. Second, during the design phase of systems that are undergoing a full custom design and hardness assurance process, detailed information about the TID response of every component under consideration for use in the design may not be available. Also if the system's TID response is not acceptable, sorting through the complex interactions leading to the system's failure can make identifying the most effective design change difficult.

In such cases, an awareness of the TID response of the components and their impact on the overall system health and performance would provide significant value. Such system-level hardness awareness schemes could enable minimal/ incremental design changes (local part selection/ shielding/ hardness improvement) in COTS systems or custom-designed high reliability systems to increase the overall system reliability in a TID environment.

A case study of a robotic system consisting of COTS components is used to illustrate the Bayesian network paradigm, with four specific examples highlighting different applications of the Bayesian network approach to model the performance of the robot in a total-ionizing dose (TID) radiation environment. A comparison to existing hardness assurance methods is included in the discussion section.

Bayesian Network Construction for TID Awareness

Using a Bayesian network can enable awareness of the impact of radiation-induced degradation on the variables of interest in a given system. An advantage of Bayesian networks is that the structure includes causality and independence information, enhancing prediction of system performance. While the structure can be learned from data using various algorithms, for electrical/mechanical systems there is significant information available about component interactions and performances in the form of datasheets, schematics, physics-based models, and expert knowledge. This work presents a flow for developing Bayesian networks for understanding the impact of TID on component and system performance using schematics, fault diagrams, key functional diagrams, and ultimately a multilayered structure to construct the Bayesian network. To better illustrate the methodology, a specific case-study of a line-tracking robot system is considered. This example is concise enough to present in this context, but highlights the major aspects of many electrical systems that application engineers could model, including sensing, actuation, control, and power electronics.

Building a Model of the System

The modeling process used for developing the Bayesian network determines the awareness and inference available from the model. The following sections provide a template for the modeling process, including the key considerations at each step.

Schematic/Block Diagram

The schematic/block diagram model captures the components involved in the system and their interactions in terms of energy and signal flow. As part of analyzing the impact of TID on the system, it is important to obtain parts lists, schematics, and other information about the system at the appropriate detail level. The appropriate detail level is determined by the user/designers' needs, such as which measurements can be made and what components can be potentially modified or replaced. For an in-house, custom-designed system this information is available in great detail and may be performed at the component level, providing information about specific components in subsystems. For a COTS system, modeling at the subsystem level may be appropriate.

The block diagram of the robot used as an example in this work is shown in Figure 25. The system consists of a microcontroller, infra-red reflectance line sensor, two motors, a two channel H-bridge driver, a voltage regulator, and a battery. For the sake of conciseness, the subsequent sections discuss a subset of the components, focusing on the line sensor and voltage regulator. These two components yield the minimal Bayesian network that illustrates the features of the modeling method, but the analysis could be expanded to the entire system or an arbitrarily large system including hundreds or more components.

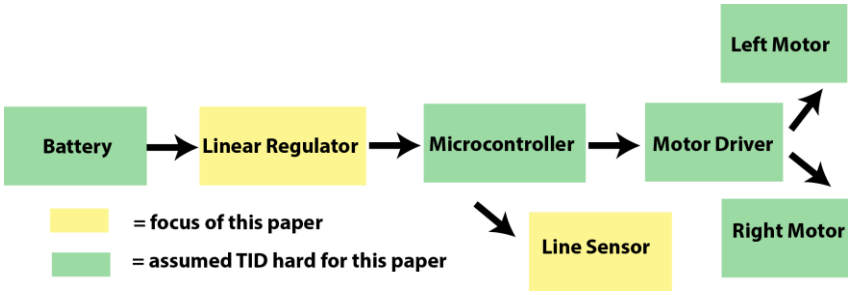

Figure 25. Block diagram of case study system. Blocks represent key physical sections of the system and lines indicate signal or energy paths. Arrows indicate dependencies for functionality, but in Bayesian networks all arrows are bidirectional so direction is unimportant.

Fault Propagation

After developing the block diagram/schematic model for the system of interest, the next step is to use expert knowledge about the components and subsystems to capture the fault propagation model. Since this work is focused on studying the impact of radiation, the fault-model considers those component and system-level fault sources (failure modes) that are related to radiation exposure. The fault propagation model captures the downstream effect of each failure mode, which could include inducing anomalies (observable and unobservable), function degradation, and possibly causing other failure modes. The information obtained from the fault propagation model is useful in the context of building the Bayesian network, to determine the observable anomaly nodes that are to be related to specific components, as well as the causal relationships between the component nodes.

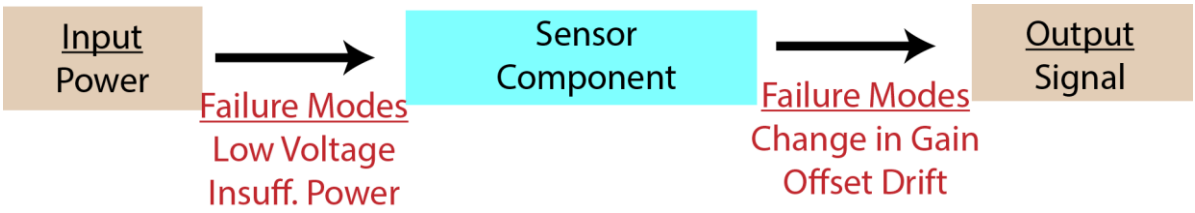

Figure 26. Failure modes for the line sensor. It can receive degraded power from the linear regulator and pass a degraded output to the microcontroller.

Figure 26 contains a simplified fault propagation model for the line sensor. It indicates the cascading effect of failure in the power-source on the operation of the sensor component, thereby affecting the quality of the sensor output signal, which could affect the performance of a downstream component/node.

Functional Dependencies

The functional dependency model determines which components and sub-functions must be operating in order for the desired higher level function to operate successfully. The diagram is orthogonal to the fault propagation model and captures the functional failures that are introduced when components/subsystems fail/degrade. The diagram does not need to be complete or capture all functionalities and can be constructed using the system expert's best estimate as to the functional dependencies. Figure 27 shows a model of the functional dependencies for the line-tracking robot. The ultimate high level function is tracking, which requires the sub-functionalities of motor control, sensing, and power. Through clearly identifying the functional dependencies, the Bayesian network can be appropriately refined to include nodes pertaining to specific-desired functionality, including their sub-functions (if desired). The functions (and sub-functions) can be connected with the associated component-nodes that provide the desired functionality.

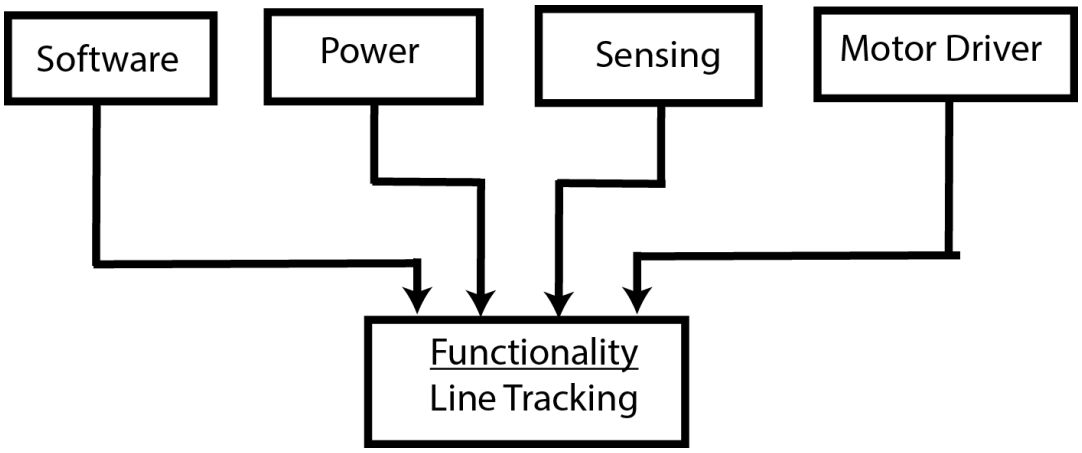

Figure 27. Model of functional dependencies for line follower robot.

Designing the Bayesian Network

A variety of Bayesian Network software packages exist. For this paper, the networks are created using GeNIe 2.0 [57], a software tool developed by the University of Pittsburgh's Decision Systems Laboratory .

Fig. 5 shows a simplified version of the full Bayesian network that can be constructed for the line-tracking robot based on the information available in the schematics/block diagram, the fault propagation model, and the functional dependency diagrams. While the simplistic structure of the case study network could be determined through inspection, the processes presented in this work can be used for constructing an arbitrarily large network.

The top node in the network represents the amount of TID each component has received. It is divided into six states, ranging from under 10 krad(SiO_2) to under 60 krad(SiO_2). In this network, a single TID node is used to represent the TID received across all components. In cases where the components have varying degrees of TID exposure, it might be desirable to have a network with multiple TID nodes, each impacting a different set of components. Additionally, a "Time" node can be included above the TID node if there is uncertainty in the time-to-TID mapping (i.e., the

dose rate) in the environment. This flexibility in network design for numerous situations is a key advantage for Bayesian networks for system health awareness in TID environments.

The next level of nodes in the network (the child nodes of the TID node) corresponds to the health of the components. Inference can then be used as feedback to update the distributions of the component level nodes to which the observation nodes connect in the network. The line sensor and linear regulator nodes capture the health of the respective components. Each of these nodes has three states: good, degraded, and failed. For the linear regulator the output voltage is the most important quantity and is directly used to describe the health of the node. For the linear regulator node's bins, good is defined as a voltage between 5 V and 4.8 V, degraded is defined as a voltage between 4.8 V and 4.5 V, and failed is defined as an output voltage below 4.5 V. The choice of these specific states and the choice of three states were determined by the characteristics of the system. The number of node states and transition points can be determined by the system designer. The line sensor has both the linear regulator and TID as parents. The addition of the connection between the linear regulator and line sensor was determined using the functional dependency and fault propagation diagrams.

Wherever possible, each of the component nodes in the Bayesian network model should include one or more child nodes that are related to measurements/observations that can be related to the health of the component. The "White Surface Sense" (the quality of the signal over a highly reflective surface) and "Supply Current" nodes correspond to the observable anomalies in the sensor and linear regulator components, respectively. They help characterize the health-status of their parent nodes. The use of observation nodes helps incorporate additional measurement information into the system in a quantitative manner. Often, variables of interest cannot be measured directly, but other variables that correlate with variables of interest can be measured

[58], [59] and hence incorporated in the Bayesian network. Though the "White Surface Sense" does not characterize the entire performance of the line sensor, it sheds light on the degradation (due to TID) of a very important part of the line sensor – the infra-red transmitter receiver pair [58]. Similarly, the "Supply Current" node does not directly describe the linear regulator but has been shown to vary with the linear regulator's health [60]. Examples of this type of inference are included in Section IV. B.

The next and final level of nodes in the Bayesian network corresponds to the system-level functionality. The parent nodes for these functionality nodes include component nodes and/or other functionality nodes. In this case, only one functionality node (tracking) is used in the Bayesian network model. This node corresponds to the highest level functionality (listed in the functional diagram). The node has three states. It has two parent nodes (the component nodes corresponding to sensor and linear regulator) and no child nodes.

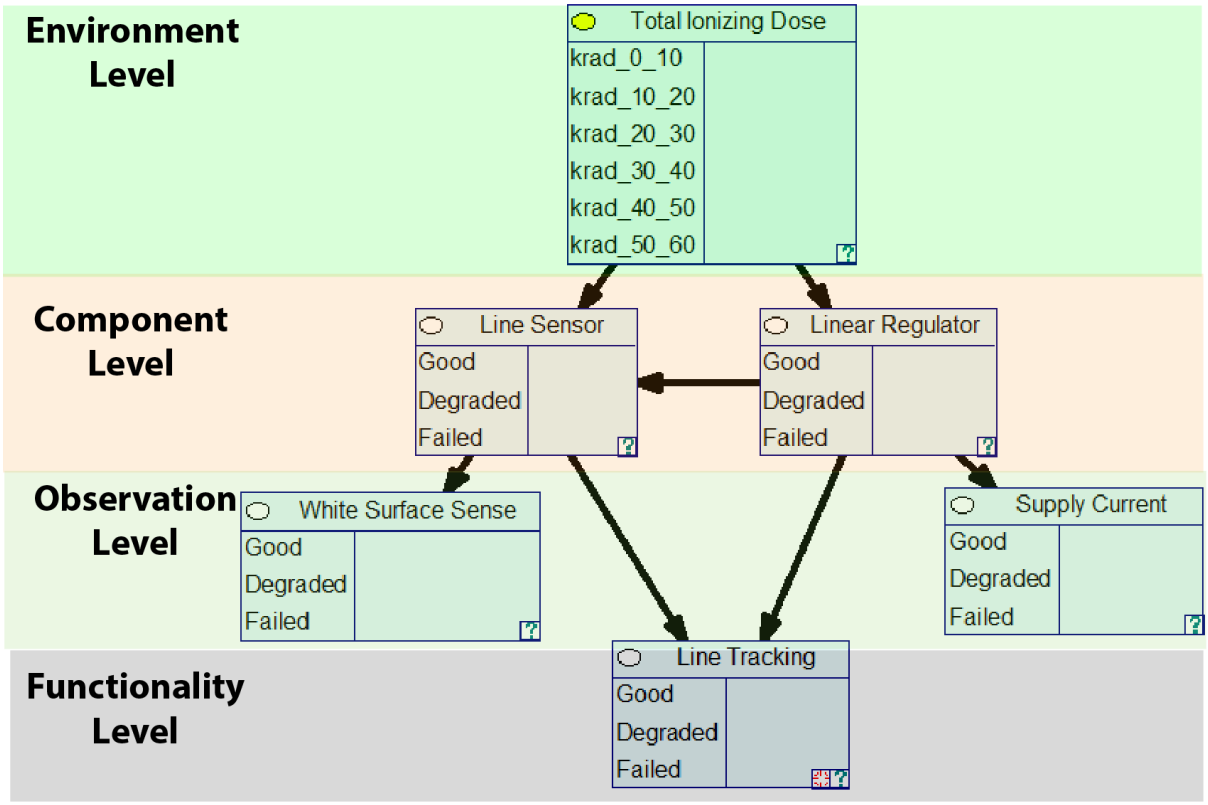

Figure 28. Simplified Bayesian network for line-follower robot. Image shows the nodes, their interconnections, and the possible discrete states, separated into highlighted layers.

Each node is independent of all the nodes to which it is not connected (all nodes that are not its direct parents or children), so changing the network structure is relatively straightforward, only requiring changes to the values of the connected nodes. If nodes are included that turn out to have a constant probability distribution versus its parent nodes it can be removed, or if a node has a 0% value for a discretization category, that category can be removed for that distribution.

Many Bayesian network applications lose correspondence between the network nodes and the physical interpretation of the network. Figure 28 follows a four level structure for radiation degraded systems containing an environment level, a component level, an observation level, and a functional level. This general structure provides a powerful framework for generating data to populate the network for radiation degraded systems, including TID and displacement damage,

where gradual degradation takes place. One advantage of this approach as compared to conventional radiation testing and modeling is that there is no direct edge between the radiation node and the final system parameter, e.g., in Figure 28, the tracking node, which is a high level performance parameter, does not have to be directly characterized as a function of TID.

Populating Bayesian Network with Probabilities

Once the structure of the Bayesian Network is determined, the next step is to populate the network parameters - the prior probabilities of the root-nodes and conditional probabilities for the other nodes. These probabilities are computed based on the data sets captured from experimentation and simulation studies. For a component like the linear regulator, multiple parts can be irradiated and characterized. For a functional node like tracking, while data can be collected through experimentation, additional data can be gathered using a physics-based simulation model and varying the parameters of the sensor and the linear regulator to simulate the tracking performance.

For a discrete Bayesian network, the continuous radiation performance data is discretized based on the specified thresholds into the discrete state of the nodes. This discretized information about the system is then used to compute the prior and conditional probabilities.

The prior probability captures the probability that a node is in a given state. Prior probability for each node-state is determined by dividing the number of occurrences of the specific state, with the sum-total of the occurrences across all the states of the node. Examples are given in the following.

The conditional probabilities represent the probability that the node is in a given state given the specified statuses of the parent nodes. In order to compute the conditional probability, the data-set is restricted to the given set of parent states. The probabilities computed for each node-state with this restricted data set, correspond to the conditional probabilities of the node for the specified

parent(s) state. This is repeated for all parent-state combinations to compute the probability numbers associated with all the cells. These probabilities can be updated as new information becomes available, or can be de-rated or additional weight given to outliers to ensure that worst case outcomes are magnified if desired. The network tool used in this work, GeNIe, accepts discrete probabilities that must sum to 1 for a given node. These probabilities can be determined in a variety of ways depending on the goals of the network (maximum likelihood estimate, worst case estimate, 95% bounds), etc. In fact, each node can on one end of the spectrum undergo a Bayesian treatment such as in [19] to construct these probabilities or on the other end of the spectrum can consist of simply binned experimental data. The full Bayesian treatment allows for incorporation into a hardness assurance process, while the binned data process may provide design insights early in the engineering process.

To provide useful inference, experimental or other sources of information must be available for all the conditions that are desired to be considered. To ensure conservative analysis, a "derating" node could be added to the network, allowing the user to select a desired dose or other parameter derating. Further information about the construction of priors can be found in [19] and [33].

In case of the linear regulator, the voltage data collected are sorted into good/degraded/failed bins based on user specified thresholds. The corresponding parent node (TID) data are also converted into the appropriate discrete states. The procedure described above is used to compute the conditional probabilities.

The same process is followed in the case of other nodes. An example table is included in Fig. 6. It shows the conditional probability table for the Line-Tracking node. It is generated from a combination of experimental data and modeling of the robotic line follower.

Each combination of parent states must have a value, and each column must sum to 1. The values can be determined through a combination of experiment, historical data, expert knowledge, or simply the best information available. The inference performed by the network depends on the quality of the information with which it is populated, so the higher quality of the population information, the more accurate the inference will be. In principle the Bayesian network is limited in the accuracy of its prediction for the target output variables only by the accuracy of the information in the input variables [10].

Line Sensor	Good			Degraded			Failed		
Linear Regulator	G	D	F	G	D	F	G	D	G
Good	0.9	0.2	0	0.2	0.1	0	0	0	0
Degraded	0.1	0.8	0	0.7	0.6	0	0	0	0
Failed	0	0	1	0.1	0.3	1	1	1	1

Figure 29. Conditional probability table showing the values used to populate the line tracking node in Figure 28.

Robotic Line Follower

The following four subsections use the case-study Bayesian network developed in the previous sections, to illustrate applications of Bayesian networks in assessing impact of TID on system health and performance.

Estimating Probability of Success of Key System Functions

The most straightforward application of a Bayesian Network to the radiation effects problem is forward inference – given the TID level, calculate the resulting distributions for all subsequent nodes. When the TID-level is known, the evidence/observation on the TID node can be set to the

specific state (corresponding to the TID-level). Figure 30 presents the results of fixing the TID state to "Under 50 krad(SiO_2)". After setting the evidence, the software tool can execute the Bayesian Network inference algorithm [22] and compute the posterior probability distributions for the other nodes, the result of performing the inference algorithm. The results of the inference should be interpreted as: given the evidence that the TID is between 40 and 50 krad(SiO_2), there is 17% likelihood that tracking functionality is good, 50% likelihood that it is degraded, and 33% likelihood it is failed. The distribution is determined by the uncertainty and part-to-part variability in the network, which is captured in the conditional probability tables of the network nodes.

Inference Using Observable Parameters

A second application of Bayesian networks is inferring the probability distribution of nodes of interest, based on the evidences observed on the nodes that are easy to measure – the observable node level. Figure 31 shows the posterior probability distribution after setting evidence in the "White Surface Sense" node to "Degraded" and in the "Supply Current" node to "Degraded" and running the inference algorithm.

The output shows a significantly narrower posterior distribution for the "TID" and "Line Tracking" nodes based on the evidences compared to performing the inference with the observable parameters unknown. In this case, the Bayesian network is able to use the observed state of the easily measurable nodes to estimate the possible range of TID values. In the field in a post-nuclear disaster environment such as the Fukushima scenario, the current TID level may be difficult to determine due to annealing and uncertainty in the knowledge of the environment.

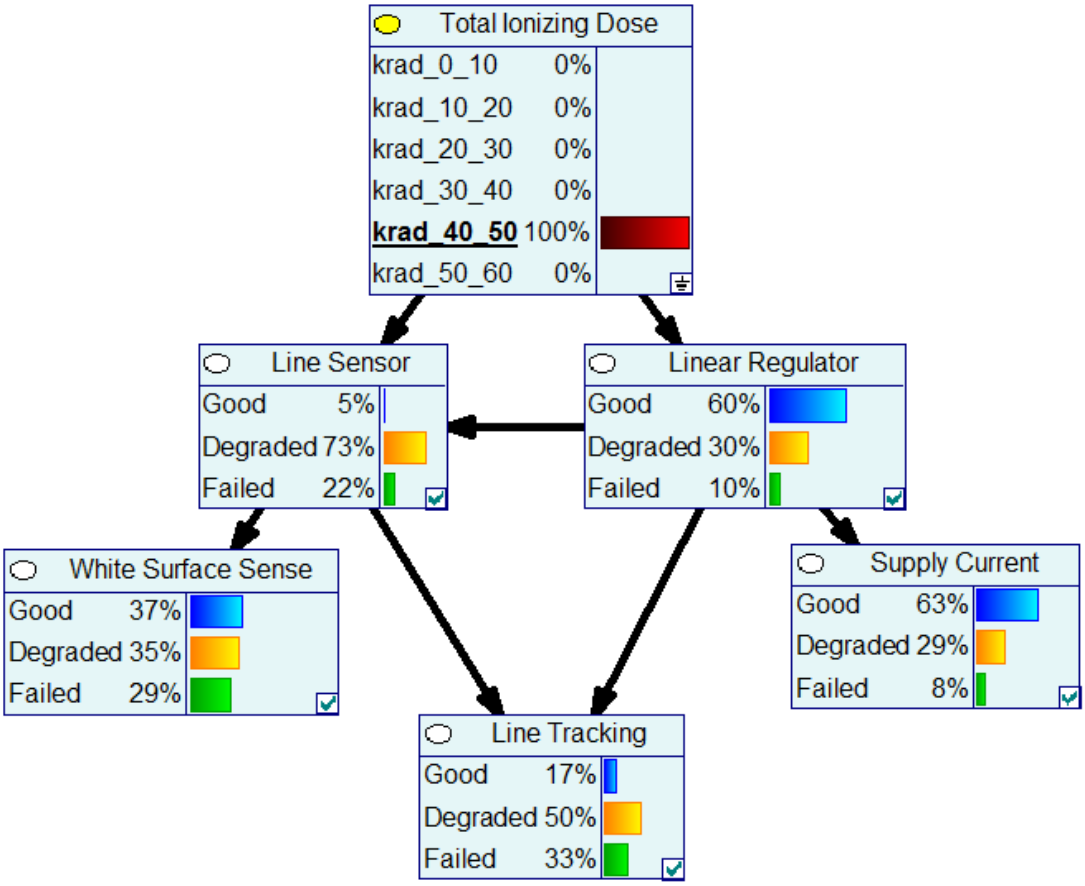

Figure 30. Example of forward inference. The evidence of the TID node is set to the "Under 50 krad(SiO2)" state. The posterior distribution of the remaining nodes is calculated using Bayesian inference update.

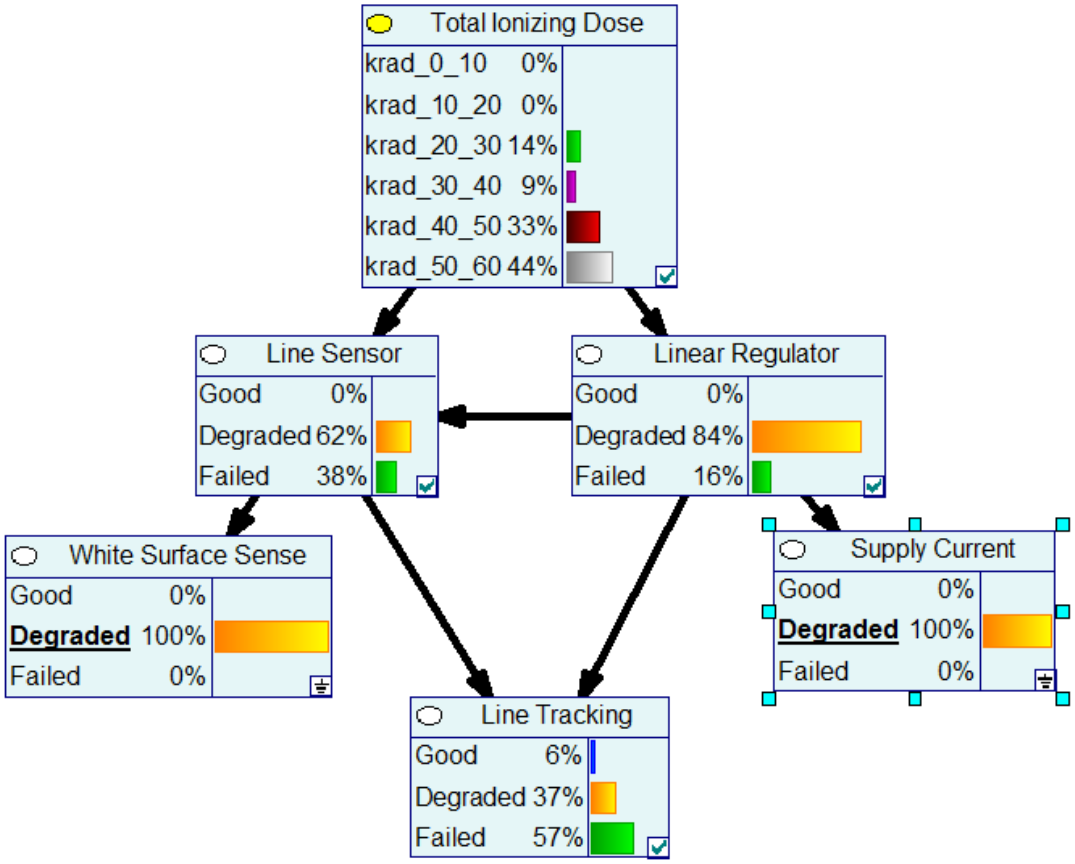

Figure 31. Example of performing inference using observable parameters. Setting the White Surface Sense and Supply Current observational nodes to Degraded allows for the inference of the status of Line Tracking and Total Ionizing Dose distributions.

Sensitivity Analysis for Design Enhancement

Bayesian networks, once designed and populated, can evaluate the sensitivity of a target node to the status of the other nodes in the network. Figure 32 shows the sensitivity of the Tracking node to the status of the other nodes in the network, with red indicating highly sensitive and grey indicating not sensitive. This approach attempts to answer the question of which component to harden/shield/replace in order to gain a more robust TID degraded performance for the Tracking functionality. In this example, the tracking node is most sensitive to the state of the TID node and slightly more sensitive to the state of the Linear Regulator than the Line Sensor. If there was a significant difference between the Tracking node's sensitivity to the Line Sensor versus the Linear

Regulator, it would provide a clear target for further investigation to see whether that node could be hardened. This type of insight may be performed by a human for small networks, but for larger networks where dependencies are less clear, the Bayesian approach becomes increasingly valuable.

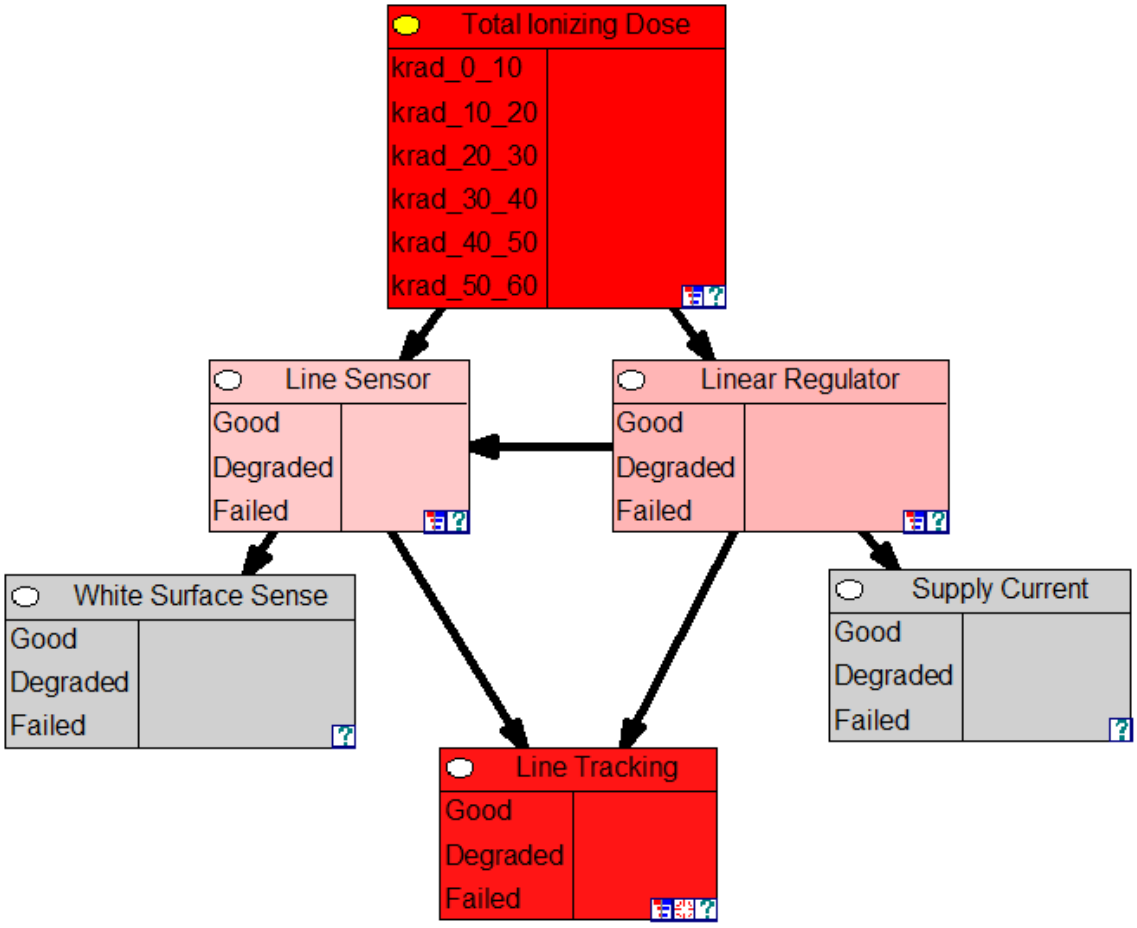

Figure 32. Automated sensitivity analysis for the Line Tracking node performed by the Bayesian network tool. Colors are coded by tracking node sensitivity. The Line Tracking performance is most influenced by Total Ionizing Dose, which is the logical outcome for this system topology. Notably, the observational nodes have the least direct impact on the Line Tracking distribution.

Estimating TID Levels Including Different TID Levels for Different Components

Small changes to the Bayesian network structure can provide insight into otherwise difficult-to-interpret situations. The network can and should adapt with the designer's assumptions about the characteristics of the system being modeled. As demonstrated in Figure 33, the impact of different components in a system receiving different TID levels can be determined by adding more TID nodes to the network. The line sensor TID node is set to 30-40 krad(SiO_2), and the second TID node is set to 40-50 krad(SiO2). The rest of the Bayesian network remains unchanged, allowing the same types of inference to be performed as in the previous examples. For example, the probability of "good" line tracking for the TID settings shown in Figure 33 is only 8% compared to 17% for the TID settings in Figure 29. By including a separate TID node, additional types of questions can be answered without any new experiments or changes to the Bayesian network. In this case, the Line Sensor may receive a different dose because of its position on the robot, shielding, or because the two components may anneal at different rates. The same types of analysis performed in the previous three subsections can be performed on this new network given the new assumption.

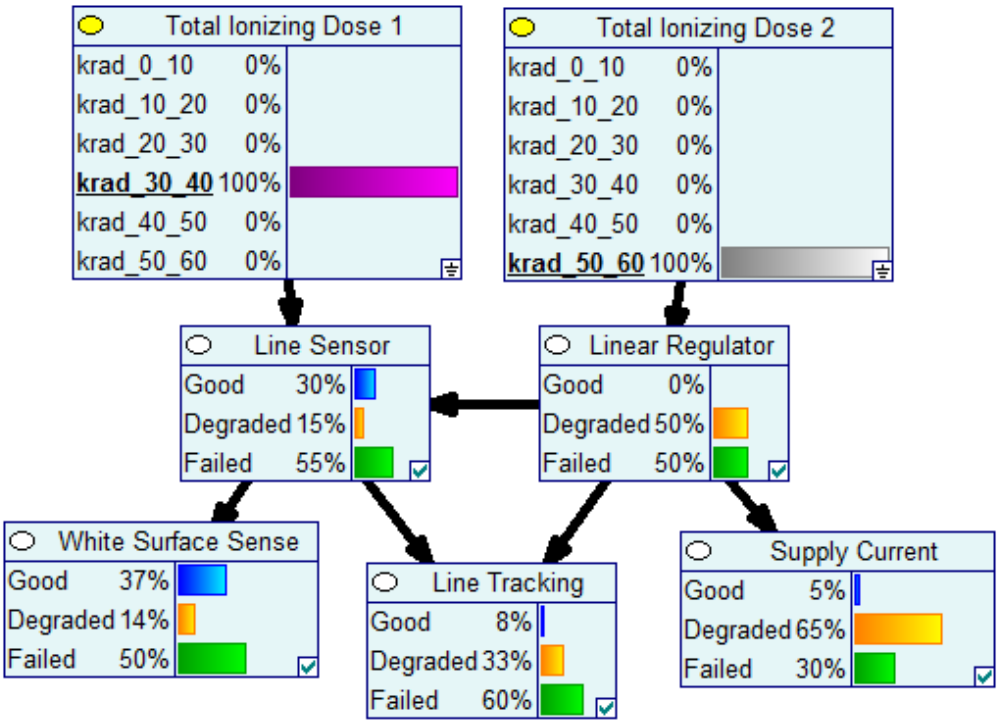

Figure 33. Modified network including a separate TID node for each of the components, accounting for part replacement, different shielding levels, or other factors that could produce different dose levels for different components.

Discussion

Discrete Bayesian networks provide a quantitative method for assessing the response of performance parameters of a system to TID. This awareness can be applied during the design phase of system development to gain insight into the TID hardness impact of different design options or during field operation to determine the likelihood that particular functions will be available given different possible operating scenarios and to interpret limited field measurements. The Bayesian network enables informed decision making about system capabilities, for example, when evaluating whether the system in its current radiation state can accomplish a specific mission, or whether a less demanding mission should be chosen in light of degraded functionality.

The quantitative and probabilistic nature of the network allows incorporation of many aspects of TID degradation. Part-to-part variability is included automatically, with the multiple components tested falling on different sides of a discretization threshold in the probability table based on their variation. For example, if 10 parts are tested at TID level 5, with eight being good and two being degraded, at TID level 5 the network incorporates this prior knowledge of the variability in component response in the discrete distributions.

At a component level each component may be within specification even though partially degraded, which would normally imply the system performance is within specification, but the Bayesian network captures the interaction between components that may lead to a systemic failure even though the individual components are nominally within specification. Finally, the posterior probability distribution on the target functionality nodes in the Bayesian network yields an interpretable result for non-radiation effects experts.

Comparison with Hardness Assurance Methods

Identifying worst case responses is an important part of the hardness assurance process, and the Bayesian network approach can assist in this process. Experimental details and environmental concerns such as low dose rate response or other testing considerations such as temperature detailed in [61] can be included in the network by adding nodes to represent those conditions. This allows the network to accept priors and likelihood data for a variety of useful circumstances simultaneously, letting the user operate the network for a variety of potential conditions. The discrete Bayesian network will calculate all combinations, which can assist with Worst Case Circuit Analysis (WCCA) by identifying possible combinations of conditions that lead to degraded or failed operation. If a WCCA is already completed before construction of the Bayesian network, it can inform which nodes to include in the network.

Bayesian networks can work in conjunction with Failure Modes Effects and Criticality Analysis (FMECA). FMECA provides a systematic procedure for analyzing the failure and reliability of systems and their underlying components and subsystems. In FMECA, the likelihood of occurrence of each failure mode is documented in terms of probability or component reliability estimates. The Bayesian network framework in this work focuses on component and system degradation associated with radiation exposure. The cause-effect relationship identified from FMECA can inform the construction of the network. Additional information is required to populate the conditional probability tables that capture the effect of the partial failures/ degradation as well as the impact of each parent node on its children to complete the Bayesian network.

Working with Hybrid Data Sets

Information about the TID response of a system can come in a variety of forms – experiments, similar parts data, historical data from different lots, simulation, or expert knowledge. For the best prediction accuracy, the network should be populated with repeated experiments on lot controlled components for the target system. However, during the design process this can be prohibitively cost- or time-intensive. Instead the mix of data types available can be used, trading expense for accuracy in a quantitative matter. Non-Bayesian hardness assurance practices do not have a quantitative method for incorporating such diverse data sets. Of course, the more accurately the data represent the actual system, the better the prediction will be.

The fundamental idea that any available component or system characterization information improves the health status prediction can leverage the wealth of information available to designers in TID databases. For example, if there is no TID degradation information about a specific desired component, use the data from a similar component and if necessary supplement the data with test data on the desired component when the data are available. This approach can be complemented

with a robust set of experimental tests to perform hardness assurance on the final design if appropriate. The requirements for constructing the probabilities and priors can be determined by an organization's risk tolerance, but in general the sources of information used to inform the network must be documented, because the quality of the information directly impacts the quality of the inference performed by the network.

Design or Model Modification

The Bayesian network independence properties make modification or addition of nodes straightforward. Each node is only affected by its parent and children nodes. If a component is replaced with a different component, only the replaced node and its direct children nodes need to have their probability tables updated. During the design process component changes may occur regularly, and the Bayesian network can use the most recent information available to make a quantitative prediction or perform a sensitivity analysis.

Structuring the network in four layers allows for easier network modification. The component layer is the only layer that interacts with the environment and may require TID testing. The rest of the layers can be populated with simulation data from physics-based circuit or system level simulations, using the component-level-degradation data in a standard electrical/mechanical modeling and simulation flow.

Sensitivity to Discretization Process

Bayesian networks can be seen as an upgrade to binary modeling of component failure (components are either within specification or not) because of the ability to model and propagate many possible node states in a probabilistic and quantitative way. For TID-degraded systems, the multiple possible node states reflect the physical reality that there are states between ideal

functionality and failure. In the models presented in this paper three states were chosen (Good, Degraded, Failed), but an arbitrarily large set of states could be chosen to reflect the nature of the system being modeled. As the number of discretization states is increased, additional experimental or simulation information is needed to populate the enlarged probability tables. This can result in a significant increase in the size of the probability tables used to populate the network. A limitation of the discrete Bayesian network technique is that the predictions are sensitive to the discretization process. A part failing at 49 krad(SiO_2) versus 51 krad(SiO_2) can be placed in different bins according to which discretization scheme is used, impacting the inference artificially. Artifacts like this can become problematic if small sample sizes are used to populate the probability tables. Continuous probability distribution Bayesian networks are available [62], which eliminate problems arising from discretization, but which introduce sampling and convergence complications.

Conclusions

Bayesian networks can be applied to the TID reliability arena through a systematic approach of analyzing the system for functional dependencies and designing a structured Bayesian network. The advantage of a Bayesian network is that it extends the benefits of using Bayesian statistics for component level reliability to systems where simultaneous component degradations can interact, leading to impacts on system level performance that might not be predicted from examining component radiation sensitivities alone. The Bayesian network helps give quantitative insight into the state and sensitivities in the system, using inference to answer questions about:

- System function availability at certain dose levels
- Interpretation of system health measurements

- Sensitivity analysis
- Estimation of current TID the system has received.

The Bayesian network approach can work with a variety of data sources, such as simulations, radiation hardness data on similar parts, etc. This allows quantitative estimation of radiation hardness even before complete radiation testing is performed on a new electronic system. Of course the quality of the estimation improves when more accurate models are used for each node in the network. Bayesian networks are flexible and can be adapted and reused efficiently due to the independence of nodes that are not directly connected by an edge, which means a model for a node in the network can be updated independently of the not directly connected other nodes in the network. The Bayesian network can provide insight into radiation impacts starting with design decisions and continuing through field deployment.

The largest drawback to this approach is the significant amount of experimental work required to populate the nodes in the network. Importantly, the amount of experimental work required increases rapidly with the number of dependencies a node has and the number of different states a node can occupy. This motivated exploring the continuous Bayesian statistics approach described in the following chapter. Methods that can scale more efficiently to larger systems and have more flexibility in representing reality can have several advantages, but the data-driven simplicity of the discrete approach should not be overlooked.

CHAPTER VI

CONTINUOUS BAYESIAN MODELING OF TID DEGRADATION

Z. J. Diggins, N. Mahadevan, E. B. Pitt, D. Herbison, R. M. Hood, G. Karsai, B. D. Sierawski, E. J. Barth, R. A. Reed, R. D. Schrimpf, and others, "Bayesian Inference Modeling of Total Ionizing Dose Effects on System Performance," *IEEE Transactions on Nuclear Science*, vol. 62, no. 6, pp. 2517–2524, 2015.

Introduction

Hardness assurance crosses many levels of science and engineering, ranging from physical mechanisms of device degradation to systems engineering concepts. Modeling radiation effects across multiple levels of abstraction is challenging, especially when uncertainty or variance is involved. Although assurance means eliminating uncertainty where possible, uncertainty is an irreducible property of any design when variability is present. Radiation exposure compounds the uncertainty, adding uncertainty about the radiation environment and the part-to-part variability of radiation response to the inevitable distributions in electrical performance of manufactured components. This work embraces the uncertainty introduced throughout the different levels in a system hierarchy, capturing the probabilistic information in the form of a Bayesian network. Individual components will go out of specification at different dose levels due to component manufacturing variability, leading to unacceptable system performance at different dose levels due

to component interactions. The contribution of this paper is a modeling flow that enables quantified estimates of system hardness from component measurements, enabling risk assessments. Case study results show that the system can operate for significantly longer duration (2x) with less than 1% chance of failure after components have degraded out of specification, demonstrating an opportunity to relax component-level hardness requirements

This work presents a modeling framework that captures the details of a system and the dose environment in the form of a Bayesian network and mathematical models of component performance (theory summarized in the appendix). Widely available software tools are used to perform inference on the Bayesian network in order to estimate distributions for parameters of interest. These distributions are then used to parameterize deterministic simulation models that capture additional details of the system, producing an estimate of the system response given the available component data. A case study of a small robot is used to demonstrate the approach; however, the method can also be applied to other types of systems such as spacecraft systems. The results establish the method's ability to predict the system response in a probabilistic manner, allowing for estimation of the probability of mission success and for the sensitivity analysis of target parameters to identify design improvements. Fundamentally this work presents a path to pursuing hardness assurance through uncertainty awareness.

Background

The effects of total-ionizing dose (TID) on electronic components have been studied extensively [61][63][9], but TID can still presents a testing and hardness assurance challenge due to large derating factors necessary to compensate for uncertainty and component variability. Due to variability from many sources (manufacturing process, device bias in circuit, etc.), the rate of the

degradation of different devices will have a degree of variability. This variability is addressed in test standards such as MIL-STD-814 through derating and categorizing. Methods such as Part Categorization Criterion (PCC) have been developed to perform a one-sided tail statistical test [11]. Worst-Case-Circuit-Analysis (WCCA) is one method for addressing this variability. By identifying the worst case in terms of radiation or other environmental reliability concerns (e.g. temperature) and ensuring the system can operate successfully under such conditions, the designer has confidence that the parts used will be reliable.

Designers who are able to accept a certain risk level with the goal of relaxing requirements may not be interested in only the worst-case response, but also the probability of successful, degraded, or failed operation. Previous work has modeled this variability using Bayesian statistics [19][18], enabling a flexible and powerful analysis of individual components. The benefits of expanding such statistical techniques to address system behavior are an improved understanding of the risk involved and the identification of potential areas of design improvement when using components that are not radiation hardened or that will be operating near the design margin, such as low cost commercial-off-the-shelf (COTS) components.

Discrete Bayesian networks have been used for addressing component and system quality for TID degraded systems [64]. These discrete Bayesian networks require categorizing or "binning" the data according to chosen thresholds, which is effective in many situations where clear discretization thresholds can be identified. Several benefits can be gained by extending this discrete statistical analysis through the system level without any thresholding, using continuous probability distributions. Instead of breaking up the chain of cascading effects using thresholds, this works models the combination of effects using continuous Bayesian networks, supplemented with multi-domain model tools where appropriate. A continuous treatment of the data is closer to

the actual physics of TID degradation, allowing for a more natural treatment of the degradation (e.g., TID damage does not occur in discrete steps). Interactions between degrading components due to TID can be taken into account in such an approach. Significant derating levels are not required to hedge against unanticipated interactions, potentially leading to parts with desired performance or cost attributes being included in the system. A low variance, low margin system can be more desirable than a high variance, high margin system. The entire part variability space can be simulated, eliminating the need to identify worst-case circuit analysis since no thresholds need to be identified. Understanding component interactions has the potential to lead to low cost, effective design changes to meet target requirements.

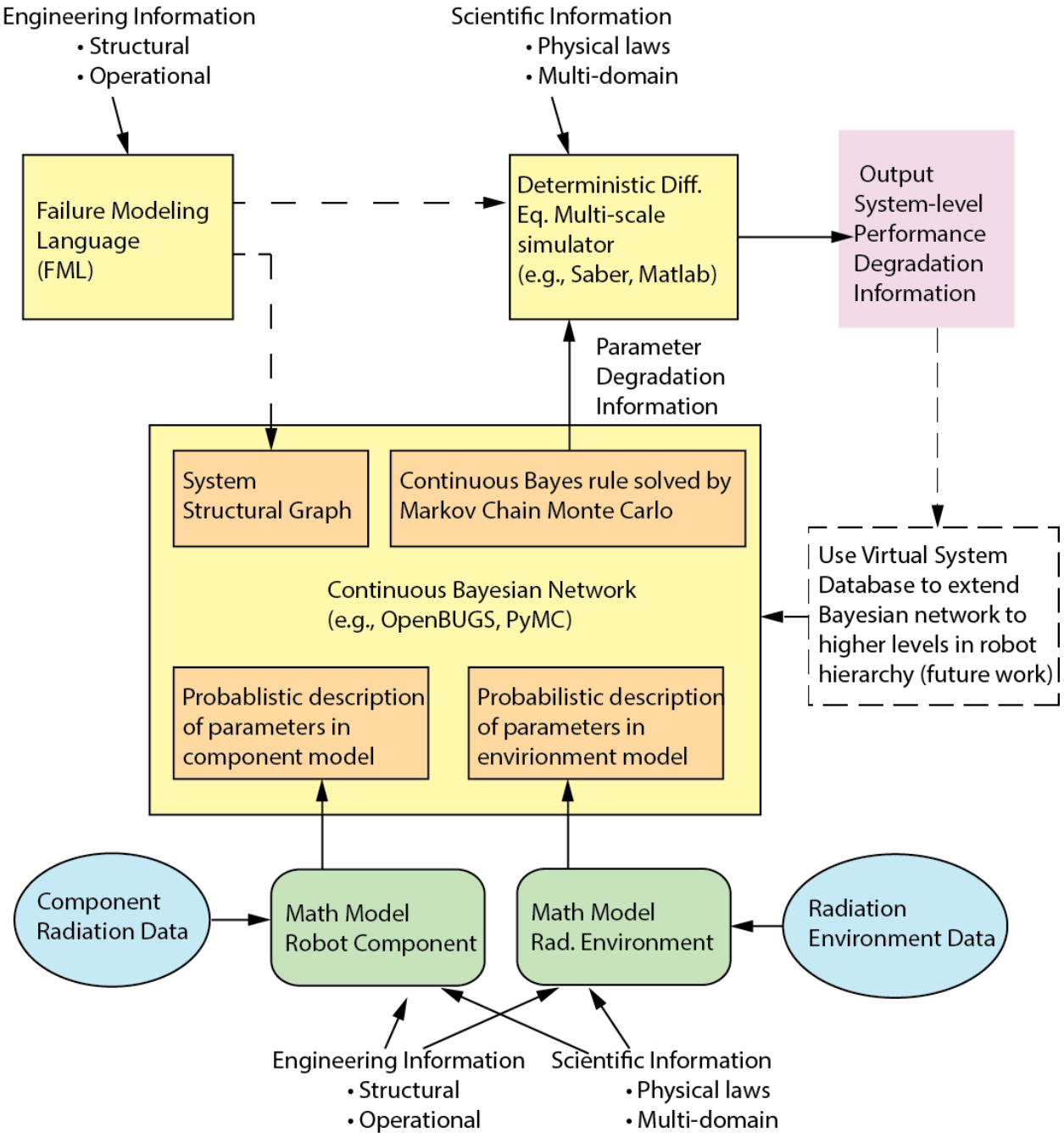

Figure 34. Overview of continuous Bayesian network modeling process. The analysis is a multi-step process, and incorporates domain specific information in math models, statistical information in the Bayesian network, and system model information in the deterministic simulator. Additionally, failure modeling language can be used to inform the network architecture.

Using Continuous Parameter Bayesian Networks for TID Degradation Modeling

This work presents a continuous Bayesian network inference process for TID degraded systems, diagramed in Figure 34. The processes are explained in greater detail in the following case study section. The overall radiation degradation modeling approach involves assimilating data and information to build different kinds of models related to the environment, component, and system. In this context "component" can refer to either a low level individual device like a transistor or a sub-assembly like a sensor. "System" in this context refers to the high-level set of related components that are designed to perform a specific set of tasks. The types of information include [33]:

1. "Engineering information," about the structure, component interconnection, and capabilities of the system.

2. "Scientific information" about the physical principles on which the various components in the system operate, such as first-principles equations from physical laws.

3. Data and measurements, both of electro-mechanical performance and degradation of individual component performance. These data have to be translated into usable information by means of mathematical models of the component operation. The data impact the mathematical models by changing the parameters of the model.

The modeling procedure uses different kinds of simulation and estimation models. Each simulator enables a different kind of inference about the system state and performance. Using the different kinds of simulation together enables system-level inference about robot performance capabilities in the presence of radiation-induced degradation.

The structure of the continuous Bayesian net is informed by the Fault Modeling Language (FML), from [64], which captures available engineering knowledge and dependency information. Physics models or empirical models are then developed to capture how the components degrade with increasing TID. All variables and effects are described by continuous probability distributions to refine the network structure and populate likelihood relationships. To capture the importance of experimental measurements of individual component degradation and the radiation environment, a Bayesian network inference engine is used that computes the posterior probability distributions based on the prior knowledge of the distribution of component parameters and radiation environment models. The Bayesian network allows assimilating evidence information in the form of observed (measured) variables about the effect of degradation to infer the impact of degradation on other unobservable variables. The Bayesian model also performs the function of quantifying the degree of uncertainty about the radiation environment and the radiation degradation. To capture information about the robot components, their interactions, and the overall system behavior, a deterministic simulation model is developed and executed using the appropriate deterministic simulation tool (in this work Synopsys Saber was used). The deterministic simulation enables quantitative estimates of the effects of component degradation on system performance.

Case Study: Line Tracking Robot

The same line tracking robot system as the previous chapter is used to exercise the continuous Bayesian framework.

System Details

The example system is a path-following robot [65][66] [59] with a sensor array for the path consisting of six optical transmitter-receiver pairs that are aligned in a row as shown Figure 35.

Only the sensor was exposed to TID, with the goal of modeling the impact of the variability in the six sensors on the system behavior, motivated by previous work showing that sensors and their subcomponents have a large impact on system performance [58], [59]. This robot was chosen as a test vehicle for modeling system radiation response because the robot design is open source, is inexpensive, almost every variable in the system is accessible for measurement, and irradiated parts can be replaced easily. The robot detects the position of a black line typically 2 to 3 cm wide on a white surface by analyzing the outputs of the individual sensors. Each sensor only produces a signal over a fraction of the length of the line sensor array. The individual sensor signals are combined to provide an indicator of the robot's position with respect to the line.

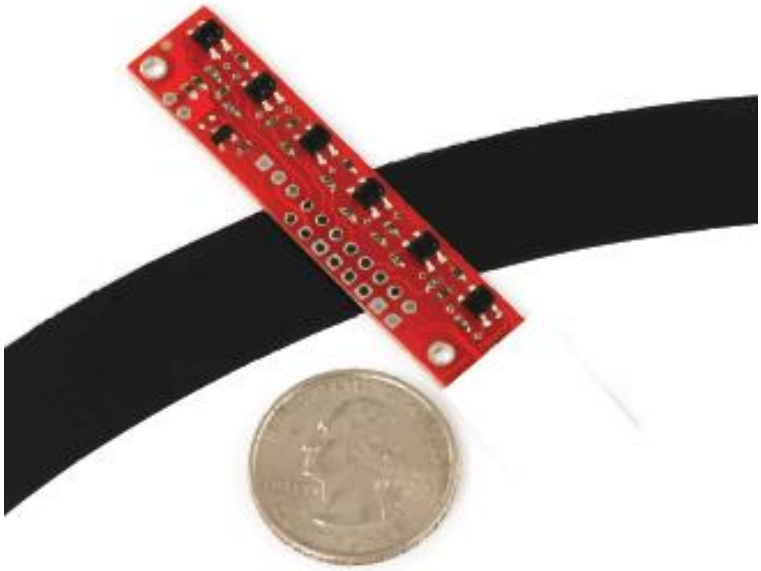

Figure 35. Image of line sensor over a track [66]. In operation the line sensor faces the track.

The outputs from each of the six sensors are combined in a weighted function to create a single input signal to the robot control system, as shown in Figure 36.

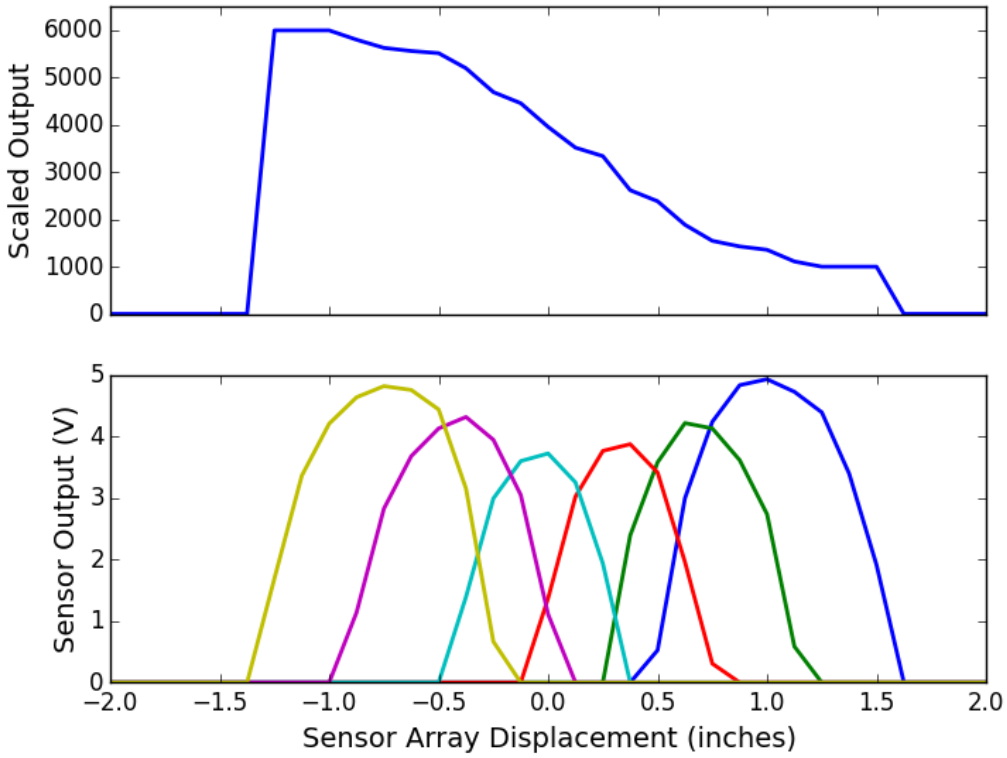

Figure 36. Figure shows individual sensor outputs (bottom) and combined output (top) for sensor board position relative to the center of the line.

The operation and theory of degradation of the line sensor are summarized in Figure 37. Theory of degradation for line sensor optical couplers. The two primary sources of degradation are a decrease in the gain of the phototransistor and the decrease in output of the LED. For TID induced degradation, a decrease in gain of the phototransitor.. Infrared light is emitted by the transmitter, which reflects off the ground. The reflected light modulates the resistance of the phototransistor. When the phototransistor receives light from a reflective surface, the sensor output voltage is pulled towards 0 V. As the sensor receives less light, the output moves towards Vcc. The performance of optocouplers is expected to degrade with TID [66], specifically the light output of the LED will decrease and the gain of the phototransistor will decrease.

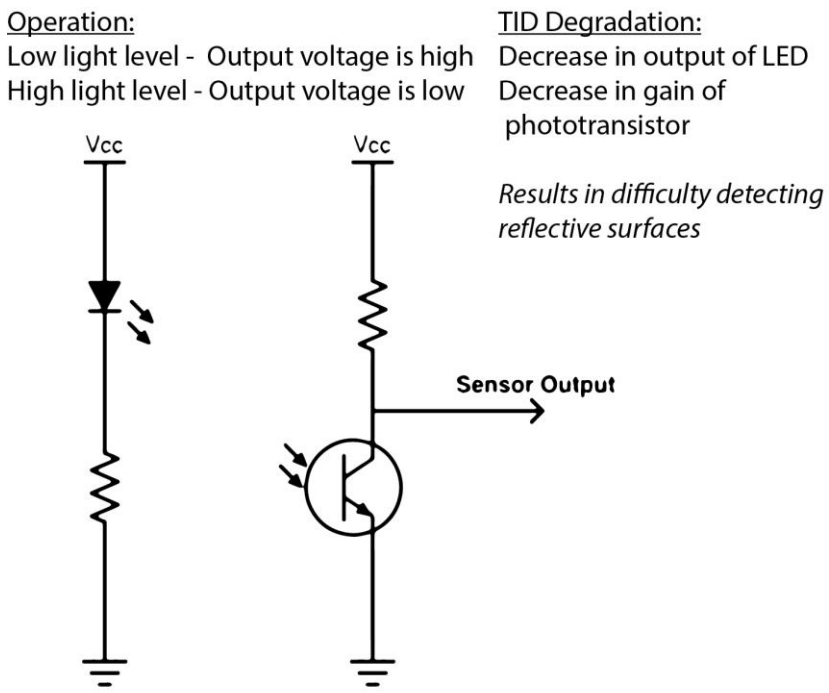

Figure 37. Theory of degradation for line sensor optical couplers. The two primary sources of degradation are a decrease in the gain of the phototransistor and the decrease in output of the LED. For TID induced degradation, a decrease in gain of the phototransitor.

Experimental Details

Four boards containing six reflectance sensors were tested for their TID response using a 660 keV gamma-ray ^{137}Cs source; a total of 60 individual sensor data sets were collected. An example of the change in sensor output for different TID levels is shown in Figure 38. The degradation due to the reduced light output and reduced gain manifests itself as an inability to achieve low voltage levels over highly reflective surfaces.

A testing set for model validation was obtained by taking measurements on sensor assembly boards. The boards were irradiated 6 sensors at a time, placed on the robot, and the tracking

performance was monitored by post-processing a video recording of the robot attempting to follow the track.

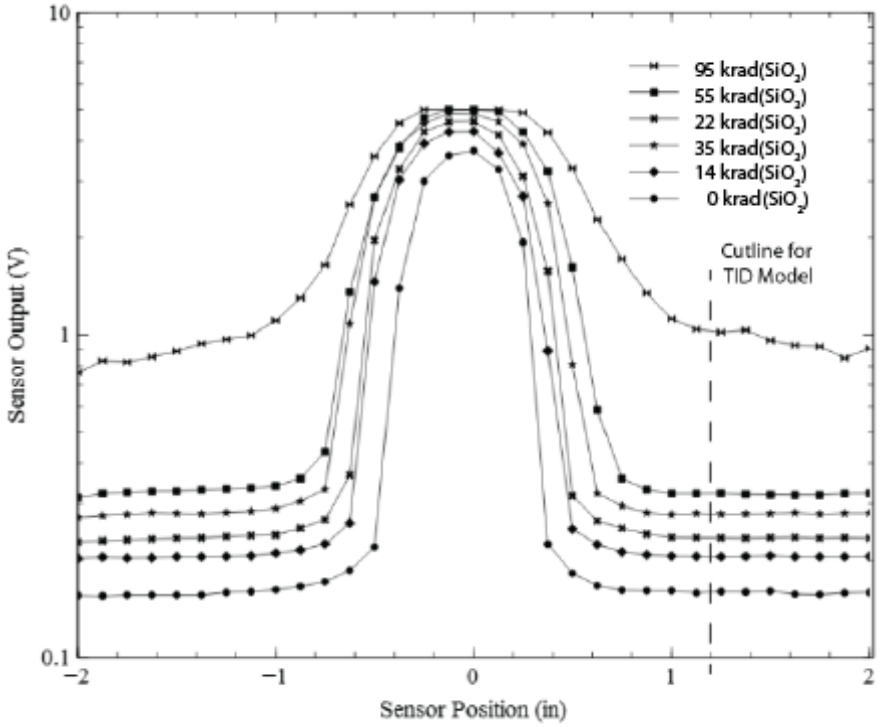

Figure 38. Example degradation of a reflectance sensor. The peak value of the sensor remains relatively constant with dose, but the low range output (over highly reflective surfaces) increases due to a decreased max gain in the phototransistor.

The combination of six degrading sensors resulted in a shift in the combined sensor curve, shown in Figure 39. With increasing TID, the slope of the curve decreased significantly, resulting in the robot eventually not having enough control gain to stay centered on the line. The curve has multiple inflection points, and when the slope of the curve reverses the robot will lose the ability to track the lane because it will have a positive feedback gain.

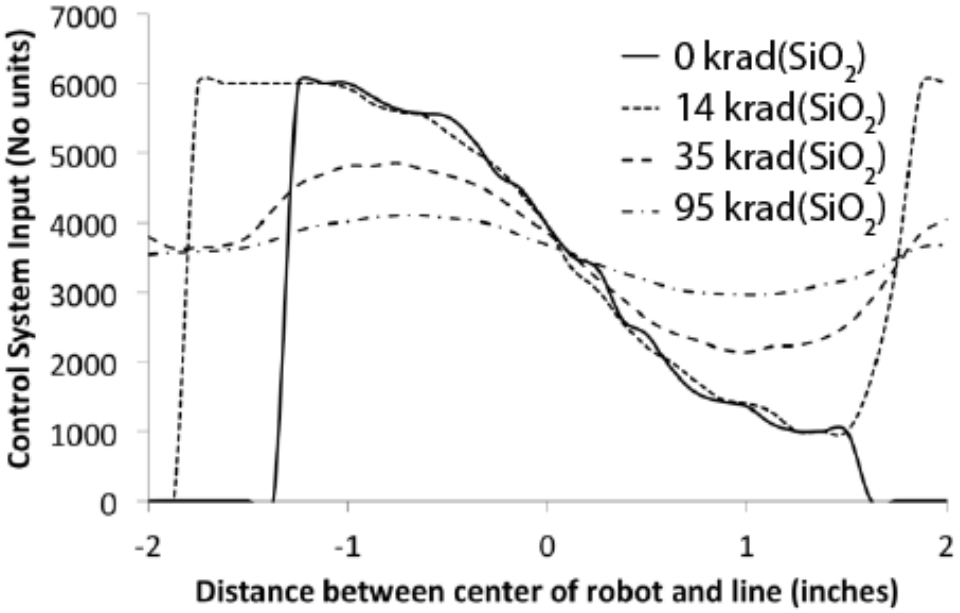

Figure 39. Degradation of combined sensor response for different TID levels. The curve is used as an input to a proportional-integral-derivative controller. As the curve becomes "flatter", the effective control gain decreases.

Bayesian Network Construction

Figure 40 shows the Bayesian network constructed to model the impact of degradation of the line sensor through the system level. Bayesian networks require a parameterized structure to incorporate the data. The increase in the white-space reflectance of individual sensors (the cutline in Figure 38) is the dominant TID effect on the sensors, so it is used to describe the change in the sensor with TID. This increase is modeled as a log-linear function (y-axis on log scale) with two parameters, a slope and an intercept. The log-linear slope intercept model is fundamentally related to the exponential relationship between trapped charge from TID and the performance of the device as described by the law of the junction, and is supported by the experimental data plotted in Figure 45. The variability in the slope and intercept of the log-linear fit is due to part to part variations, which is modeled with normal distributions (for smaller datasets, t-distributions would be appropriate). Each normal distribution is described with two parameters, a mean and a standard

deviation, resulting in four total parameters to descript the TID degradation (slope mean, slope std. dev., intercept mean, and intercept std. dev). The form of the model is expressed mathematically below:

$$\log(\text{whitelevel}_{\text{cutline}}) = \text{slope} * \text{TID} + \text{intercept} \quad (1)$$

$$slope = Normal(mean_{slope}, std.dev._{slope}) \quad (2)$$

$$intercept = Normal(mean_{intercept}, std.dev._{intercept}) \quad (3)$$

Normal distribution priors were used for the four parameters. Non-informative priors were used, setting the prior means to 0, and the prior standard deviation to a large value (100,000) producing effectively a flat prior distribution. If information was available to produce informed priors, it could be included by changing the prior mean and prior std. dev. Guidelines for designing informative priors for a variety of models can be found in [8].

The line sensor TID degradation experimental data are included at this point, populating the network with the experimental data. There are six copies of the component model in the top block, one for each sensor in the system. Modeling each individual sensor best captures the variability in the system, since the source of variability in a six sensor board is due to the variability in the individual sensors.

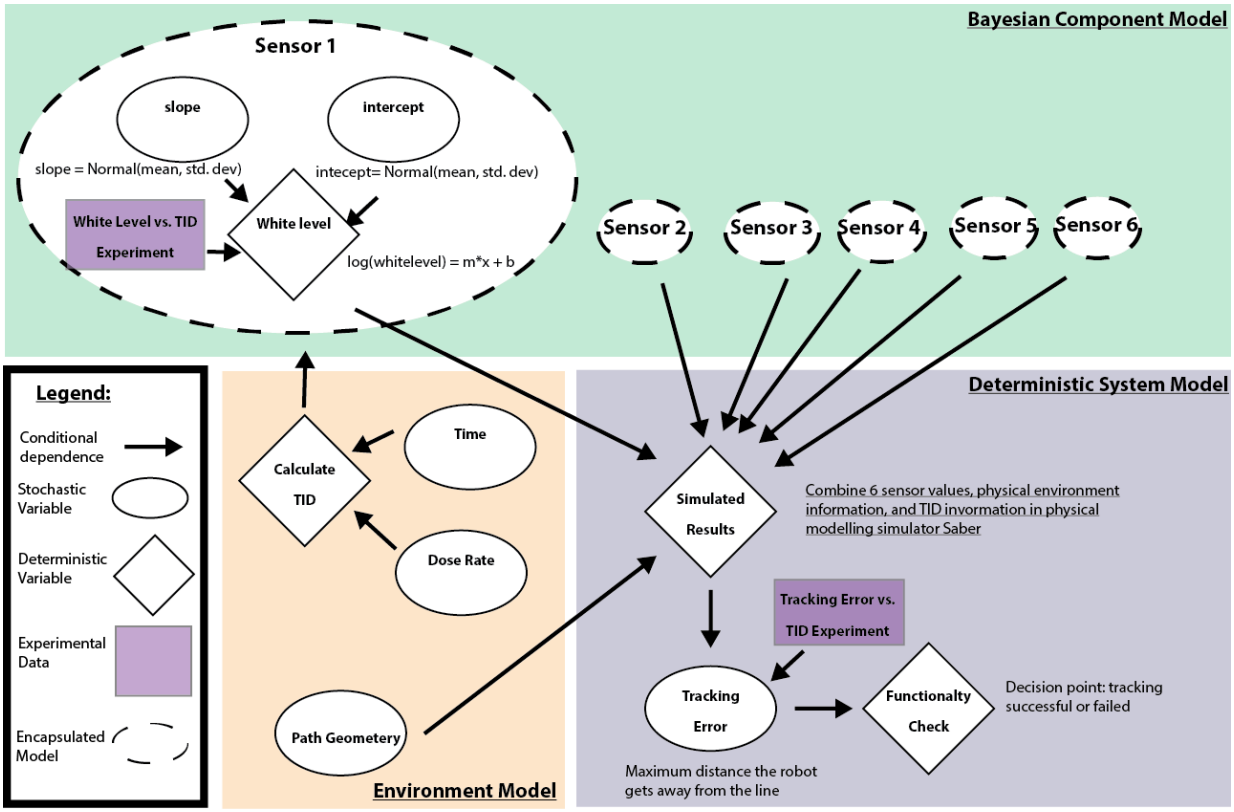

Figure 40. Bayesian network for line-follower robot with degraded sensor. The model is broken up into three sections: Bayesian component model, environment model, and deterministic system model.

The environmental nodes allow for modeling of different dose profiles or paths the robot is commanded to follow. The environment information and sensor component information are passed to the simulator, where a deterministic simulation is performed, producing a distribution of tracking error values. These tracking error values can be compared to system level data. The tracking error can be processed by the functionality check to determine if the robot is tracking the line or losing the line.

Saber Model of Robot System

A hierarchical model of the robot system based on the operation and interaction of the underlying components was constructed using Synopsys Saber, an industry tool capable of multi-domain full

system models. The inputs to the simulation model include environment data pertaining to the desired robot trajectory (a circular track), as well as component degradation data for each of the six sensors in terms of parameters such as sensor white-level bias. The parameter data is output from the Bayesian network. The simulation outputs the system response (the actual path followed by the robot) and metrics to qualify the system response.

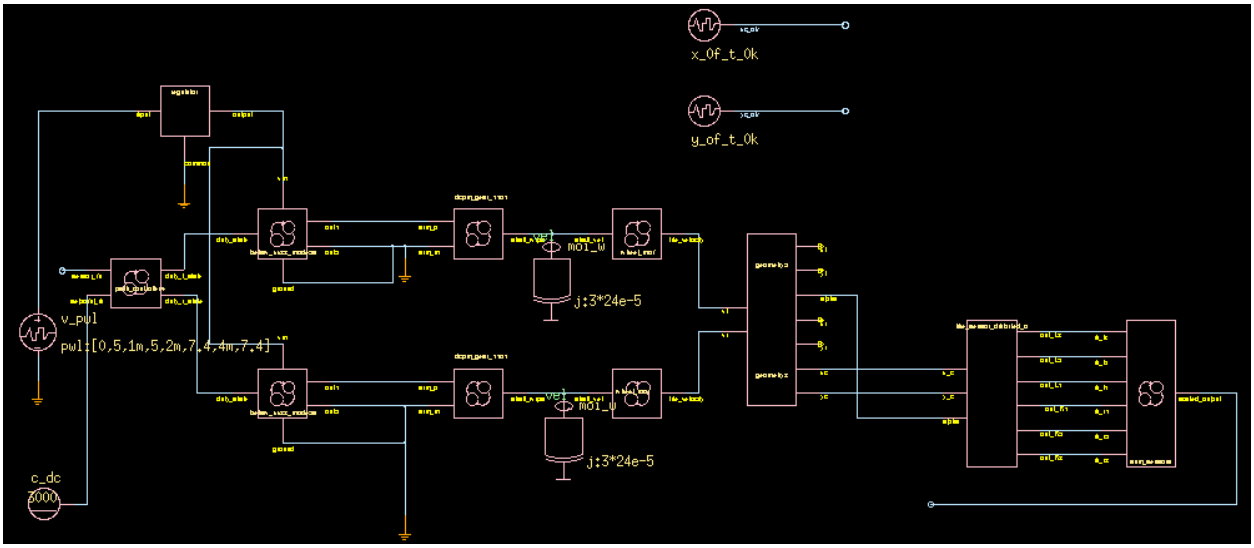

Figure 41. Completed system-level SABER model of Pololu robot.

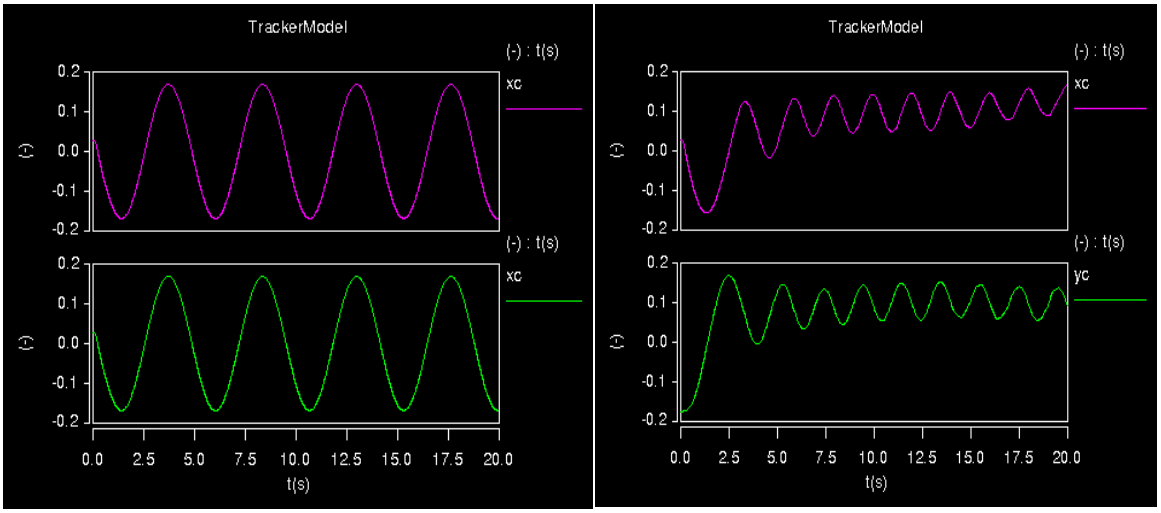

Figure 42. Simulated position components for nominal parameters (left) and degraded parameters (right).

Case Study Results

Verification of Parameter Estimation (Line sensor parameter estimation)

The parameter estimates for the slope and intercept were performed using PyMC. PyMC uses a Markov chain Monte Carlo approach. Diagnostic information of the simulation is contained in Figure 43. In the upper left corner good mixing is observed of the chain samples. In the lower right corner there is initially some autocorrelation, indicating that more thinning or burn-in should be performed. The results follow the shape of the normal distribution, indicating that a sufficient number of samples have been run for the network. In our experiments, the PyMC Bayesian inference engine was able to achieve good mixing, minimal auto-correlation, and in general a normal distribution shape. Mixing indicates the chains have reached a steady state, indicating that the simulation has converged sufficiently. Auto-correlation is a measure of error introduced by subsequent samples. If it is excessively high, the samples can be thinned (ever other sample or more thrown away for example) to reduce autocorrelation. The normal distribution for the histogram indicates that the data is dominating the prior for defining the model inference, and that a sufficient number of simulation samples have been calculated to populate the model sufficiently.

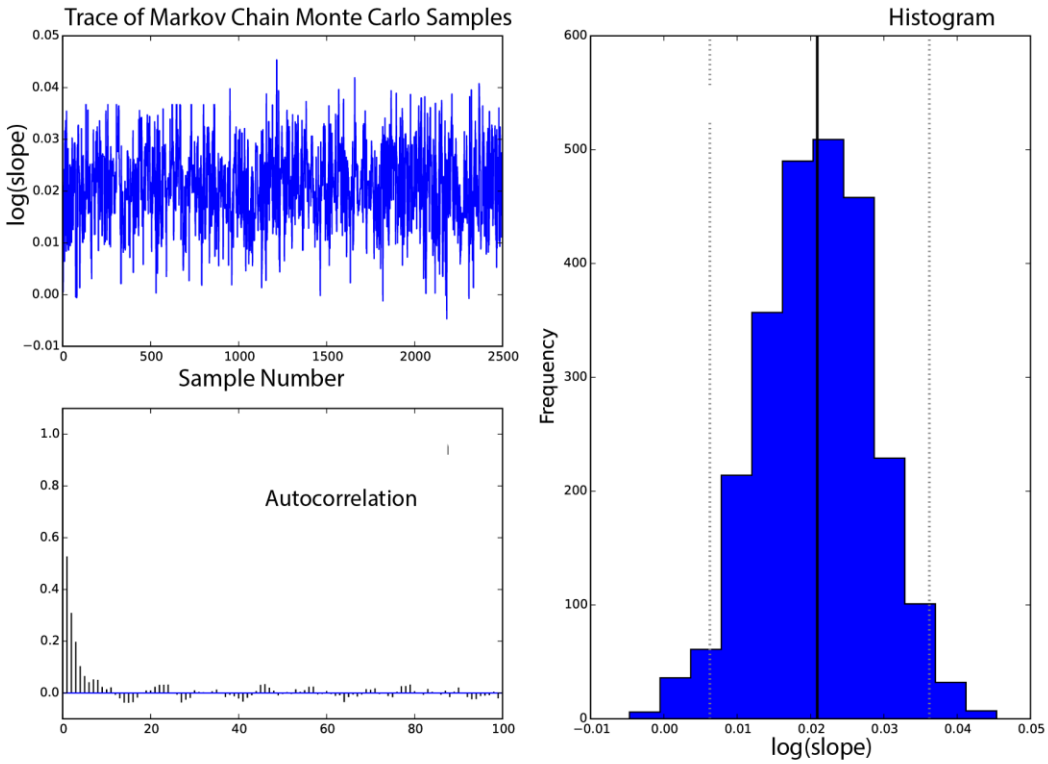

Figure 43. Summary of Bayesian sampling for the slope of one of the sensors.

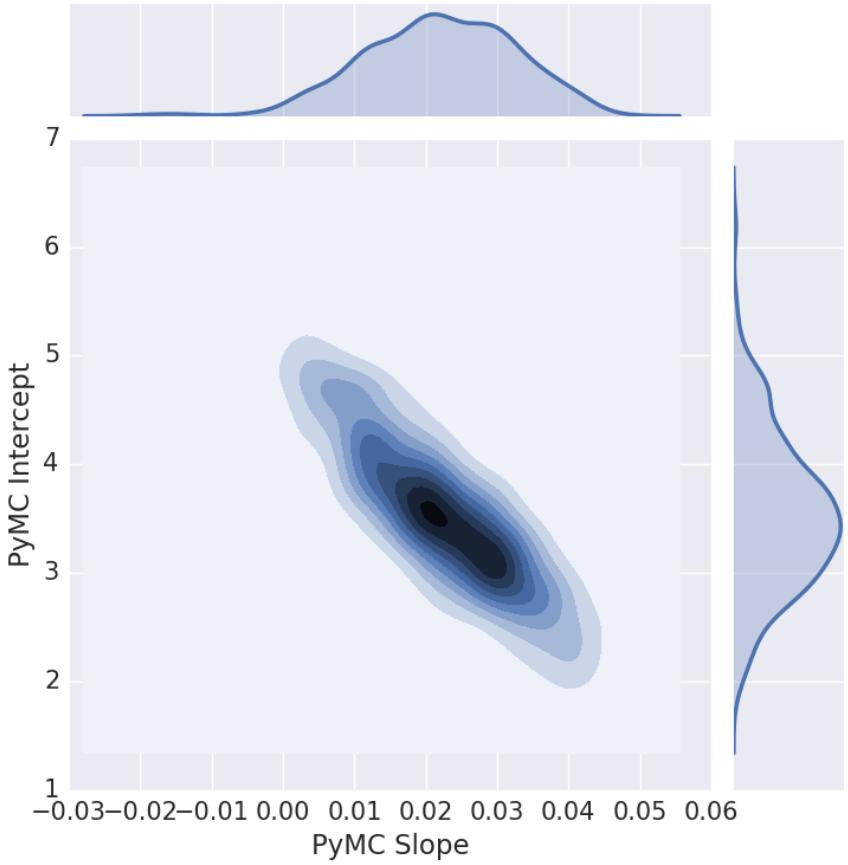

Figure 44. Parameter estimation for the slope and intercept of the PyMC model. The slope and intercept values correspond to the log-linear white-level model parameters. The shape of the surface describes the part-to-part variability in radiation response.

This parameter information can be combined with random TID levels to produce a plot that provides additional information about the quality of the fit. Figure 45 shows the comparison between the Bayesian estimates and the actual component data, with the larger dots and error bars representing the experimental mean and standard deviation. The PyMC simulation results are a combination of the proposed model form (log-linear slope/intercept model) and the experimental data. Each smaller point plotted in the graph corresponds to a sample point generated by the MCMC sampling algorithm used by the PyMC Bayesian inference engine. Each sample point

corresponds to a specific value of the TID (dose), as well as parameters (slope and intercept) for this particular optical sensor in the line sensor assembly.

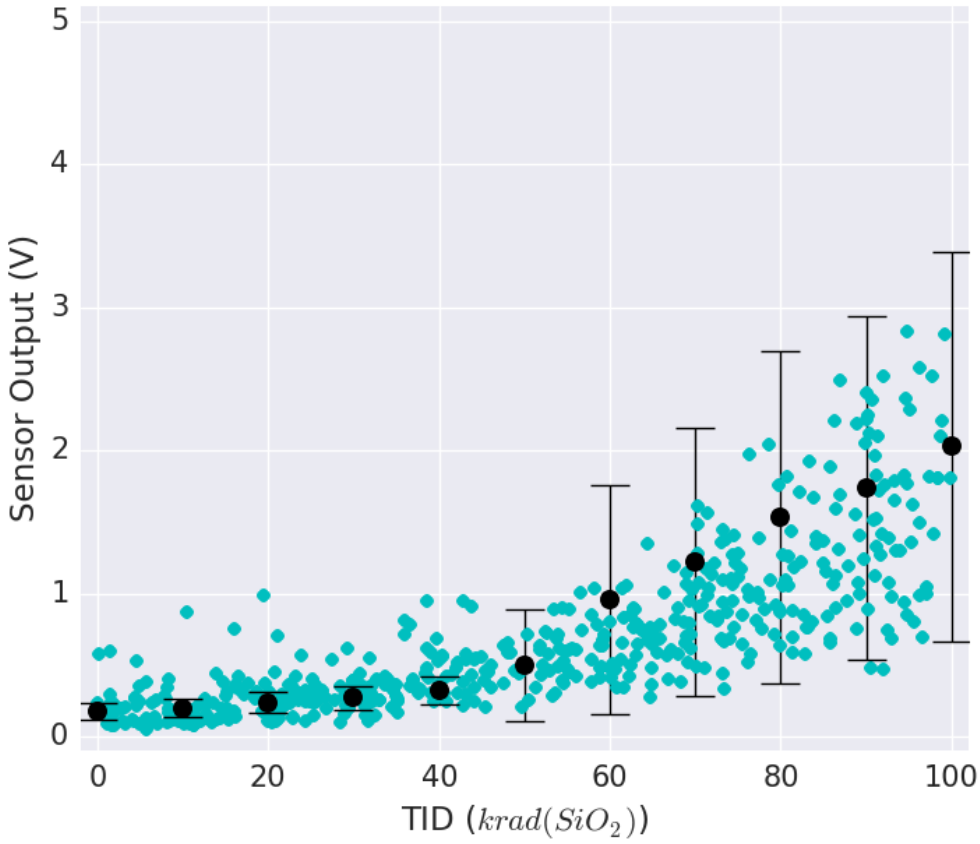

Figure 45. Comparison between Bayesian generated samples and experimental data. Larger dots with error bars represent the data set mean and standard deviation. Smaller dots correspond to simulation results.

Prediction of System Response

The results of the PyMC sampling of the posterior distribution using the specified model and the data from the 60 tested sensors and simulation using the Saber deterministic simulator are shown in Figure 45, comparing the maximum simulated deviation from the line to the ratio of experimental system tests that result in failed tracking. Five systems were tested with radiated line sensors using video to measure line-tracking performance. The Bayesian-deterministic simulations

align well with the experimental video data (a different data set using only line sensor subsystem data similar to Figure 38 was used to generate the PyMC samples). System data was plotted as a cumulative population distribution function because misalignments in the max distance from the line and the video data makes a one-to-one comparison misleading. The transition between a low maximum tracking distance and a high maximum distance has some overlap, indicating a region where tracking is uncertain due to part-to-part variability.

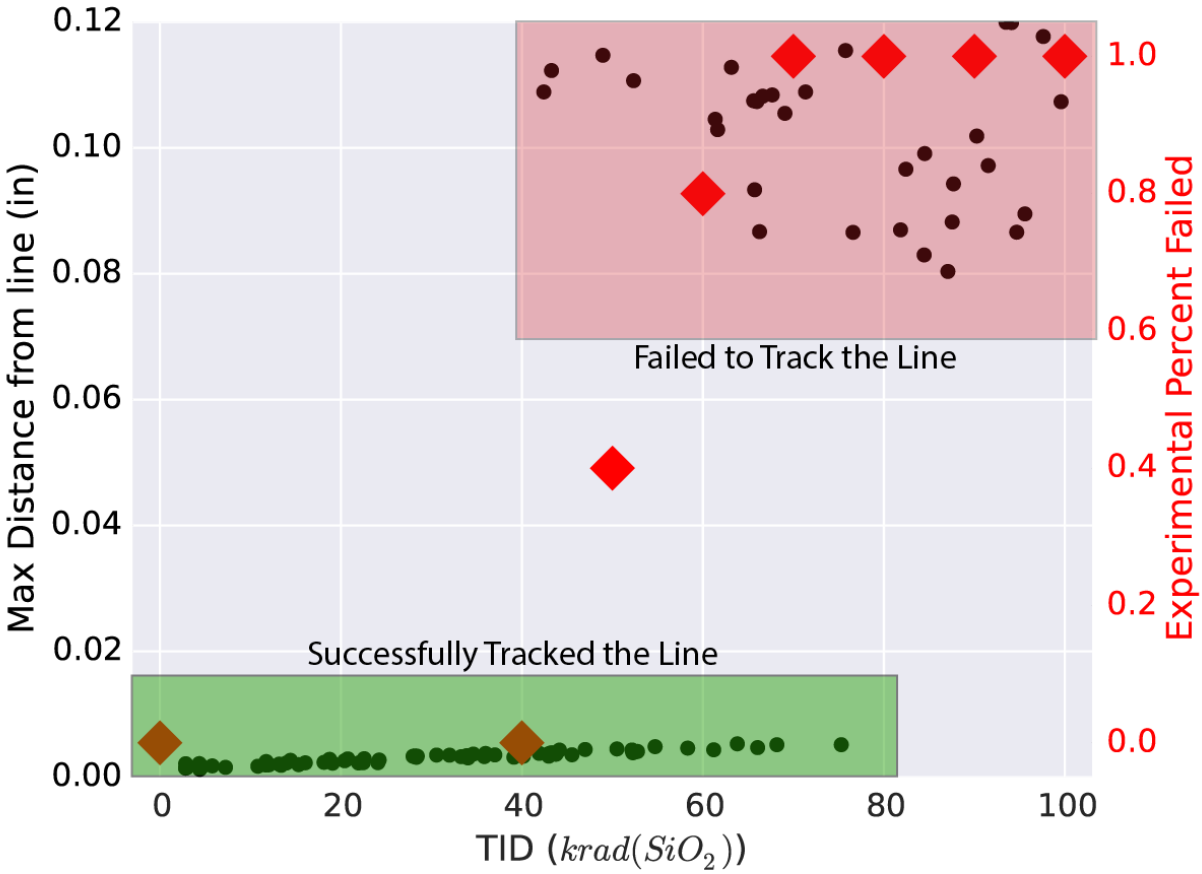

Figure 46. Comparison of Bayesian network and Saber result to actual full system test. Circles represent simulation results for maximum distance the robot deviates from the line (left axis) and diamonds represent experimental results for the ratio of robots that fail to track the line (right axis).

The framework can output results that are straightforward to interpret, such as cumulative probability distribution functions (CDF), informing design or operation decisions. Using the

samples generated by the PyMC tool and the Saber model, a binary search is performed to find the TID level where that sample will no longer track the line for two cases – nominal gain and an increased gain of the controller, show in in Figure 47. Performing this Bayesian-deterministic simulation with a sufficient number of sampling iterations allows for generation of the CDF. This result is highly informative and shows that our approach succeeds in our objective of predicting the effect of radiation degradation of components on robot system-level performance. The component is out of specification at 20 krad(SiO_2), but the system has <1% probability of failure through 40 krad(SiO_2). Using a derating factor of 3 in a component-hardness paradigm, the system would only be qualified for 7 krad(SiO_2) of TID, while the Bayesian network with a derating of 3 would qualify for 14 krad(SiO_2) or greater with the increased controller gain. In this case the Bayesian inference revealed that the qualification based on component information was too conservative. In other situations, for different systems, it is also possible to detect situations where the component qualification information is not conservative enough. System level testing attempts to address this concern, but it is expensive to test full systems, while the simulation approach can generate thousands of samples for only the cost of the simulation time. Furthermore, it is often difficult to diagnose from a system radiation test which component or subsystem caused the failure, whereas the Bayesian simulation approach can be used in a diagnostic mode to identify weak or failed components.

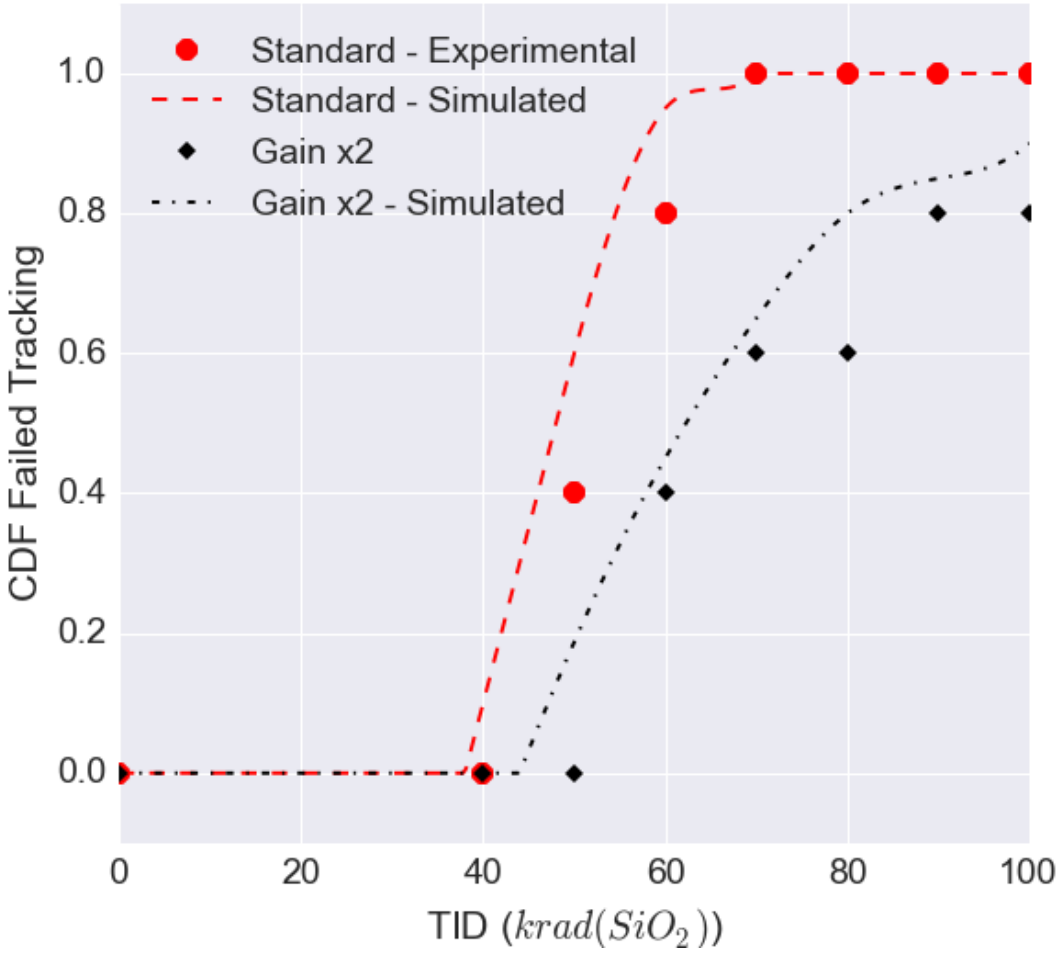

Figure 47. Cumulative probability distribution function of the robot's ability to track the line versus dose for nominal operating conditions and an increased controller gain. Points represent experimental data from 5 tested systems. Lines represent model predictions.

Different system configurations resulted in different dose levels where the transition from the normal line tracking mode to the failure to track mode occurs. A benefit of this framework is the capability of simulating the effect of radiation on a variety of configurations rapidly. Running the second control condition requires the change to one variable in the simulation (the controller gain), requiring 2 hours of run time on a dual core machine (the simulation can be parallelized to test more configurations rapidly or perform sweeps) In general the robot failure depends not only on component degradation, dose rate, and the propagation of the degradation through the system, but

also on software configuration of the robot at the time of failure. Appropriate component hardness levels should work for any software configuration, but this may be overly conservative since the control algorithm can potentially compensate for significant degradation.

Discussion

Model Flexibility

Since the inference is handled by the software package, model forms can be changed rapidly by altering single lines of code, without requiring any re-derivations of posterior distributions. Switching from a normal distribution to a log-normal distribution, a student-T distribution, or any other desired form is very straightforward. Additionally, the model is independent of the data, so new data can be incorporated into a preexisting model. Because of the independence assumptions inherent to a Bayesian network, each node is only affected by those to which it is directly connected, allowing subsections of models to be reused in larger system designs.

Bayesian Network's Role in Hardness Assurance

The modeling approach presented in this paper should be thought of as another tool available to designers of systems that will operate in a radiation environment. The network inference quality benefits from high quality component test information that follows the standards developed by the radiation effects community.

The models used to convert available data into parameters for the network can and should use available knowledge on how the devices degrade. A wealth of knowledge is available on the TID degradation of common components, and the models used in the network can naturally reflect this.

The use of radiation hardened parts can greatly simplify the analysis by eliminating nodes from the network. A part that shows no substantial degradation for the TID ranges of interest reduces

the number of TID dependent variables in the system. The Bayesian network approach can help identify where radiation hardened parts are necessary for high reliability designs.

Managing Model Complexity

Managing complexity is the primary challenge for expanding the approach to large systems (e.g. a spacecraft or robot). It is unrealistic to write equations that directly relate semiconductor parameters (the level impacted by radiation) to higher-level system functions. The solution to the complexity is modeling the system hierarchically, where each level corresponds to a level of abstraction. The work in this paper has three hierarchical levels – the environment, the components, and the system. A more complex system may have multiple hierarchy levels, between components and the system.

Defining the transitions and connections between hierarchical levels varies with the reliability question that is being investigated and the insight of designer/subject matter expert. In this work the TID degradation of the sensors was abstracted to only the white level of the sensor, the robot performance was abstracted to the max distance from the line, which was abstracted further to a success/failure criterion. For a complex system models could be built of each subsystem passing the appropriate level of detail up the hierarchy. The lower levels of the hierarchy will use voltages and currents, while the higher levels will use probability of availability.

Complex systems often defy intuitive analysis, but the Bayesian network simulation approach can allow visibility to the root cause of failures. Additionally, system data is very difficult to acquire for complex systems, and many tests must be performed to attempt to capture worst case conditions. For these reasons Bayesian networks may become more valuable the more complex the system becomes.

Software Injection of TID Degradation

Potential novel applications of this method include the determination of system level TID degradation through software injection. An example would be adjusting the voltage on an adjustable voltage regulator that could be reprogrammed to output different values that correspond to the degraded values corresponding to specified TID levels. This is a potential way to understand the system response without having to test all the components simultaneously.

Conclusions

The challenge in statistics based system prediction is an efficient representation and accurate modeling of the available information. The model must accurately and flexibly describe the characteristics of the system in a way that allows for efficient computation on a large parameter space. The method of choice in this work is continuous Bayesian networks, supplemented with traditional deterministic simulators where appropriate. The technique allows for the inclusion of all available radiation effects knowledge of the components and design information about the system, enabling the statistical prediction of the system degradation from component test information. The probabilistic Bayesian network modeling approach accounts for the uncertainty in the radiation environment, the part-to-part variation in radiation response, manufacturing uncertainty, and the uncertainty of the impact of simultaneous degradation of multiple components. The modeling method combines engineering system knowledge with component radiation measurement sets and physics-based models of the system behavior. The method is implemented on a small path-following robot, and it is shown that the probability of system failure as a function of dose can be predicted quantitatively. We demonstrate that the radiation failure point of the system occurs at a dose two to three times higher than the dose for the failure points of the individual components.

CHAPTER VII

SUMMARY AND CONCLUSIONS

This work identifies characteristics of robotic COTS component degradation through a case-study of distance sensors, primarily part-to-part variability and the interactions between the degradation of multiple components. Insight into the health of a class of components, microcontrollers, was developed using timing characteristics. A framework using either continuous or discrete Bayesian networks was developed to model the degradation observed in sensors and other electronic components. The Bayesian network can be incorporated with deterministic models for powerful analysis.

In summary, this work has identified the impact of and characteristics of sensor degradation. It proposed, measured, and modeled component health status indicators. Methods for integrating statistical modeling techniques, Bayesian networks, and commercial deterministic modeling tools were developed. Translation of fault propagation and functional effects captured in fault-modeling language models into Bayesian Network models were formalized. Saber simulation models were integrated into the Bayesian inference process. A functional discrete Bayesian modeling technique that incorporates the effects of radiation environment, component radiation measurements, electrical measurements related to radiation degradation, and system-level radiation measurements was demonstrated. This approach has many applications, enabling the design of systems with adequate hardness but without massive overdesign or large design margins to attain radiation hardness. Information from a graphical fault-modeling reliability program, component radiation data, continuous Bayesian statistical distributions, and physics-based deterministic simulators

were combined to create a continuous Bayesian modeling method that can predict the system-level radiation impact on performance measures with quantitative bounded uncertainty, using only radiation measurements on individual components. This result was demonstrated to agree well with actual system-level radiation degradation measurements. Conceptually, this method is a systematic way of capturing engineering or technological information, physics-based modeling information, and experimental or simulation data from various sources. The model gives a rigorous bound on uncertainty in component radiation measurement, part-to-part variation, and radiation environment.

The fundamental question of whether or not the system will function as designed in a radiation environment is challenging to answer because the abstraction layers used to make the system manageable to model and develop breakdown when basic circuit parameters such as threshold voltage and leakage current change significantly and in a highly variable manner from component to component. In each chapter in this work, the functionality of each system in question was recast from a new viewpoint to incorporate the impact of radiation while still approaching the system from a perspective that allows for manageable model complexity and simulation complexity. For the sensors this is done through modeling the dependencies of each component. For the microcontrollers, this is done through adding a health metric using maximum clock frequency. In order to make this approach scalable to a larger class of systems and not an ad-hoc case-by-case approach, Bayesian networks were adopted to provide a generalized solution to modeling the variability in radiation degradation of electromechanical systems. Using both continuous and discrete Bayesian networks depending on the datasets and the complexity of the model required, this work demonstrates that the functionality of the system can be modeled in such a way where probabilistic TID degradation information and deterministic system operation to answer high level

questions of system operation in TID environments and identify areas for design improvement. This combined approach allows for the evaluation of system health in TID environments using a quantitative and scalable approach that allows for detailed insight into system behavior. Given the appropriate model structure and radiation performance datasets, it is possible to quantitatively answer the motivating question – will the system operate as intended in the targeted radiation environment.

Ultimately, the next step would be to apply these developed techniques to a real world full scale system. This would require a significant team effort, including development of a component database for prior population, instrumentation of available health status indicators, and total-ionizing dose experiments. This would be a significant investment, but could yield a low-cost COTS rad-aware system.

APPENDIX A - FUTURE DIRECTIONS

This appendix highlights some of the promising future directions that were explored during the final stages of this work.

Cluster Computing for System Design Optimization

Like any Monte Carlo simulation, run time and sample size play a major role in the practicality of the approach. The Markov chain simulation process requires sequential sampling, which unfortunately limits the possibility of parallelization, although some work has been completed in this area [67], [68]. This process utilizes one burn-in chain, followed by several parallel chains to reach the desired sample count. Additionally, the deterministic portion of the simulation can be parallelized extensively to accelerate code execution. Towards this end preliminary simulations were developed on the Vanderbilt ACCRE compute cluster. This motivated the transition from Saber to the Python model presented in Appendix D.

If model run-time can be increased sufficiently (or if long runtimes are acceptable) the approach presented in Chapter VI can be used to optimize systems for their radiation response. Changes in control loop parameters or other design features can lead to dramatic improvements in radiation tolerance.

Informative Priors

One of the major advantages of the Bayesian approach is the simplicity of incorporating additional data sources through the use of informative priors. Figure 48 shows the data-set containing information on the degradation of the reflectance sensor from Chapter VI analyzed in a new way.

The dataset consisting of all the data was split into two equal sized datasets containing 30 samples each. Then, the modeling process from Chapter VI was calculated for the newly separated datasets. In the resulting plot, the “all data” distribution lies between the individual distributions for “dataset 1” and “dataset 2”. The “all data” dataset is the combination of both separate datasets. This shows that adding the second dataset to the first dataset produces a mixture of the two distributions. This illustrates the property of Bayesian analysis to combine disparate sources of information into a cohesive distribution.

Additionally, dataset 2 was re-analyzed with the addition of an informed prior. The informed prior was defined in PyMC to have a lower slope value than the media value for the non-informative prior posterior using dataset 2. This informed prior pulls the distribution towers lower slope values. The impact of the informed prior is show through skewing the distribution to the left on the plot. This illustrates how priors consisting of meaningful information such as similar type component performance can be used to pull the data towards the prior, allowing greater model flexibility.

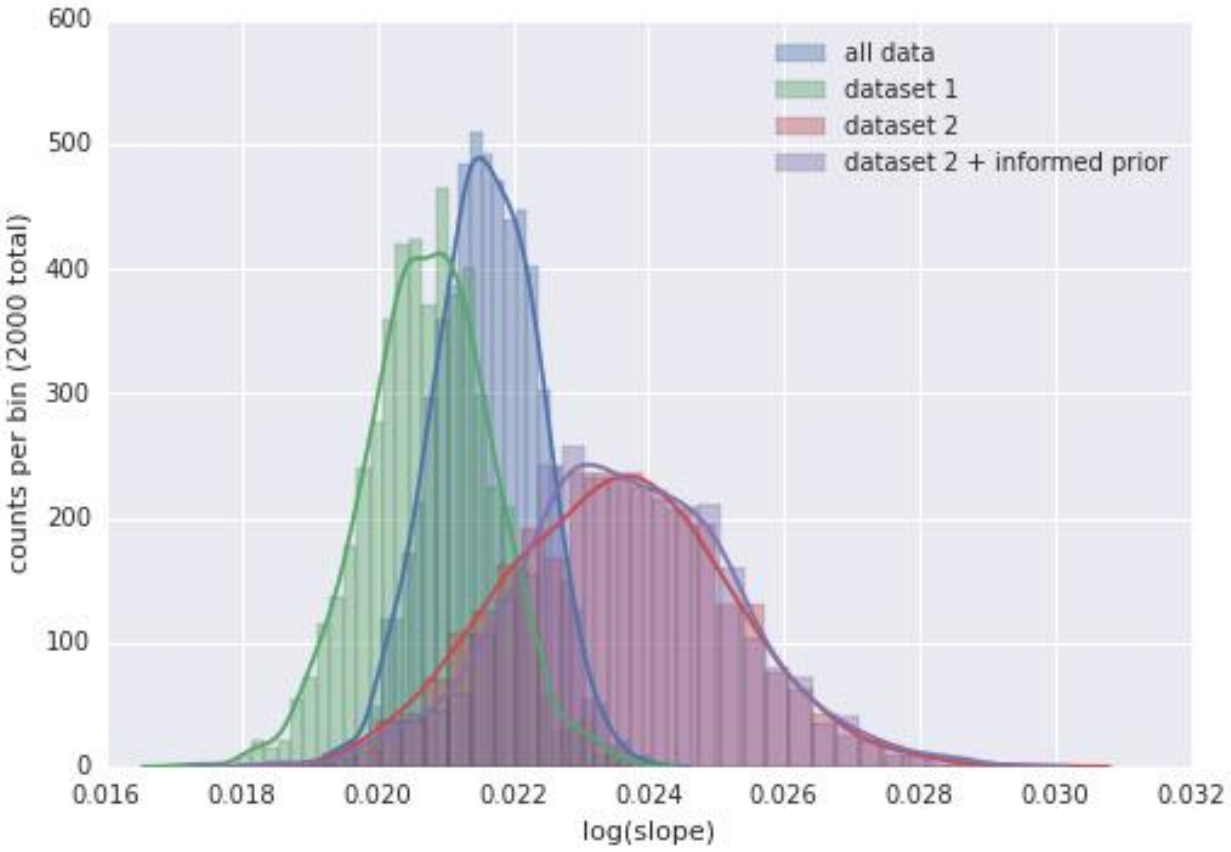

Figure 48. Comparison of modeling the data together, or as two groups using one of the groups as an informed prior. The distribution shifts depending on how the priors are chosen.

APPENDIX B - pyMC MODEL

Below is the fully pyMC model used for Chapter VI. All the datasets in Chapter VI were processed using this model, which generated the posterior distribution used in all subsequent plots. This code is written in Python and is meant to be run using the PyMC python package. Note the input data has been previously log transformed, so the model is a linear fit.

```
def sampling():#(true_m, true_c, true_tau)
        precision = 10
        # priors
        dose_model = pymc.Uniform("x", lower=0, upper=100, value=dose_matrix, observed=True)
        slope1 = pymc.Normal("slope1", mu=0.035, tau=precision)
        intercept1 = pymc.Normal("intercept1", mu=3.0, tau=precision)
        slope2 = pymc.Normal("slope2", mu=0.03, tau=precision)
        intercept2 = pymc.Normal("intercept2", mu=3.0, tau=precision)

        slope3 = pymc.Normal("slope3", mu=0.03, tau=precision)
        intercept3 = pymc.Normal("intercept3", mu=3.0, tau=precision)

        #define the linear model
        @pymc.deterministic(plot=True)
                def mu_1(slope1=slope1, intercept1=intercept1, dose_model=dose_model):
        return slope1*dose_model + intercept1

        @pymc.deterministic(plot=True)
                def mu_2(slope2=slope2, intercept2=intercept2, dose_model=dose_model):
        return slope2*dose_model + intercept2

        @pymc.deterministic(plot=True)
                def mu_3(slope3=slope3, intercept3=intercept3, dose_model=dose_model):
        return slope3*dose_model + intercept3

        #attach the model to the data in the variable "log_sensors_1"
        sensor1_model = pymc.Normal("sensor1", mu=mu_1, tau=precision, value=log_sensors_1,
        observed=True)
        sensor2_model = pymc.Normal("sensor2", mu=mu_2, tau=precision, value=log_sensors_2,
        observed=True)
        sensor3_model = pymc.Normal("sensor3", mu=mu_3, tau=precision, value=log_sensors_3,
        observed=True)
        dose_pred = pymc.Uniform("x_pred", lower=0, upper=100)

return locals()
```

APPENDIC C – MICROCONTROLLER TEST CODE

Below is the test code running on the microcontroller during the timing delay tests in Chapter III.

```
#include <avr/io.h>
#define F_CPU 16000000L
#include <util/delay.h>
#include "testdefines.h"
extern void test_counter0(void);
extern void test_counter_comparator_0_assy(void);
extern void test_opcodes(void);
//extern void test_pwm_in_assy(void);
//extern void test_pwm_out_assy(void);

void init_set_ack_pins()
{
	DDRC |= (1<<(PC1)); //set slave output ack
	DDRC &= ~(1<<(PC0)); // set slave input ack from master

	_delay_ms(50);

	PORTC &= ~(1<<(PC1)); //set slave out to 0.

}
void init_set_result_pins()
{
	DDRD=0xFF;
	PORTD=0xFF;
}
void init_set_test_code_input_pins()
{
	DDRC &= ~((1<<(PC2)) | (1<<(PC3)) | (1<<(PC4)) | (1<<(PC5)));
}

unsigned char read_test_code()
{
	return (PINC & (_BV(PC2) | _BV(PC3) | _BV(PC4) | _BV(PC5))) >> 2;
}


void  test_d2d()
{
	init_set_test_code_input_pins();
	init_set_result_pins();
	PORTD = read_test_code();

}
```

```
// read adc value
uint16_t adc_read(uint8_t ch)
{
	// select the corresponding channel 0~7
	// ANDing with '7' will always keep the value
	// of 'ch' between 0 and 7
	ADMUX = 0b11000010;
	//ch &= 0b00000111;  // AND operation with 7
	//ADMUX = (ADMUX & 0xF8)|ch;     // clears the bottom 3 bits before ORing

	// start single conversion
	// write '1' to ADSC
	ADCSRA |= (1<<ADSC);

	// wait for conversion to complete
	// ADSC becomes '0' again
	// till then, run loop continuously
	while(ADCSRA & (1<<ADSC));

	return (ADC);
}


void test_adc()
{
	uint16_t adc_result0;

	// initialize adc
	// AREF = AVcc
	ADMUX = (1<<REFS0);
	// ADC Enable and prescaler of 128
	// 16000000/128 = 125000
	ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);

	_delay_ms(50);

	while(!(PINC & 0x01))
	{
		adc_result0 = adc_read(2);      // read adc value at PC2 (ADC2)
		PORTD = (adc_result0 <<2);

	}
	return;

}


void test_sram(void)
{
```

```
	register unsigned char *p_val,sav,i;

	/* test all locations with 0x55, 0xAA and complement of address value */
	for (p_val=(unsigned char *)RAMSTART;p_val<((unsigned char *)(RAMEND));p_val++)
	{
		sav = *p_val;
		*p_val = 0x55;
		i = *p_val;
		if (i!=0x55) { PORTD = TEST_FAIL;  *p_val=sav; return; }
		*p_val = 0xAA;
		i = *p_val;
		if (i!=0xAA) {PORTD= TEST_FAIL;  *p_val=sav; return; }
		*p_val=sav;
	}

	PORTD = SRAMTEST;
	return;

}

/*
unsigned char test_counter_comparator_0(void)
{

	TCCR0A = 0x00;// Set the Timer Mode to Normal
	TCNT0=0x00; //reset
	TIFR0 = 0x00;//reset
	OCR0A = 0x55;// Set the value that you want to count to
	TCCR0B |=  (1 << CS00);// set prescaler to 1 and start the timer

	while ( (TIFR0 & (1 << OCF0A) ) == 0) {}	// wait for the overflow eve
	TCCR0B = 0;
	PORTD=0x02;
	return 0x02;
}
*/

void test_pwm_out()
{

	/*DDRD |= (1 << DDD6);// PD6 is now an output
	OCR0A = 128;  // set PWM for 50% duty cycle
	TCCR0A |= (1 << COM0A1);// set none-inverting mode
	TCCR0A |= (1 << WGM01) | (1 << WGM00);// set fast PWM Mode
	TCCR0B |= (1 << CS00) | (1 << CS02) ;// set prescaler to 1024 and starts PWM
	//while(1){}
	_delay_ms(100);
	TCCR0B = 0;
	*/
	PORTD = PWMOUTEST;
	return;
```

```
}

void test_pwm_in()
{
	/*
	DDRD |= (1 << DDD6);// PD6 is now an output
	PORTD &= ~(1 << PORTD6);

	TCCR0A = 0x00;// Set the Timer Mode to Normal
	TCNT0=0x00; //reset
	TIFR0 = 0x00;//reset
	OCR0A = 0x64;// Set the value that you want to count to
	TCCR0B |=  (1 << CS02)| (1 << CS01) |(1 << CS00);// set external clock source on T0 pin. Clock
on rising edge

	while ( (TIFR0 & (1 << OCF0A) ) == 0) {}	// wait for the overflow eve
	TCCR0B = 0;
	PORTD |= (1 << PORTD6);
	*/
	PORTD = PWNINTEST;
	return;
}


void run_test(unsigned char test_code)
{
	switch( test_code)
	{
		case D2DTEST: test_d2d(); break;
		case OPTEST: test_opcodes();break;
		case FLASHTEST: PORTD=FLASHTEST; break;
		case SRAMTEST: test_sram();break;
		case ADCTEST: test_adc(); break;
		case CTRTEST: test_counter0(); break;
		case COMPTEST: test_counter_comparator_0_assy(); break;
		case PWMOUTEST: test_pwm_out(); break;
		case PWNINTEST:test_pwm_in(); break;
		default:
			break;
	}
	return;

}

int main()
{
	init_set_ack_pins();
	init_set_result_pins();
	init_set_test_code_input_pins();
```

```
	unsigned char slaveStatus= 0;
	unsigned char masterStatus = 0;
	unsigned char testCode = 0;
	while (1)
	{


		//checking for master status on PC0
		//setting  slave status on PC1

		masterStatus = PINC & _BV(PC0); //read master status

		if ((slaveStatus == 0) && (masterStatus > 0))
		{
			init_set_result_pins();
			init_set_test_code_input_pins();
			testCode = read_test_code();
			PORTC |= (1<<(PC1)); // set slave status to 1
			slaveStatus = 1;
		}
		else if ((slaveStatus == 1) && (masterStatus == 0))
		{
			PORTC &= ~(1<<(PC1));// set slave status to 0
			run_test(testCode); //run test code
			slaveStatus = 0;
			init_set_test_code_input_pins();
		}
	}

	return 0;
}
```

APPENDIX D – PYTHON ROBOT MODEL

A detailed python physical deterministic model of the line-tracking robot was constructed to improve upon the Saber model. The motivation for putting the model in python was to enable faster simulation time, especially through parallel simulations on the Vanderbilt supercomputing cluster. By eliminating the need for a Saber license and enabling direct interaction with the simulation variables, a much more streamlined simulation approach was achieved. The full code is at the end of the section. The python robot model was used to evaluate the combined effect of degradation in the linear regulator on the robot and the line sensor on the robot.

Figure 49 shows the model track with the robot in three positions following the course. The smaller circles indicate the position of the robot and the x's indicate the position of each reflectance sensor. The axes are in pixels, where 10 pixels corresponds to 1 centimeter. The progression around the track is as expected for the nominal condition.

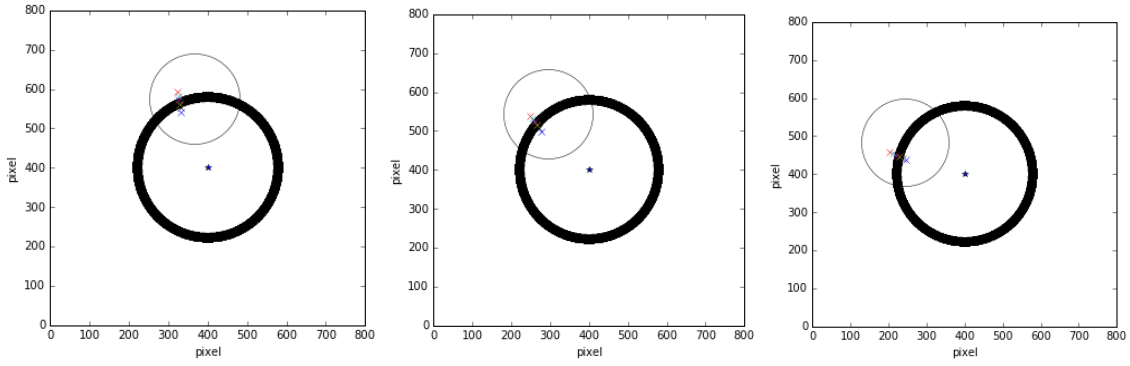

Figure 49. Simulations of python robot model progressing around the track.

Figure 50 shows the calibration curve for the line sensor as a whole and for each individual sensor, using the same data used to generate Figure 36. This curve uses a parabolic model with two parameters to model the response of the sensor to the presence of the line, which is documented in

the code and is an improvement on the model used in Chapter VI. By adding individual parameters for the vertex of the parabola and the width of the parabola, the accuracy of the simulation increases. This highlights the key point that this modeling approach has significant flexibility on how the deterministic model of the system is constructed.

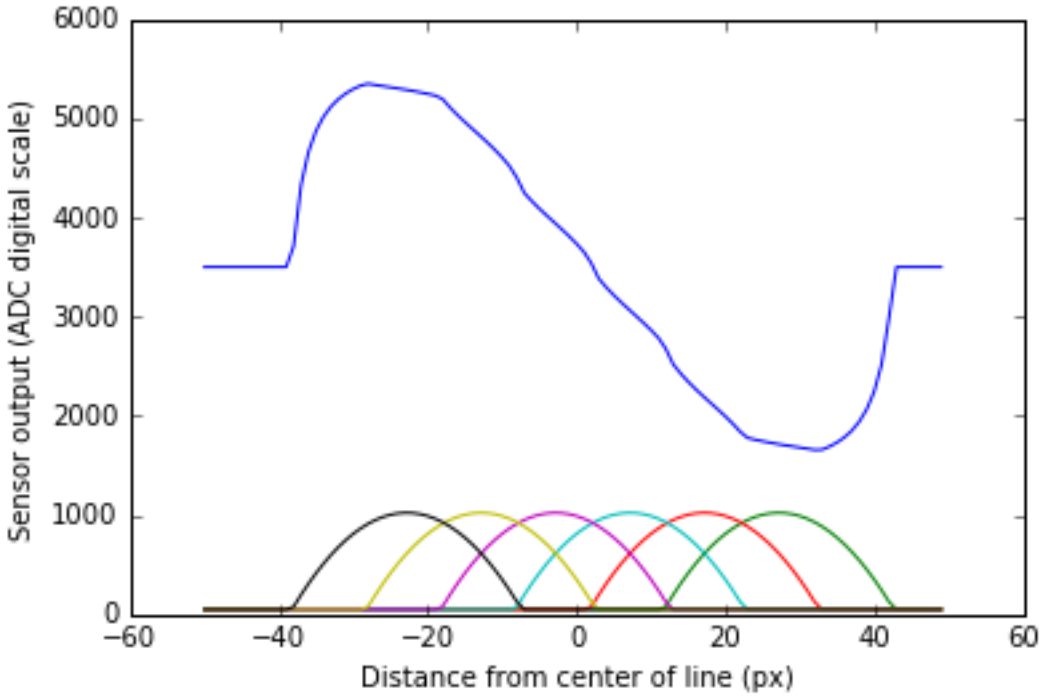

Figure 50. Calibration curve generated by sweeping the software model over the line, using the updated parabola model.

Figure 51 shows the degradation of the line sensor after 50 krad of dose based on the average data from Chapter VI with significant linear regulator degradation included. The impact of radiation on the linear regulator is primarily on the output voltage of the regulator, resulting in a 300 mV decrease per 50 krad of dose. The post-radiation performance of the linear regulator impacts the line sensor by reducing the max magnitude that the sensor can output due to the decrease in the voltage rail. The output voltage when the sensor is directly over the line has reduced by approximately 50%. The impact on the sensor directly due to TID manifests itself in the increase

in the output offset when the sensor is not over the line, increasing to approximately 200 on the analog-to-digital converter scale from approximately 50. The combined effect is a substantial reduction in the measureable signal because the signal range is reduced on the low end by the offset in the line sensor and on the high end by the decrease in max output voltage.

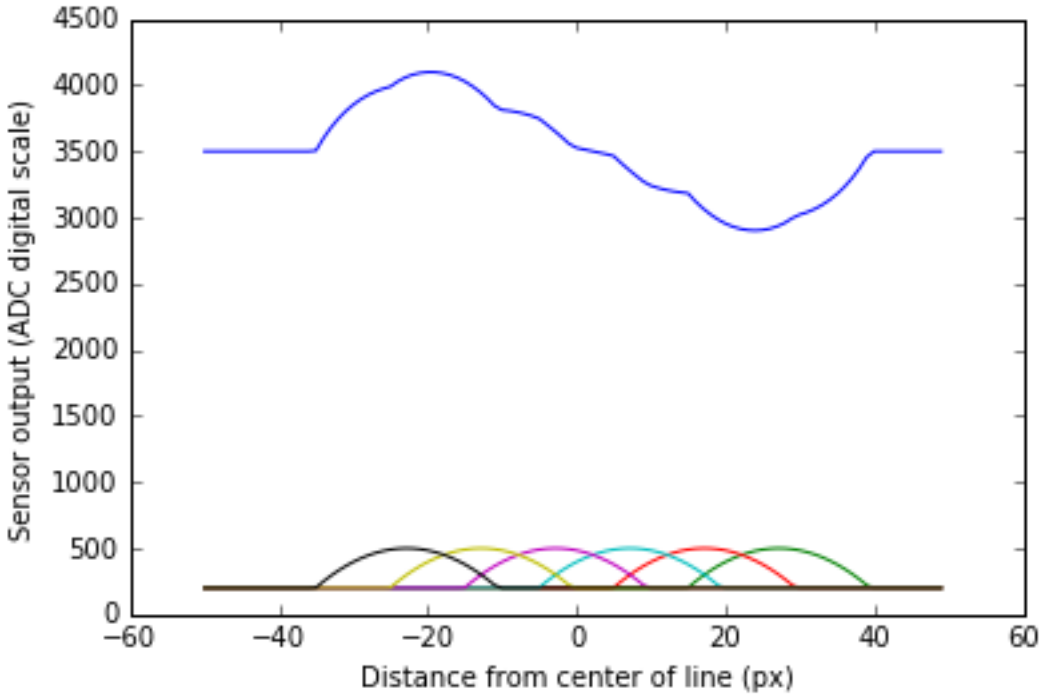

Figure 51. Degraded calibration curve at 50 krad. Note the decreased amplitudes of the parabolas and the increase from 0 in the offset.

Figure 52 shows the results of running multiple simulations, sweeping the simulated degradation in the voltage regulator from full degradation at 0 (corresponds to 0 V) to no degradation at 1023 (corresponds to 5 V). Each simulation was for a different ADC max value, with 1024 simulations run in total. The plot shows a sharp transition at ~100 ADC units (500 mV) in the ability to track the line. Below this point, the robot cannot track the line due to inadequate sensing range. After this point, the robot gains the ability to track the line successfully. This effect is a new addition and was not modeled in the previous chapters. The benefit of this modeling approach is the

straightforward ability to add basic mechanisms and data iteratively to the model, gaining increased fidelity with the additional modeling effort.

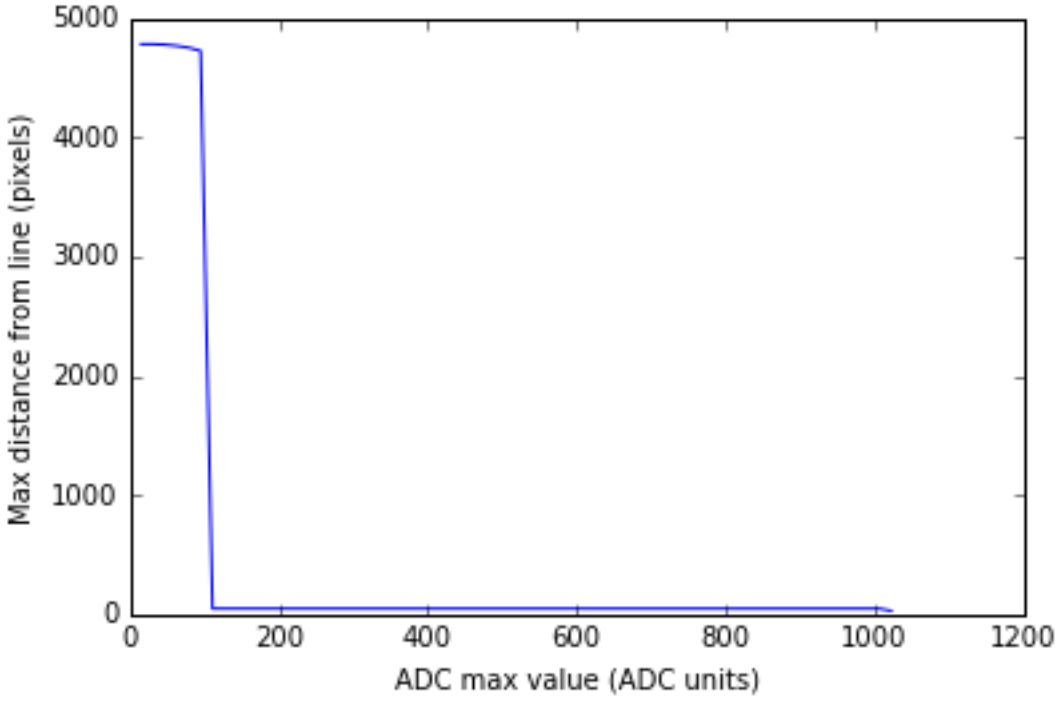

Figure 52. Simulation of robot line-tracking performance versus the max achievable sensor output due to degradation in the linear regulator.

Figure 53 shows the results of a simulation of the impact of TID on the sensor element. The primary impact of TID in the reflectance sensor is an increase in the output when the sensor is not over the line. The plot shows that for sensor white-level offset values below approximately 450, the robot is able to track the line successfully. Once the sensor has degraded to the point that the reflectance offset white level is greater than 450 units on the ADC scale (~2.3 V), the robot is no longer able to track the line successfully.

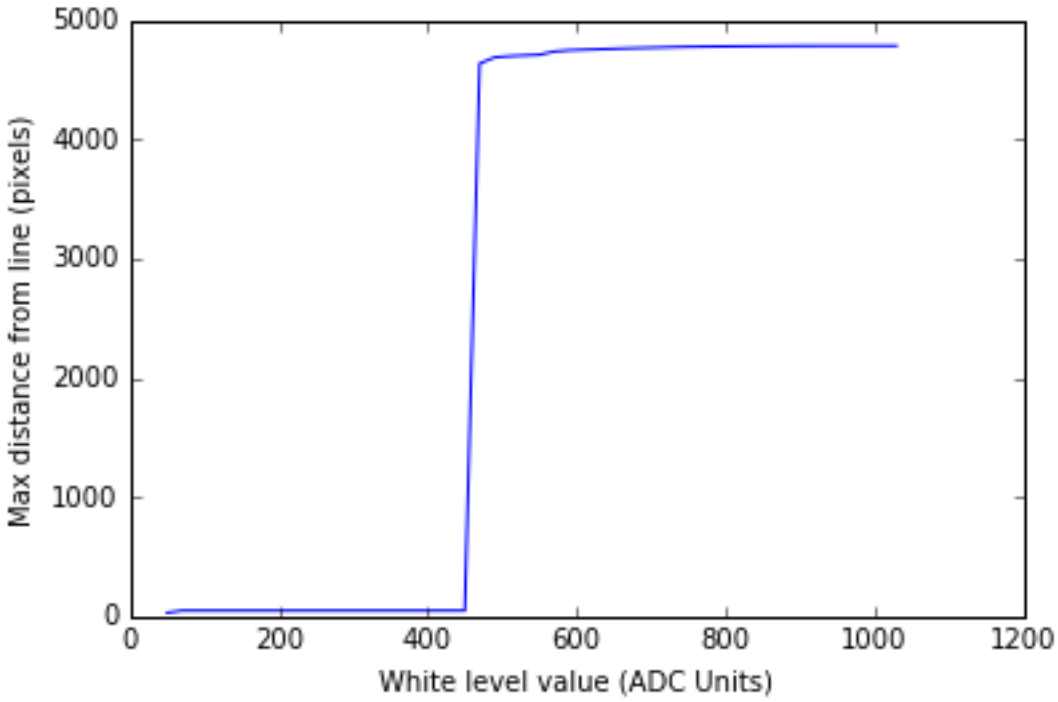

Figure 53. Simulation of the effect of the white level offset in the sensor on tracking performance.

The above figures demonstrate the functionality of the model in isolation. The combined effect of degradiation in the sensor and degradiation in the linear regulator was simulated using an expanded pyMC model. To incorporate the degradiation of the line sensor, the following line was added to the model in Appendix B:

```
@pymc.deterministic(plot=True)
        def mu_regulator(slope=regulator_slope,intercept=regulator_intercept,
dose_model=dose_model):
return slope1*dose_model + intercept1
regulator_model = pymc.Normal("regulator", mu=mu_regulator, tau=precision, value=regulator_data,
observed=True)
```

The “regulator_model” assumes linear change in the output of the regulator with dose. The model uses non-informative priors for its mean and precision. The model accepts values for regulator output voltage versus dose, producing a distribution for the posterior distribution of slope and intercept variables that define the response of the regulator to dose.

Figure 54 shows the output of the pyMC model for the linear regulator and sensor. The experimental data for the linear regulator showed minimal spread, producing a tighter distribution versus dose than the sensor data. Overall, the sensor degradation shows more drift than the linear regulator data does. The combined effect is a reduction in sensor output range.

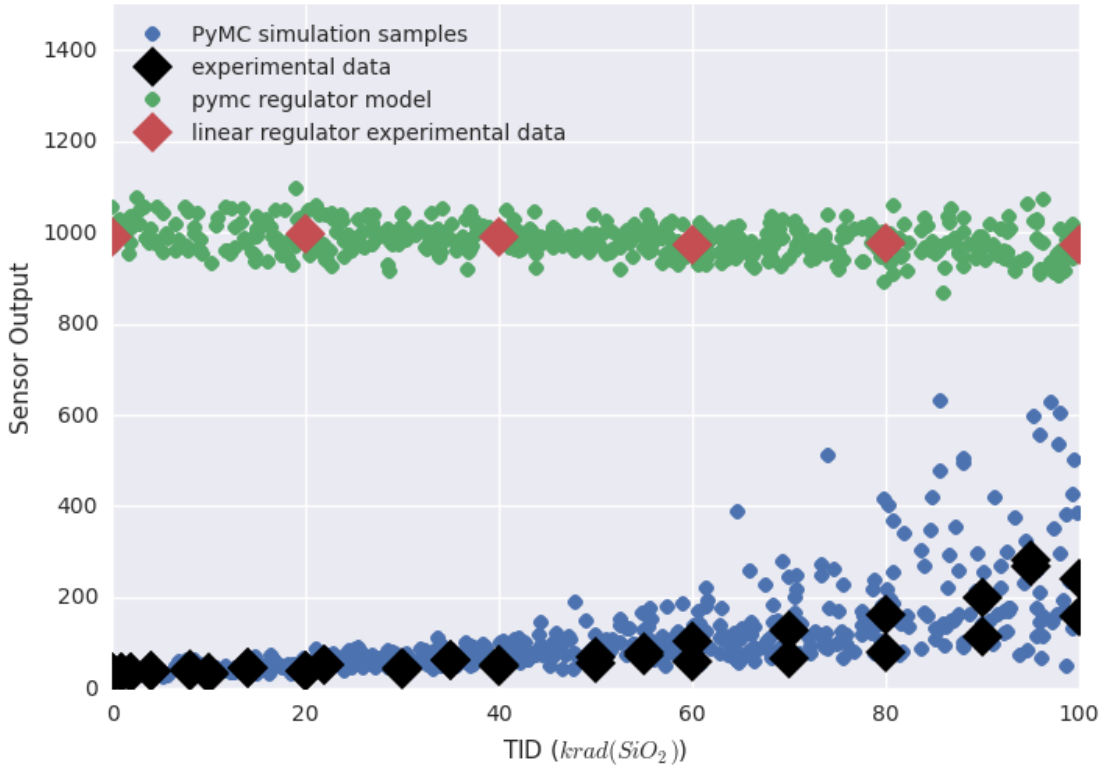

Figure 54. Results of combined linear regulator model and sensor model.

The linear regulator datasets showed limited degradition in output voltage, so the slope was increased by a factor of 10x for the plot in Figure 55 to highlight the combined effect of the two degradition modes. The plot of the cumulative probability of failed tracking is identical from Chapter VI Figure 47, and there are two new lines: the probability of failure due to just the regulator versus dose, and the combined probability of failure due to degradiation in both devices.

Importantly, the combined degradation occurs earlier in dose than either curve individually, but it overall follows the characteristics of the sensor only curve.

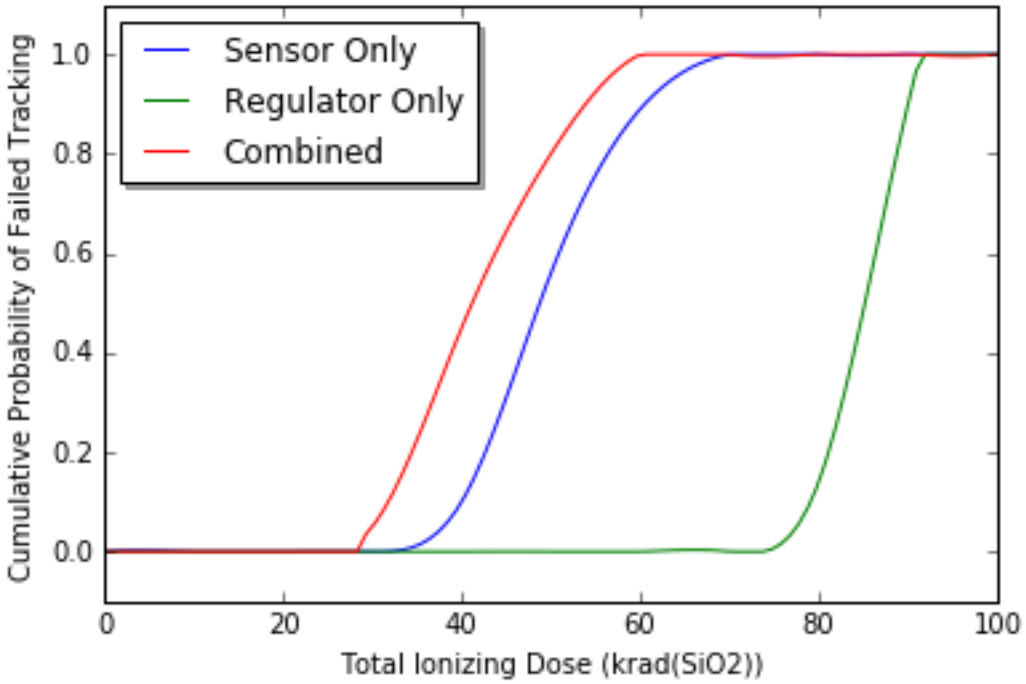

Figure 55. Plot of probability of failed tracking of just the sensor degrading, just the regulator degrading, and both degrading.

Below is the Python code that produces all the graphs in this section:

```
import numpy as np
from PIL import Image
import scipy as sp
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.animation as animation

position = 50

class Course:
    origin = [400,400]
    size = [800,800]
    track = np.zeros(size)
    def generate(self,option):
        if(option == 1):
```

```
            radius = 14.0*25.4/2.0
            width = 25.4/2.0
            for i in range(self.size[0]):
                for j in range(self.size[1]):
                    distanceFromCenter = sp.spatial.distance.pdist([[self.origin[0],self.origin[1]],[i,j]])
                    if((distanceFromCenter < (radius + width)) & (distanceFromCenter > (radius - width))):
                        self.track[i,j] = 0
                    else:
                        self.track[i,j] = 255

class Robot:

    max_error = 0

    origin = [400,400]
    radius = 14.0*25.4/2.0

    max_adc = 1024
    white_level_offsets = [50,50,50,50,50,50]

    track = []
    position = [360,220]
    sensor_positions = [[0,0],[0,0],[0,0],[0,0],[0,0],[0,0]]
    angle = 0

    wheel_width = 90
    sensor_forward_offsets = 40.0
    sensor_sideways_offsets = [-25.0,-15.0,-5.0,5.0,15.0,25.0]

    sensor_height = 10.
    #field_of_view = 60.0/180.0*np.pi
    field_of_view = .001

    num_rays = 1000

    proportional = 0
    integral = 0
    derivative = 0

    p_constant = 1.0/40.0
    i_constant = 1.0/10000
    d_constant = 3.0/2.0

    command_position = 3500
```

```
max_speed = 200

def initialize(self, proportional, integral, derivative, start_position, command_position, track):
    self.p_constant = proportional
    self.i_constant = integral
    self.d_constant = derivative
    self.position = start_position
    self.command_position = command_position
    self.track = track
    self.proportional = self.calculatePosition()  - command_position
    self.integral = 0
    self.derivative = 0
    self.updateSensorPositions()

def update(self):
    left_increment=0
    right_increment=0

    power_difference = self.controllerOutput()
    #print(power_difference)
    if(power_difference > self.max_speed):
        power_difference = self.max_speed
    if(power_difference < (-1.0*self.max_speed)):
        power_difference = (-1.0*self.max_speed)

    if(power_difference < 0):
        left_increment = self.max_speed/20.0
        right_increment = (self.max_speed + power_difference)/20.0
    else:
        right_increment = self.max_speed/20.0
        left_increment = (self.max_speed - power_difference)/20.0


    new_proportional = self.calculatePosition() - self.command_position
    self.derivative = new_proportional - self.proportional
    self.integral = self.integral + new_proportional
    self.proportional = new_proportional

    #print(self.calculatePosition())
    #print(left_increment,right_increment)
    #print(self.angle)

    newPosition_x = (left_increment+right_increment)/2.0 * np.cos(self.angle)
    newPosition_y = (left_increment+right_increment)/2.0 * np.sin(self.angle)
    self.position = [self.position[0]+newPosition_x,self.position[1]+newPosition_y]
```

```
    self.angle = self.angle + np.arctan((right_increment-left_increment)/self.wheel_width)
    #print(self.angle)
    self.updateSensorPositions()

def runCourse():
    print('test')

def controllerOutput(self):
    new_proportional = self.calculatePosition() - self.command_position
    #self.derivative = new_proportional - self.proportional
    #self.integral = self.integral + new_proportional
    #self.proportional = new_proportional
    power_difference = (self.p_constant * new_proportional)
                        #+ self.i_constant * self.integral
                        #+ self.d_constant * self.derivative)

    return(power_difference)

def plotCenter(self,ax):
    ax.plot(self.position[0],self.position[1],'o',markersize=80,mfc='none')

def calculatePosition(self):
    sensor_readings = self.readSensors()
    if(sum(sensor_readings)==0):
        position = 0;
    else:
        position = ((1000*sensor_readings[0]+2000*sensor_readings[1]+
                    3000*sensor_readings[2]+4000*sensor_readings[3]+
                    5000*sensor_readings[4]+6000*sensor_readings[5])/
                    (sum(sensor_readings)))
    #print(position)
    return(position)

def readSensors(self):
    option = 2
    output = []
    for i in range(0,6):
        output.append(self.readSensor(i,option,self.track))

    return(output)

#2 methods to do this
#     weigh spatial distance equally -> option 1
#     weigh angular distance equally -> option 2
def readSensor(self,index,option,track):
```

```
            option = 3
        if(option==1):
            print('test')
        if(option==2):
            in_view = 0
            roll_angles = np.linspace(0,2*np.pi,10)
            pitch_angles = np.linspace(0,pololu.field_of_view*2,10)
            for roll in roll_angles:
                for pitch in pitch_angles:
                    distance = self.sensor_height * np.tan(pitch)
                    x_offset = distance * np.cos(roll)
                    y_offset = distance * np.sin(roll)
                    x_check = np.round(self.sensor_positions[index][0] + x_offset)
                    y_check = np.round(self.sensor_positions[index][1] + y_offset)
                    if(track[x_check,y_check]==0):
                        in_view = in_view + 1
            return(in_view/100.0*1024.0)
            if(option==3):
                x = self.sensor_positions[index][0]
                y = self.sensor_positions[index][1]
            h = self.origin[0]
                k = self.origin[1]
                distance_from_line = np.absolute(np.sqrt((x-h)*(x-h)+(y-k)*(y-k))-self.radius)
                if (distance_from_line > self.max_error):
                        self.max_error = distance_from_line

            new_value = distance_from_line * distance_from_line/(-1.0/(4*self.field_of_view *
self.max_adc))+self.max_adc

            if new_value < (self.white_level_offsets[index]):
                new_value = self.white_level_offsets[index]
            if new_value > (self.max_adc):
                new_value = self.max_adc

                return new_value


    def updateSensorPositions(self):
        i = 0
        for sensor_sideways in self.sensor_sideways_offsets:
            self.sensor_positions[i][0] = (self.position[0] + self.sensor_forward_offsets * np.cos(self.angle)
                                    + sensor_sideways * np.sin(-1.0*self.angle))
            self.sensor_positions[i][1] = (self.position[1] + sensor_sideways * np.cos(-1.0*self.angle)
                                    + self.sensor_forward_offsets * np.sin(self.angle))
            i = i+1
```

```
def plotSensors(self,ax):
    for sensor in self.sensor_positions:
        ax.plot(sensor[0],sensor[1],'x')

def plotSixSensorCalibration(self):
        pololu.position = [360,170]
        pololu.updateSensorPositions()
        x = []
        y = []
        z = []
        for i in range(100):
            x.append(i - 50)
            y.append(self.calculatePosition())
            z.append(self.readSensors())
            self.position[1] = self.position[1]+1
            self.updateSensorPositions()

        sensor1 = []
        sensor2 = []
        sensor3 = []
        sensor4 = []
        sensor5 = []
        sensor6 = []
        for row in z:
            sensor1.append(row[0])
            sensor2.append(row[1])
            sensor3.append(row[2])
            sensor4.append(row[3])
            sensor5.append(row[4])
            sensor6.append(row[5])

        plt.figure()
        plt.plot(x,y)
        plt.plot(x,sensor1)
        plt.plot(x,sensor2)
        plt.plot(x,sensor3)
        plt.plot(x,sensor4)
        plt.plot(x,sensor5)
        plt.plot(x,sensor6)
        plt.xlabel("Distance from center of line (px)")
        plt.ylabel("Sensor output (ADC digital scale)")
        plt.show()

def plotSensorCalibration():
```

```
        print("not implemented")



class Sensor:
    test = 0

circleCourse = Course()
circleCourse.track = np.load("course_variable.npy")
#circleCourse.generate(1)
#trackImg = Image.fromarray(circleCourse.track)
#trackImg.save("test2.gif")

pololu = Robot()
pololu.initialize(proportional=1.0/5.0, integral=1.0/10000.0, derivative = 3.0/2.0, start_position=[360,220],
command_position=3500, track=circleCourse.track)

def plot_frame(i):
        f, ax = plt.subplots()
        ax.imshow(circleCourse.track,origin="lower",cmap="gray")
        ax.plot(400,400,'*')
        #ax.plot(500,600,'x')
        pololu.plotCenter(ax)
        pololu.plotSensors(ax)
        f.set_size_inches(10, 10, forward=True)
        plt.xlim(0,800)
        plt.ylim(0,800)
        #plt.plot(515.345240492,262.814019017,'*',color='white',markersize=20)
        plt.savefig("plots/fig_" + str(i))
        plt.close()

def update_10():
        for i in range(10):
                pololu.update()
def plot_course():
        for i in range(50):
                update_10()
                plot_frame(i)
                print(pololu.position)

def plot_summary():
        x = []
        y = []
        for i in range(500):
                pololu.update()
```

```
                x.append(pololu.position[0])
                y.append(pololu.position[1])

        f,ax = plt.subplots()
        ax.plot(x,y)
        plt.show()

def pass_fail(adc_max,white_levels):
    pololu.initialize(proportional=1.0/5.0, integral=1.0/10000.0, derivative = 3.0/2.0, start_position=[360,220],
command_position=3500, track=circleCourse.track)
    pololu.max_adc = adc_max
    pololu.white_level_offsets = white_levels

    for i in range(500):
        pololu.update()


    return pololu.max_error
    if(pololu.max_error > 100):
        return "fail"
    else:
        return "pass"

pololu.max_error = 0
#pololu.plotSixSensorCalibration()
#plot_summary()
#plot_course()
adc_run_value = 1023
#white_offsets = [50,50,50,50,50,50]
white_offsets = np.full(6,50)

def fig1():
    print "generating figure 1"
    print "3 frames of track"
    for i in range(3):
        f, ax = plt.subplots()
        ax.imshow(circleCourse.track,origin="lower",cmap="gray")
        ax.plot(400,400,'*')
        #ax.plot(500,600,'x')
        pololu.plotCenter(ax)
        pololu.plotSensors(ax)
        f.set_size_inches(5, 5, forward=True)
        plt.xlim(0,800)
        plt.ylim(0,800)
        plt.xlabel("pixel")
```

```
        plt.ylabel("pixel")
        update_10()

def fig2():
    pololu.max_error = 0
    pololu.plotSixSensorCalibration()

def fig3():
    pololu.max_adc = 500
    pololu.white_level_offsets = [200,200,200,200,200,200]
    pololu.white_level_offsets = [200,200,200,200,200,200]
    pololu.plotSixSensorCalibration()

def fig4():
    x = []
    max_distance = []
    for i in range(0,64):
        x.append(1023-i*16)
        max_distance.append(pass_fail(adc_run_value-i*16,white_offsets))

    plt.plot(x,max_distance)
    plt.xlabel("ADC max value (ADC units)")
    plt.ylabel("Max distance from line (pixels)")

def fig5():
    x = []
    max_distance = []
    for i in range(0,50):
        x.append(50+i*20)
        max_distance.append(pass_fail(adc_run_value,np.full(6,50+i*20)))

    plt.plot(x,max_distance)
    plt.xlabel("White level value (ADC Units)")
    plt.ylabel("Max distance from line (pixels)")

def fig7():
    x = [0,10,20,30,40,50,60,70,80,90,100]
    regulator = []
    line_sensor = []
    combined = []
    plt.plot(x,regulator)

#print("Max error: " + str(pololu.max_error))

#print("Success")
```

#fig1()
#fig2()
#fig3()
#fig4()
#fig5()
fig7()

APPENDIX E – ROBOT EXPERIMENTAL DETAILS

The primary element of interest in the model robot is the reflectance sensor. Full details of the reflectance sensor can be found at the product information at:

https://www.pololu.com/product/960

A paper guide was developed to characterize the line sensor. The robot was swept through a series of measurement points on the paper that correspond to distances from a printed line to characterize the performance of the reflectance sensor.

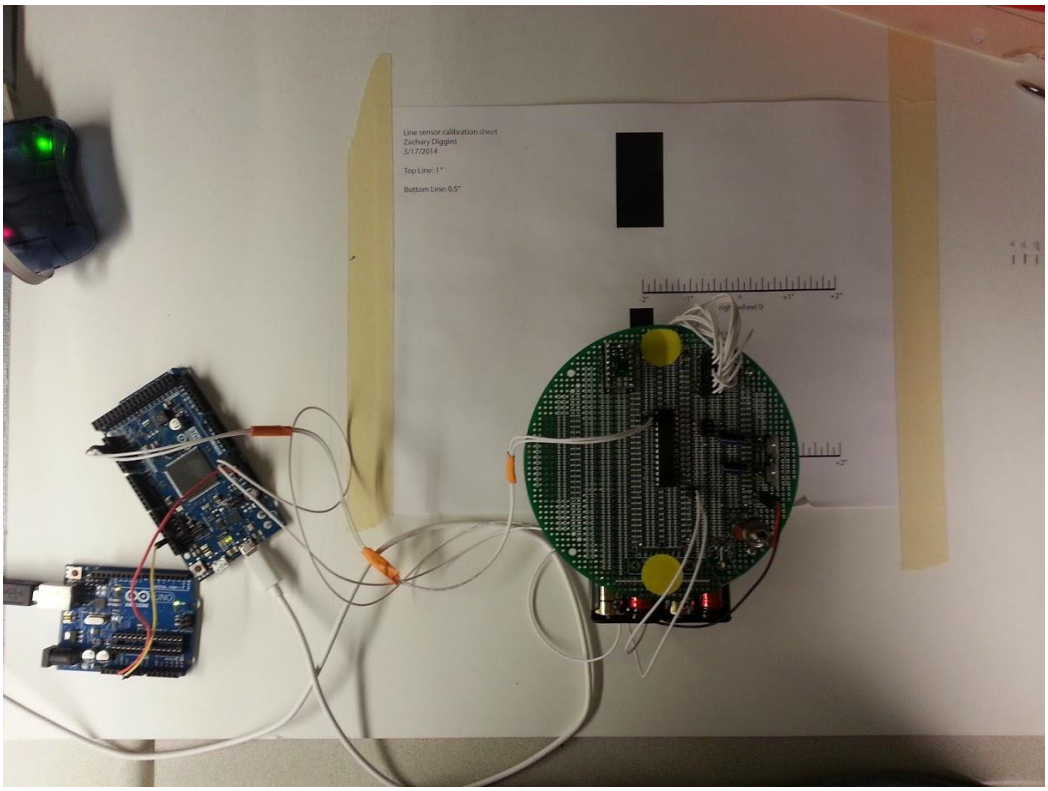

Figure 56. Image of model robot and line sensor characterization setup.

A camera mount was used to record video of the full system functioning. This allowed for straightforward post-processing with minimal error introduced due to the angle of the video. The video data allowed for comparison with system level simulation models.

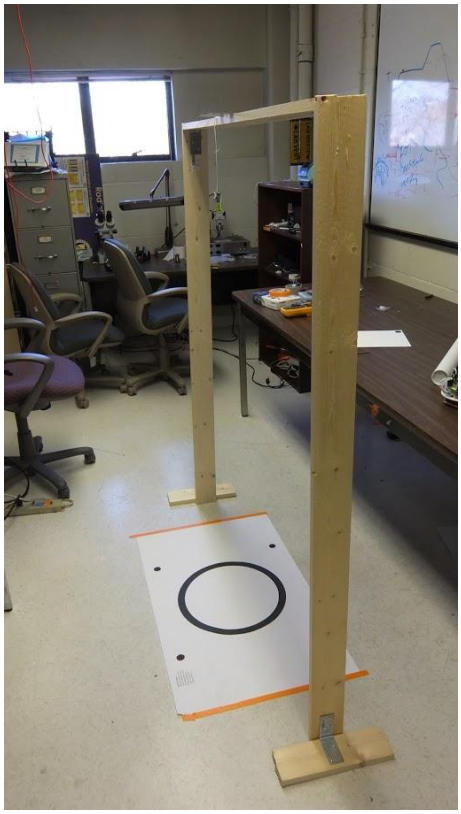

Figure 57. Camera mount with track on the floor. Camera is placed in the middle of the mount and uses the four dots on the floor for image processing.

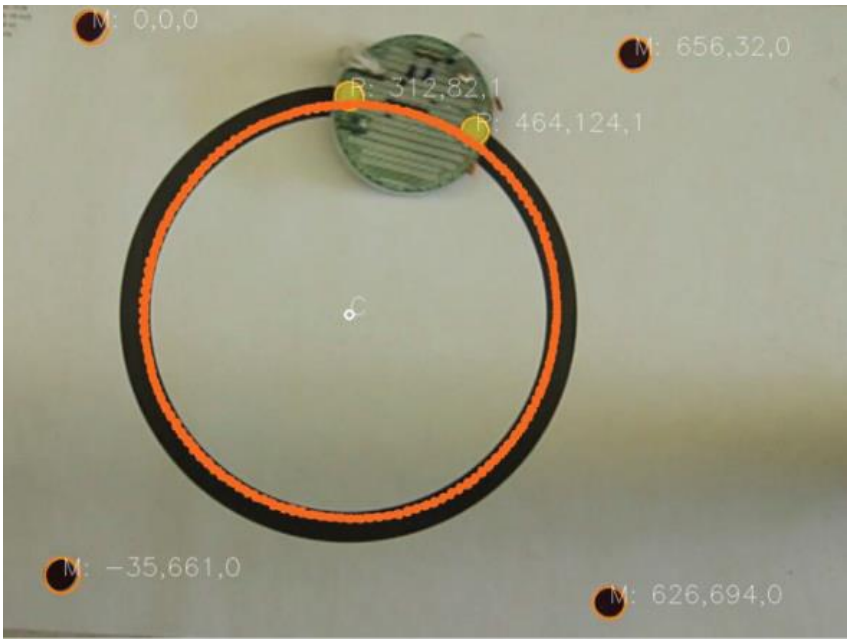

Figure 58. Image after video processing, showing the track the robot followed in orange. This is an example of successful tracking.

Figure 59. Image after video processing. This is an example of failed tracking, with the robot spiriling in circles near the bottom of the image.

Code for video processing:

```
import cv2, cv
import numpy as np
import math, time

inch = 0
inches = 0
view=1

#####
def show(img):
    cv2.namedWindow('Untitled',flags=cv2.WINDOW_NORMAL)
    cv2.imshow('Untitled',img)

def process_image(h):
    img = cv2.imread(h,cv2.CV_LOAD_IMAGE_COLOR)
    # Find the red boundary markers
    markers = locate_markers(img,hue=(-20,0),sat=(100,255),val=(70,255),show=True)
    print markers
    show(img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

def locate_track(img,show=False):
    track = threshold(img,hue=(0,179),sat=(0,255),val=(0,60))
    #cv2.imshow('video', track)
    #return

    #contours,hierarchy = cv2.findContours(track,cv2.RETR_LIST,cv2.CHAIN_APPROX_SIMPLE)
    #track = None
    #track_perimeter = 0
    #for contour in contours:
    #    p = cv2.arcLength(contour, True)
    #    if p > track_perimeter:
    #        track_perimeter = p
    #        track = contour
    if show:
        #cv2.drawContours(img,contours,-1,(255,0,255),5)
        cv2.imshow('video', track)
    return track

def show_mask(h,hue,sat,val):
    cap = cv2.VideoCapture(h)
    frame_count = 0
    while True:
        flag, frame = cap.read()
```

```
        frame_count = frame_count + 1
        # Preprocess
        blur = cv2.GaussianBlur(frame,(5,5),0)
        #img = threshold(blur,hue,sat,val)
        locate_markers(blur,hue,sat,val,sz=(15,30),show=True)
        cv2.imshow('video', blur)
        if cv2.waitKey(10) == 27:
            break

def dot(v1,v2):
    return v1[0]*v2[0] + v1[1]*v2[1]

def mag(v1):
    return math.sqrt(v1[0]**2 + v1[1]**2)

def process_video(h):
    cap = cv2.VideoCapture(h)
    fps=30
    second=30*fps
    #out = cv2.VideoWriter('output.avi',cv2.cv.CV_FOURCC('M','J','P','G'),20,(1920,1080))
    #out = cv2.VideoWriter('output.avi',cv2.cv.CV_FOURCC('M','J','P','G'),30,(1280,720))

    pos_history=[]
    frame_count = 0
    while True:
        flag, frame = cap.read()
        frame_count = frame_count + 1
        # Preprocess
        blur = cv2.GaussianBlur(frame,(5,5),0)
        boundaries = locate_boundary(blur,show=True)
        center = locate_center(blur,boundaries,show=True)

        calibrateDistance(boundaries)
        robot = locate_robot(blur,show=False)

        # Add overlays
        if center:
            annotate_track(blur,tuple([int(x) for x in center]),7.*inches)
        torigin = annotate_boundaries(blur,boundaries)
        if torigin:
            origin = torigin
        origin=(0,0)
        pos = annotate_robot(blur,robot,origin)
        if pos:
            robot_pos = ((pos[0][0]+pos[1][0])/2., (pos[0][1]+pos[1][1])/2.)
```

```
pos_history.append(robot_pos)

print frame_count, robot_pos[0], robot_pos[1]
if center:
    r = distance_from_center(robot_pos,center)
    center = (center[0]-origin[0],center[1]-origin[1])
    error = 7.0*inches - r

    if len(pos_history) > 2:
        for pos in pos_history:
            updated_pos = (int(pos[0]+origin[0]),int(pos[1]+origin[1]))
            cv2.circle(blur,updated_pos,5,(20,105,255),-1)

    v_angle = 0.0
    v_mag = 0.0
    tan_angle = 0.0
    H = 0.0

    #vel_vector = velocity_vector(pos_history[-2],pos_history[-1],1)
    orient_vector = (robot[1][0] - robot[0][0],robot[1][1] - robot[0][1])
    (dx,dy)=orient_vector
    v_angle = math.atan2(dy,dx)
    if v_angle < 0:
        v_angle = v_angle + 2*math.pi
    v_angle = v_angle * 360. / (2.*math.pi)
    v_mag = mag(orient_vector)/inches*fps

    pos_vector = (robot_pos[0]-center[0],robot_pos[1]-center[1])
    (dx,dy)=pos_vector
    # Get tangent vector by rotating position pi/2 radians
    tan_vector = (dx*math.cos(-math.pi/2.) - dy*math.sin(-math.pi/2.),
                  dx*math.sin(-math.pi/2.) + dy*math.cos(-math.pi/2.))
    tan_vector = unit_vector(tan_vector)
    # Angle between orientation and tangent
    H = math.acos( dot(orient_vector,tan_vector)/(mag(orient_vector)*mag(tan_vector))) # radians
    #H = math.atan2(tan_vector[1],tan_vector[0]) - math.atan2(vel_vector[1],vel_vector[0])

    #print "F=%d T=%0.3g sec" % (frame_count, frame_count/float(fps)),
    #print " P=(%.3g in, %.3g in)" % ((robot[0]-center[0])/inch, (robot[1]-center[1])/inch),
    #print " E=%.3g in" % (error/inches),
    #print " V=(%.3g in/sec,%.3g deg)" % (v_mag, v_angle),
    #print frame_count, pos_history[-1][0], pos_history[-1][1]
    ang = H * 360 / (2.*math.pi)
    if ang > 90:
        ang = 180 - ang
```

```
            orient_vector = (-orient_vector[0],-orient_vector[1])
        orient_vector = unit_vector(orient_vector)

        # Is the orientation inside or outside?
        a=distance_from_center(add_vector(pos_vector,orient_vector),center)
        b=distance_from_center(add_vector(pos_vector,tan_vector),center)
        if a > b:
            # Moving outside
            ang = abs(ang)
            ocolor = (0,255,0)
        else:
            # Moving inside
            ang = -abs(ang)
            ocolor = (0,0,255)

        draw_vector(blur,robot_pos,[100*x for x in tan_vector])
        draw_vector(blur,robot_pos,[100*x for x in orient_vector],ocolor)

        #print frame_count, ang
        time.sleep(0.1)

    cv2.imshow('video', blur)
    #out.write(blur)
    if cv2.waitKey(10) == 27:
        break

# Functions that work on the image
def threshold(img,hue=(0,179),sat=(0,255),val=(0,255)):
    hsv = cv2.cvtColor(img,cv2.COLOR_BGR2HSV)

    if hue[0] < 0:
        low = (0,sat[0],val[0])
        high = (hue[1],sat[1],val[1])
        cv = cv2.inRange(hsv,np.array(low),np.array(high))

        low = (179+hue[0],sat[0],val[0])
        high = (179,sat[1],val[1])
        cv = cv2.add(cv,cv2.inRange(hsv,np.array(low),np.array(high)))
    else:
        low = (hue[0],sat[0],val[0])
        high = (hue[1],sat[1],val[1])
        cv = cv2.inRange(hsv,np.array(low),np.array(high))
    cv = cv2.erode(cv,None,10)
    return cv
```

```
def arrow(img,pt1,pt2,color):
    ipt1=(int(pt1[0]),int(pt1[1]))
    ipt2=(int(pt2[0]),int(pt2[1]))
    cv2.line(img,ipt1,ipt2,color,2)

def draw_vector(img,pt,direction,color=-1):
    arrow(img,pt,(pt[0]+direction[0],pt[1]+direction[1]),color)

def unit_vector(v):
    (dx,dy) = v
    l=math.sqrt(dx**2 + dy**2)
    return (dx/l, dy/l)

def add_vector(v1,v2):
    return (v1[0]+v2[0],v1[1]+v2[1])

def annotate_marker(img,center,w,text,hue=255,sat=255,val=255):
    cv2.circle(img,center,w/2,(hue,sat,val),2)
    cv2.putText(img,'%s' % (text),(center), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (255,255,255))

def place_tag(img,position,text=None):
    cv2.circle(img,position,5,(255,255,255),2)
    if text:
        cv2.putText(img,text,position,cv2.FONT_HERSHEY_SIMPLEX, 1.0, (255,255,255))

def locate_circle(img,hue=(0,179),sat=(0,255),val=(0,255),sz=(35,50),show=False):
    """Take an HSV bound and locate the markers within the image."""
    objects = threshold(img,hue,sat,val)
    contours,hierarchy = cv2.findContours(objects,cv2.RETR_LIST,cv2.CHAIN_APPROX_SIMPLE)
    # Remove artifacts
    # Filter by area
    contours = filter(lambda c: cv2.contourArea(c) > 3.14*(sz[0]/2.)**2, contours)
    contours = filter(lambda c: cv2.contourArea(c) < 3.14*(sz[1]/2.)**2, contours)
    contours = filter(lambda c: 2*cv2.minEnclosingCircle(c)[1] >= sz[0], contours)
    contours = filter(lambda c: 2*cv2.minEnclosingCircle(c)[1] <= sz[1], contours)

    retval = [ cv2.boundingRect(c) for c in contours ]
    # Draw a circle over the object
    if show:
        for (x,y,w,h) in retval:
            center = (x+w/2,y+h/2)
            avghue = sum(hue)/2
            if avghue < 0: avghue = avghue + 179
            cv2.circle(img,center,w/2,(avghue,sum(sat)/2,255),2)
    return retval
```

```
def locate_markers(img,hue=(0,179),sat=(0,255),val=(0,255),sz=(35,50),show=False):
    """Find all boundary markers in the frame and return the pixel coordinates"""
    markers = locate_circle(img,hue,sat,val,sz,show)
    return markers

def locate_boundary(img,show=False):
    """Find four boundary markers in the frame and return the pixel coordinates"""
    if view == 1:
        markers = locate_markers(img,hue=(-20,60),sat=(0,200),val=(0,60),sz=(17,23),show=True)
    else:
        markers = locate_markers(img,hue=(-20,60),sat=(0,200),val=(0,60),sz=(5,15),show=True)

    if len(markers) != 4:
        print >> sys.stderr, "Lost boundary markers=%d"%(len(markers))
        return
    markers = sorted(markers, key=lambda m: m[0]**2+m[1]**2)
    return markers

def locate_center(img,markers,show=True):
    """Return the center pixel position using the boundary markers"""
    cx=0
    cy=0
    if not markers:
        return
    m = len(markers)
    for (x,y,w,h) in markers:
        cx = cx+(x+w/2.)
        cy = cy+(y+h/2.)
    if show:
        cv2.circle(img,(int(cx/m),int(cy/m)),15,(255,255,255),2)
        place_tag(img,(int(cx/m),int(cy/m)),text="C (%d,%d)" % (int(cx/m),int(cy/m)))
    return (cx/m,cy/m)

def locate_robot(img,show=False):
    # Find the yellow vehicle markers
    markers = locate_markers(img,hue=(20,40),sat=(100,255),val=(60,240),sz=(0.4*inch,1.5*inch),show=True)
    if len(markers) != 2:
        print >> sys.stderr, "Lost robot, markers = %d" % (len(markers))
        return []
    return markers

def velocity_vector(p1,p2,dt):
    # Compute the differences
    if p1 and p2 and dt:
```

```
        dx=(p2[0]-p1[0])
        dy=(p2[1]-p1[1])
        return (dx/dt,dy/dt)
    else:
        return None

def distance_from_center(pos,center):
    (x1,y1) = pos
    (x0,y0) = center
    dx = (x1-x0)
    dy = (y1-y0)
    r = math.sqrt(dx**2 + dy**2)
    return r

def annotate_boundaries(img,markers):
    # Locate first boundary marker
    if not markers:
        return None
    (x,y,w,h) = markers[0]
    origin = (x+w/2.,y+h/2.)
    for (x,y,w,h) in markers:
        center = (x+w/2,y+h/2)
        pos = (center[0]-origin[0],center[1]-origin[1])
        cv2.putText(img,'M: %d,%d,%d' % (pos[0],pos[1],w),(center), cv2.FONT_HERSHEY_SIMPLEX, 1.0,
(255,255,255))
    return origin

def annotate_robot(img,markers,origin):
    robot = []
    for (x,y,w,h) in markers:
        center = (x+w/2,y+h/2)
        pos = (center[0]-origin[0],center[1]-origin[1])
        robot.append(pos)
        cv2.putText(img,'R: %d,%d,%d' %(pos[0],pos[1],w),(center), cv2.FONT_HERSHEY_SIMPLEX, 1.0,
(255,255,255))
    return robot

def annotate_track(img,center,radius):
    if center:
        cv2.circle(img,center,int(radius),(0,0,0),2)

# Functions that work in real dimensions
def calibrateDistance(markers,dist=18.):
    """Calibrate dimension knowing that boundary markers are 18 inches apart"""
    global inches, inch
```

```
    # Markers 0 and 1 should be adjacent
    if not markers or len(markers) != 4:
        return
    (x0,y0,w0,h0) = markers[0]
    (x1,y1,w1,h1) = markers[1]
    inches = max((x1-x0),(y1-y0))/dist
    inch = inches
    #print "Calibration: %g pixels/inch" % (inches)

if __name__ == '__main__':
    import os, sys, string
    # Get the argument
    fileName, fileExtension = None, None
    if len(sys.argv) > 1:
        h = sys.argv[1]
        fileName, fileExtension = os.path.splitext(h)
        fileExtension = string.lower(fileExtension)
    if fileExtension in ['.jpg']:
        process_image(h)
    elif fileExtension in ['.mov']:
        process_video(h)
    else:
        process_video(0)
```

APPENDIX F – PyMC TUTORIAL

Below is a tutorial on using the PyMC approach for radiation effects. The content below is intended to be displayed as an IPython Notebook and contains all the code necessary to execute the tutorial.

Bayesian Inference for Radiation Effect using Python: A Tutorial

Zachary Diggins

zdiggins@gmail.com

Contents

1. Introduction
2. Chapter 1: Inference on Normally Distributed Data
3. Chapter 2: Modeling Lot-to-Lot Variability with a Hierarchical Model
4. References

Introduction

The purpose of this document is to demonstrate how to perform Bayesian inference in Python, and to demonstrate the numerous advantages of the approach. Many tutorials and resources have been extremely helpful in the construction of this document. The goal of this tutorial is to describe in detail how the technique works for engineers and scientists who rarely work with statistics.

In this document I generate all the data before I model it. The goal of this notebook is to test the modeling methods, so knowing the "correct answer" ensures that we are building the models and using the tools correctly.

The examples in this notebook are centered around my area of research, the effects of radiation on electronics. Domain specific knowledge is very important in choosing appropriate model forms, but the analysis techniques demonstrated here can be applied to a wide range of reliability or statistical inference problems. This notebook models the impact of radiation on a bipolar junction transistor's (BJT) gain, which is the ratio of base current to the collector current in the forward active region, commonly referred to as Beta. This parameter usually decreases with increasing total-ionization dose due to radiation (TID).

We will first model the part-to-part variation in Beta's post-irradiation value. We will expand this model to include variations between different manufacturing lots of the transistor. We will then expand the analysis again to include different types of transistors as well as different lots and part-to-part variation using hierarchical modeling.

Next, we explore longitudinal analysis, modeling how a parameter changes over time (or in this case increasing radiation dose). We will compare linear regression and random effects models to Bayesian analysis. Finally, we will extend this analysis a simple system, a voltage regulator consisting of multiple transistors of the same type as our model.

Import Dependencies

I have chosen to import all the dependencies at the start of the document. A description of each package is included below:

Numpy: Provides powerful numerical computation capabilities to python.

Pandas: Used for data organization. Provides dataframe support that is easy to plot and index.

Scipy: Scientific computation package. Provides statistical functions.

Matplotlib: Provides plotting capabilities, similar to Matlab.

Seaborn: Plotting supplement to Matplotlib that improves graphics quality and allows for quickly plotting statistical distributions.

Pymc: Provides model building and Markov Chain Monte Carlo capabilities.

In [3]:

```
%matplotlib inline
import numpy as np
import pandas as pd
import scipy as sp
import matplotlib.pyplot as plt
import seaborn as sns
import pymc3
import daft
```

Chapter 1 - Inference on Normally Distributed Data

Generate Data: Transistor gain after irradiation

The first step is to generate some data to work with. We will make up a transistor that has a post-irradiation Beta value of 75, with a standard deviation of 5. Let's generate and print the data. We use the Scipy Stats package random variates sampling function to generate 10 samples and save it in the "transistor1_data" array.

In [4]:

```
beta_postrad_mean = 75.0
beta_postrad_std_dev = 5.0
transistor1_data =
sp.stats.norm.rvs(beta_postrad_mean,beta_postrad_std_dev,10)
print(transistor1_data)
[ 72.13163412  84.4397247   73.56069485  74.82876467  71.43288644
  71.92564894  73.97212881  72.41094776  74.83938552  80.85544353]
```

Seaborn has some very nice distribution plotting functions. Here we use "distplot" to plot the generated values as a histogram, overlaying the data with a normal distribution fit from the Scipy package.

In [5]:

```
sns.distplot(transistor1_data, fit=sp.stats.norm, bins=5, kde=False)
```

Out[5]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff634093bd0>
```

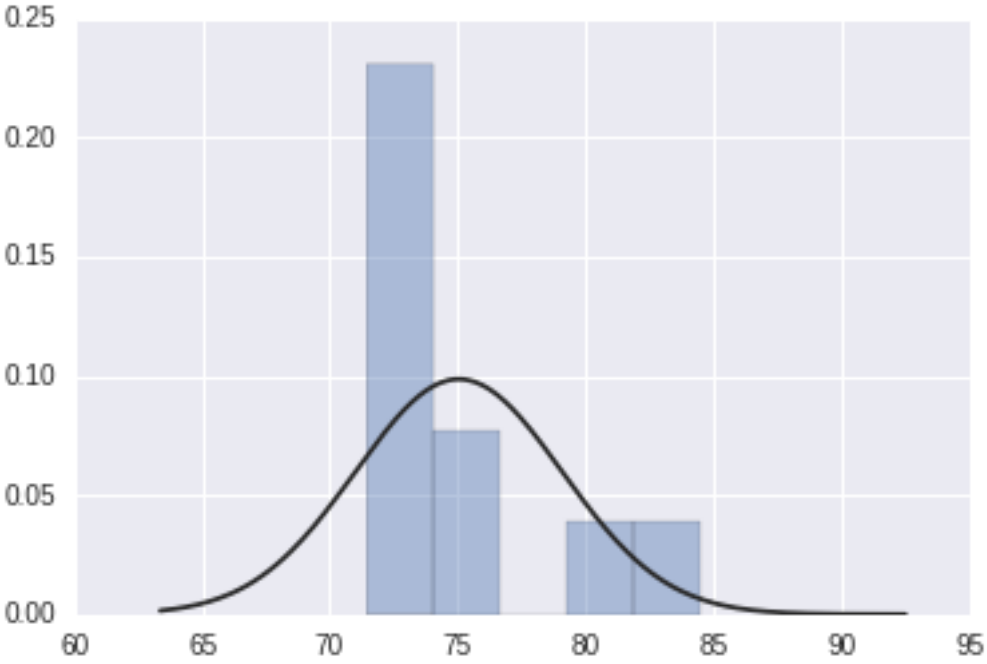

Figure 60. Visualization of generated data.

Let's print those normal distribution values resulting from the Scipy fit using least-squared methods. They should be close to 75 and 5 (they will be slightly different since we have a relatively small number of samples.

In [6]:

```
least_sq_mean, least_sq_sd = sp.stats.norm.fit(transistor1_data)
print(least_sq_mean, least_sq_sd)
(75.039725933553854, 4.0455991483175788)
```

Fit using MCMC methods

Everything we have done up to this point is very basic. Let's start using Bayesian analysis and fit the 10 data points with a normal model using Markov Chain Monte Carlo.

What does a Bayesian model of this data look like? We need to specify a distribution for the parameter we are observing. Beta will follow a normal distribution. A normal distribution is parameterized with 2 parameters, a location parameter (the mean) and a scale parameter (the standard deviation or precision). So our model will look like using equations:

$$\beta = Normal(\mu, \sigma)$$

Let's draw the Directed Acyclic Graph representation using the Daft Package:

In [7]:

```
from matplotlib import rc
rc("font", family="serif", size=16)
rc("text", usetex=False)
pgm = daft.PGM([4, 2], origin=[1.15, 0.65])
pgm.add_node(daft.Node("mean", r"$\mu$", 2, 2, aspect=1.5))
pgm.add_node(daft.Node("std. dev.", r"$\sigma$", 4, 2, aspect=3))
pgm.add_node(daft.Node("beta", r"$\beta$", 3, 1, aspect=2.4, observed=True))
pgm.add_edge("mean", "beta")
pgm.add_edge("std. dev.", "beta")
pgm.render()
```

Out[7]:

```
<matplotlib.axes._axes.Axes at 0x7ff5fb67b590>
```

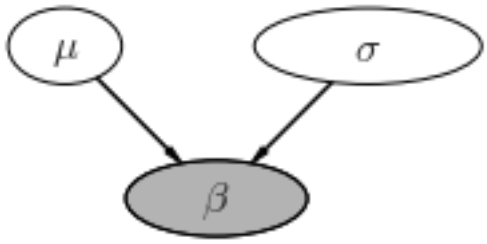

Figure 61. Basic Bayesian network for normally distributed transistor Beta.

We need priors for the mean and standard deviation values. We want our model to be dominated by the data so we can use weakly informative or flat priors.

For the mean value we will use a uniform prior over the parameter space. Beta will not be larger than 200 for our transistor and can't be negative, so let's use a uniform distribution with a range of 0 to 1. For the std. dev. parameter we can use a half normal distribution with a large standard deviations, which is a normal distribution centered around 0, restricted to non-negative values.

Notice in the plots that the value for the prior is positive for each and very low (<0.01 probability for each entry).

In [8]:

```
x = np.linspace(-200,200,100)
y_mean = sp.stats.uniform.pdf(x,0,200)
y_sd = sp.stats.halfnorm.pdf(x,0,15)
f, (ax1, ax2) = plt.subplots(1, 2, sharey=False)
ax1.plot(x,y_mean,label='mean prior')
ax1.set_title('Mean Prior')
ax2.plot(x,y_sd,label='std. dev. prior')
ax2.set_title('Std. Dev. Prior')
ax1.set_ylabel('Probability')
ax1.set_xlabel('Mean')
ax2.set_xlabel('Std. Dev.')
```

Out[8]:

```
<matplotlib.text.Text at 0x7ff8062eedd0>
```

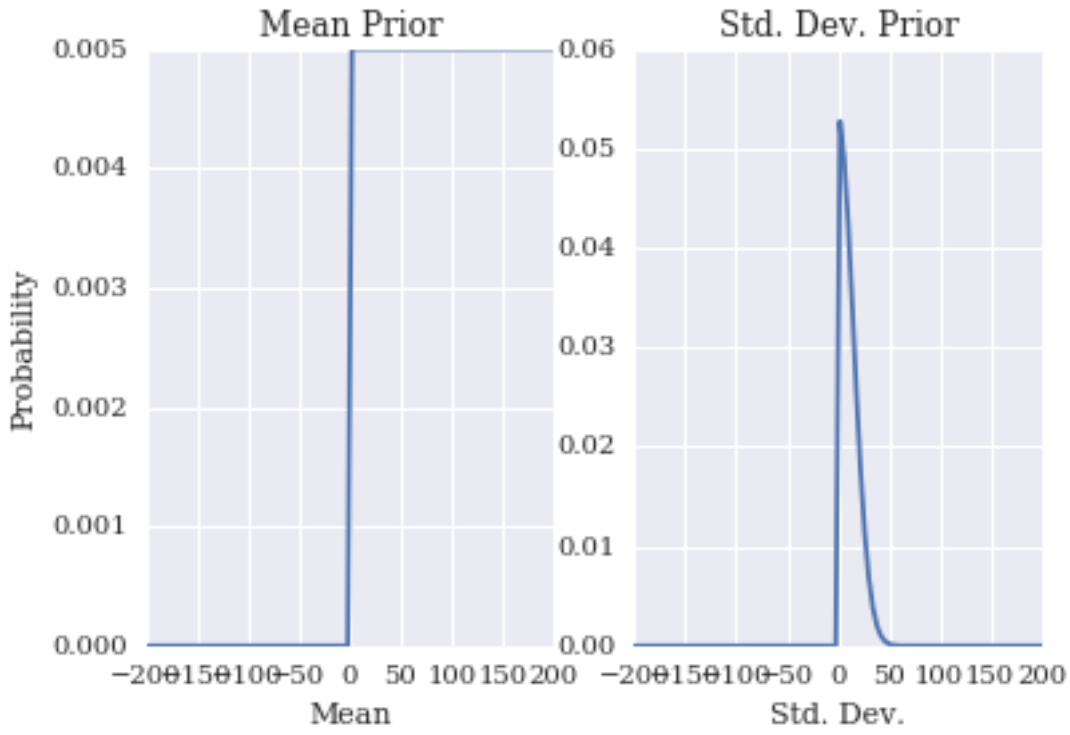

Figure 62. Comparison of step function prior versus std. dev. prior.

We are now ready to specify our Bayesian model using Pymc3 notation.

First we create a new model, and name it "basic model".

Next we specify the model's priors as described above.

Finally, we specify the transistor_beta parameter as we described above in Output 6. Notice we add the observed tag and insert our data into the model here. This is the likelihood distribution. The combination of the likelihood and priors produces the posterior distribution, which is what we use for inference.

In [9]:

```
basic_model = pymc3.Model()

with basic_model:

    # Non-informative priors
    mean=pymc3.Uniform('mean',lower=0, upper=200)
    standard_deviation = pymc3.HalfNormal('standard_deviation',sd=15)

    # Likelihood (sampling distribution) of observations
    transitor_beta = pymc3.Normal('transistor_beta', mu=mean,
sd=standard_deviation, observed=transistor1_data)
    beta_samples = pymc3.Normal('beta_samples', mu=mean, sd=
standard_deviation)
```

Now that we have the model specified, we need to sample the model using Pymc3's sampling methods. We will generate 10,000 samples of the posterior of our model. There are many options in this step, such as burn-in, sampling algorithms, number of sample, how to store the samples. For now we will use the recommended sampler, the No-U-Turn_Sampler (NUTS) and use a Pymc3 optimization function to find the starting point [29].

Note: We can do the whole Markov Chain Monte Carlo process for any model we can think of in 2 lines! This is why Bayesian analysis is now practical for specialists who aren't experts in the details of Bayesian algorithms, described in more detail in the end of the tutorial.

In [10]:

```
with basic_model:

    # obtain starting values via MAP
    start = pymc3.find_MAP(fmin=sp.optimize.fmin_powell)

    # draw 2000 posterior samples
    trace = pymc3.sample(10000, start=start)
Assigned NUTS to mean_interval
Assigned NUTS to standard_deviation_log
Assigned NUTS to beta_samples
 [-----------------100%-----------------] 10000 of 10000 complete in 306.5
sec
```

The results of the sampling are stored in a data structure in the variable trace. Let's use Pymc's summary and plotting functions to display the results below.

Notice a few things, the mean is close to 75 and the std. dev. is close to 5. Looks like our model worked.

In [11]:

```
pymc3.summary(trace)
pymc3.traceplot(trace)
mean_interval:

  Mean             SD               MC Error         95% HPD interval
  ------------------------------------------------------------------

  -0.498           0.042            0.000            [-0.585, -0.420]

  Posterior quantiles:
  2.5            25             50             75             97.5
  |--------------|==============|==============|--------------|

  -0.583         -0.522         -0.497         -0.473         -0.417


standard_deviation_log:

  Mean             SD               MC Error         95% HPD interval
  ------------------------------------------------------------------

  1.700            0.276            0.008            [1.223, 2.254]

  Posterior quantiles:
  2.5            25             50             75             97.5
  |--------------|==============|==============|--------------|

  1.248          1.499          1.676          1.868          2.318


beta_samples:

  Mean             SD               MC Error         95% HPD interval
  ------------------------------------------------------------------

  75.622           6.656            0.073            [61.948, 87.898]

  Posterior quantiles:
  2.5            25             50             75             97.5
  |--------------|==============|==============|--------------|

  62.586         71.597         75.626         79.656         88.624
```

```
mean:

  Mean             SD               MC Error         95% HPD interval
  ------------------------------------------------------------------

  75.601           1.991            0.020            [71.552, 79.308]

  Posterior quantiles:
  2.5              25               50               75               97.5
  |--------------|================|================|--------------|

  71.641           74.459           75.632           76.785           79.432


standard_deviation:

  Mean             SD               MC Error         95% HPD interval
  ------------------------------------------------------------------

  5.702            1.793            0.051            [3.127, 9.051]

  Posterior quantiles:
  2.5              25               50               75               97.5
  |--------------|================|================|--------------|

  3.483            4.478            5.342            6.473            10.160
```

Out[11]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7ff8043be450>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff802f5e090>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7ff8030f6bd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff8030fd750>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7ff803fab3d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff8030ac890>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7ff803cfbbd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff803ff61d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7ff803eceb90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff803dce190>]],
dtype=object)
```

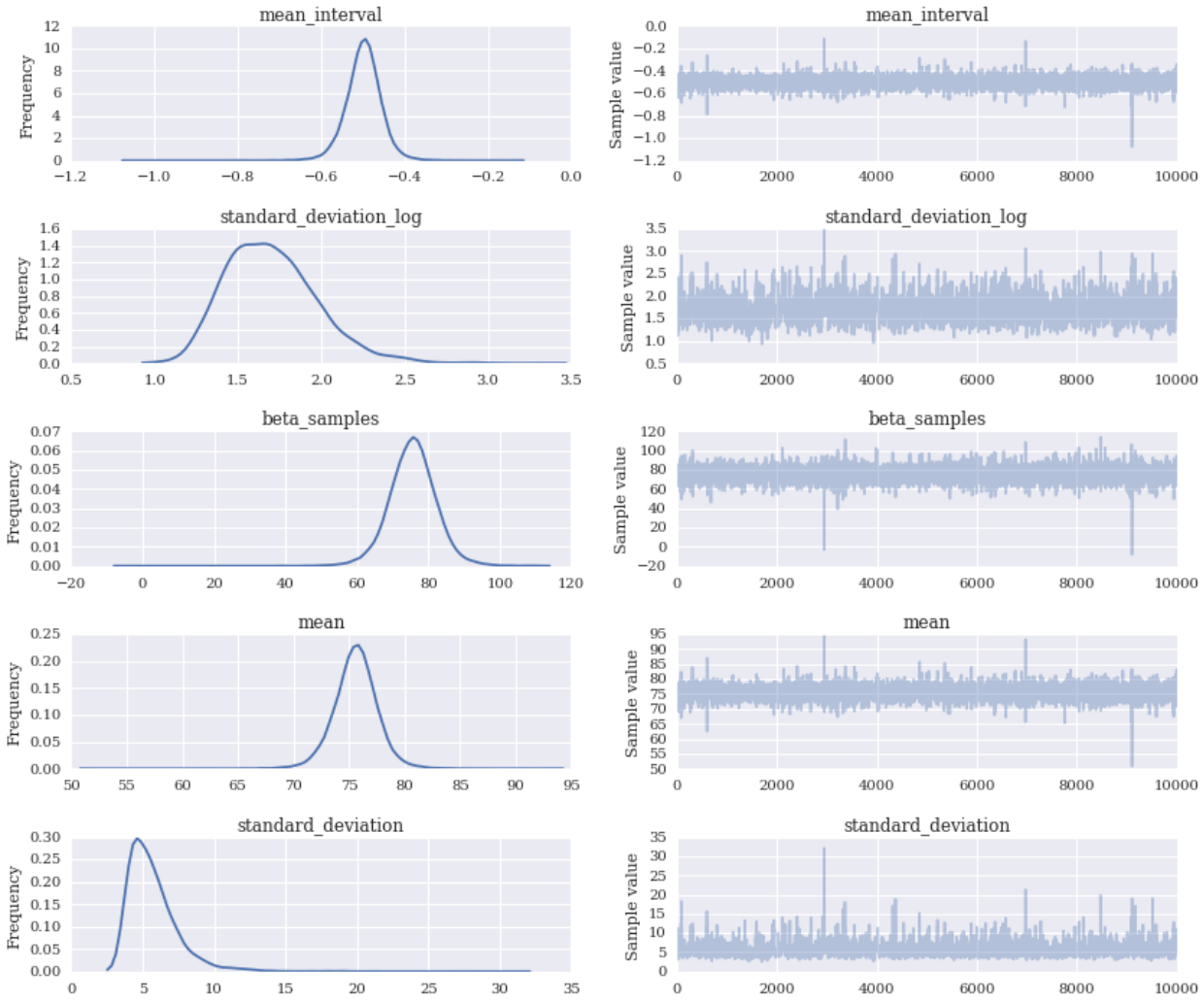

Figure 63. Distributions and chain plots for modeled parameters in the basic_model.

Comparison between Bayesian and Frequentist methods:

Let's compare the mean, sd, and lower 10th percentile for the least-squares fit and the Bayesian fit. Both methods reveal comparable results. Our standard deviation is slightly off for the Bayesian analysis, which could be improved with a better prior (ours is probably too broad) or with appropriate burn-in or thinking. It is being biased by some very large values visible in the summary, which is a result of the relatively low sample count.

In [12]:

```
least_sq_tenth_percentile = sp.stats.norm.ppf(.1,least_sq_mean, least_sq_sd)
print("Least-squares:")
print(least_sq_mean, least_sq_sd, least_sq_tenth_percentile)
print("Bayesian analysis:")
print(np.mean(trace['mean']),np.mean(trace['standard_deviation']),np.sort(tra
ce['beta_samples'])[1000])
Least-squares:
(75.603516710915258, 4.2040204665486733, 70.21584770042827)
Bayesian analysis:
(75.601049252196219, 5.7018737169843838, 67.766081176503405)
```

Chapter 2 - Modeling Lot-to-Lot Variability with a Hierarchical Model.

Generate data for multiple lots.

Let's use the same approach as before, generate data, model it, and compare back to the original values. For this section a "part" is a single transistor. A "lot" is a group of transistors all produced at the same time that are expected to follow a normal distribution. We want to now add the complexity of lot-to-lot variation, a variation in the mean of each lot based on some value. We need to generate a value that we will shift the mean value of each simulated lot by, in order to introduce lot-to-lot variation in the mean of the lot. A reasonable assumption is that all the lots will be centered around some mean with a giving variation, so the variation for each lot will be consistent, but the center of each lot will vary. Let's assume that the mean of each lot is offset by a random sample from a normal distribution with a mean of 0 (so it can be offset positive or negative) and a standard deviation of 10. This will keep the mean of the overall dataset in the same location while producing lots with their own skewed means.

We need to do something more eloquent with data handling for this larger data sets, so we will use a Pandas dataframe. Below is the code to generate the data and the resulting dataframe displayed.

In [13]:

```
process_mean = 75
lot_to_lot_sd = 10
part_to_part_sd = 5

#draw lot center
lot_offset = sp.stats.norm.rvs(0,lot_to_lot_sd,10)

temp = []
columns = []

for i in range(0,10):
    beta=np.zeros(10)
    beta = sp.stats.norm.rvs(process_mean + lot_offset[i], part_to_part_sd,
10)
    temp.append(beta)
    columns.append("Lot "+str(i+1))

transistorData = pd.DataFrame(temp, index=columns).transpose()
pd.set_option('display.precision', 2)
print(transistorData)
print(lot_offset)
```

```
   Lot 1  Lot 2  Lot 3  Lot 4  Lot 5  Lot 6  Lot 7  Lot 8  Lot 9  Lot 10
0  79.41  81.14  92.95  75.07  70.33  81.68  63.78  68.27  68.03   90.81
1  71.11  62.58  81.85  75.48  78.56  75.99  72.98  65.61  69.34   85.73
2  72.77  83.38  92.42  72.02  74.83  80.00  68.39  74.43  67.27   86.49
3  79.56  70.59  97.98  76.21  68.58  75.38  68.12  65.65  72.50   93.99
4  76.49  77.55  85.72  56.75  71.49  79.24  67.71  73.48  70.05   86.25
5  66.59  73.28  91.34  73.05  75.71  79.06  74.01  69.91  76.78   93.81
6  73.87  79.33  87.61  74.79  77.83  77.92  71.42  70.13  70.23   82.70
7  74.36  75.49  83.66  71.51  74.38  63.70  78.32  69.14  80.56   78.26
8  74.95  72.09  94.44  69.62  73.99  72.23  77.98  63.96  67.10   82.38
9  73.69  70.94  95.01  72.18  70.84  73.64  71.98  64.21  77.49   89.67
[ -1.06641884   0.47448023  17.60927772  -0.48260833  -1.47066097
   1.94082845  -6.03870594  -5.63501337  -4.09269033  12.76469513]
```

Now we have a good dataset to work with, lets plot all the distributions using Seaborn.

In [14]:

```
for column in transistorData.columns:
    sns.distplot(transistorData[column],bins=20, rug=True, hist=True,
fit=sp.stats.norm, kde=False)
    sns.axlabel("Transistor Post-rad Beta", "Probability")
```

```
#sns.distplot(transistorData['Lot 2'],bins=20, rug=True, hist=False,
fit=stats.norm, kde=False)
```

We will first analyze the dataset by performing a least-squares normal distribution fit on each lot and then performing the fit on the lot means:

In [15]:

```
temp = []
for lot in transistorData.columns:
    temp.append(sp.stats.norm.fit(transistorData[lot]))

lot_fits = pd.DataFrame(temp,columns=['Mean','Std. Dev.'])
print(lot_fits)

lot_to_lot_mean = sp.stats.norm.fit(lot_fits['Mean'])
print(lot_to_lot_mean)
lot_fits.plot(style='.')
    Mean  Std. Dev.
0  74.28       3.62
1  74.63       5.78
2  90.30       5.05
3  71.67       5.34
4  73.65       3.12
5  75.88       4.93
6  71.47       4.39
7  68.48       3.47
8  71.94       4.50
9  87.01       4.85
(75.931178457020081, 6.6951041911414277)
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff8010b7e50>
```

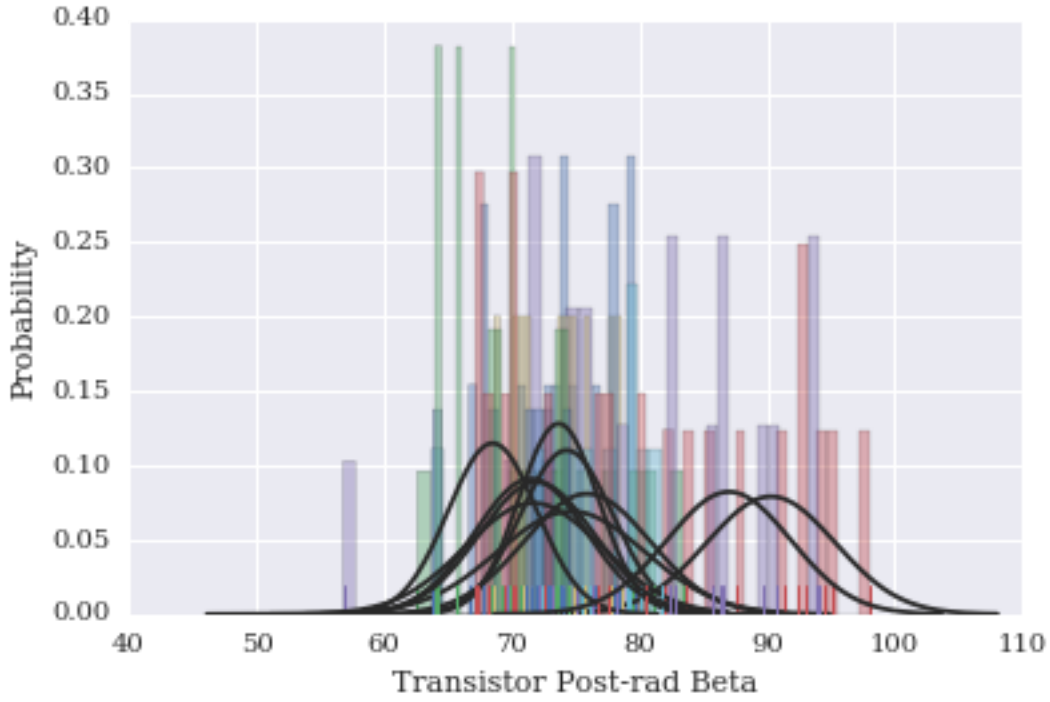

Figure 64. Lot-to-lot variability plot.

These results match with our data generation model well. Now let's do the analysis using Bayesian analysis. First we need to write the equations for our new lot-to-lot variation model and draw the DAG. This model is considered hierarchical and introduces plate notation.

$$\beta = Normal(\mu_{part,i}, \sigma_{part,i})$$

$$\mu_i = Normal(\mu_{lot}, \sigma_{lot})$$

In [16]:

```
pgm = daft.PGM([5, 5], origin=[0, 0])

pgm.add_node(daft.Node("mu_lot", r"$\mu^{lot}$", 1.5, 4.5, scale=1.5,))
pgm.add_node(daft.Node("sigma_lot", r"$\sigma^{lot}$", 2.5, 4.5,scale=1.5))
pgm.add_node(daft.Node("mu_part", r"$\mu^{part}_i$", 2, 3.5,scale=1.5))
```

```
pgm.add_node(daft.Node("sigma_part", r"$\sigma^{part}_i$", 3, 3.5,scale=1.5))
pgm.add_node(daft.Node("beta", r"$\beta_i$", 2.5, 2,observed=True,scale=1.5))

pgm.add_edge("mu_lot", "mu_part")
pgm.add_edge("sigma_lot", "mu_part")
pgm.add_edge("sigma_part", "beta")
pgm.add_edge("mu_part", "beta")

# And a plate.
pgm.add_plate(daft.Plate([1.5, 1, 2, 3], label=r"lot $i$",
    shift=-0.1))

pgm.render()
```

Out[16]:

```
<matplotlib.axes._axes.Axes at 0x7ff800f20bd0>
```

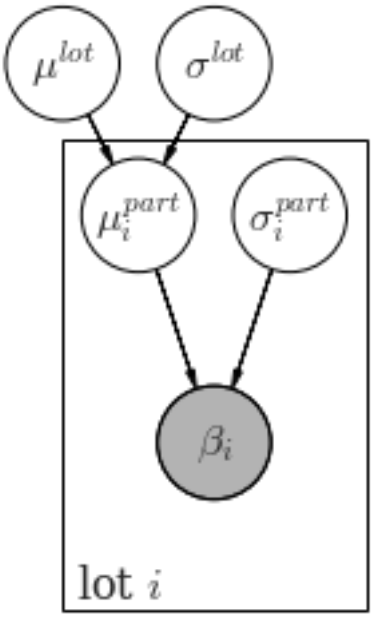

Figure 65. Lot-to-Lot variability model.

In [17]:

```
ten_lot_model = pymc3.Model()

with ten_lot_model:

    #hyperpriors for group parameters
```

```
    mu_lot=pymc3.Uniform('mu_lot',lower=0, upper=200)
    sigma_lot=pymc3.HalfNormal('sigma_lot',sd=20)

    # Non-informative priors
    sigma_part = pymc3.HalfNormal('sigma_part',sd=20,shape=10)
    mu_part = pymc3.Normal('mu_part',mu=mu_lot,sd=sigma_lot,shape=10)

    # Likelihood (sampling distribution) of observations
    transitor_beta = pymc3.Normal('transistor_beta', mu=mu_part,
sd=sigma_part, observed=True)
    beta_samples = pymc3.Normal('beta_samples', mu=mu_part, sd=sigma_part,
shape=10)
```

In [18]:

```
with ten_lot_model:

    # obtain starting values via MAP
    pymc3.start = pymc3.find_MAP(fmin=sp.optimize.fmin_powell)

    # draw 2000 posterior samples
    trace = pymc3.sample(1000, start=pymc3.start)
Assigned NUTS to mu_lot_interval
Assigned NUTS to sigma_lot_log
Assigned NUTS to sigma_part_log
Assigned NUTS to mu_part
Assigned NUTS to beta_samples
```

In [ ]:

```
pymc3.summary(pymc3.trace)
pymc3.traceplot(pymc3.trace)
```

In [ ]:

These models generate the posterior distributions used for further analysis. See the following section for more tutorials and resources on this type of analysis.

References and Helpful Resources

Bayesian Data Analysis using Pymc3: Thomas Wiecki's blog post on video with accompanying notebook describing how to use Pymc3 for a linear model.

http://twiecki.github.io/blog/2013/12/12/bayesian-data-analysis-pymc3/

REFERENCES

[1] R. Murphy, *Disaster Robotics*. MIT Press, 2014.

[2] P. C. Bennett and L. D. Posey, "RHOBOT: Radiation Hardened Robotics," Sandia National Laboratories, Unlimited Release SAND97-2405, Oct. 1997.

[3] H. Laurent P., "Robotics and Radiation Hardening in the Nuclear Industy," University of Florida.

[4] American Nuclear Society Special Committee, "ANS Committee Report - Fukushima Daiichi," Mar. 2012.

[5] K. Nagatani, S. Kiribayashi, Y. Okada, K. Otake, K. Yoshida, S. Tadokoro, T. Nishimura, T. Yoshida, E. Koyanagi, M. Fukushima, and S. Kawatsuma, "Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots: Emergency Response to the Fukushima Nuclear Accident using Rescue Robots," *J. Field Robot.*, vol. 30, no. 1, pp. 44–63, Jan. 2013.

[6] E. Strickland, "Meet the Robots of Fukushima Daiichi," *IEEE Spectrum*, 28-Feb-2014.

[7] E. Strickland, "Dismantling Fukushima: The World's Toughest Demolition Project," *IEEE Spectrum*, 28-Feb-2014.

[8] J. McCurry, "Fukushima robot stranded after stalling inside reactor," *The Guardian*.

[9] D. M. Fleetwood, "Total Ionizing Dose Effects in MOS and Low-Dose-Rate-Sensitive Linear-Bipolar Devices," *IEEE Trans. Nucl. Sci.*, vol. 60, no. 3, pp. 1706–1730, Jun. 2013.

[10] J. R. Schwank, "Total Dose Effects in MOS Devices," Proc. NSREC Short Course, 1993.

[11] US Department of Defense, "Military standard. Test methods and procedures for microelectronics," Mil-Std- 833D, 1991.

[12] American Society for Testing and Materials, "Methods for application of ionization chambers to assess the low energy gamma component of Cobalt-60 irradiators used in radiation hardness testing of silicon electronic devices," ASTM E1250-88, 1988.

[13] American Society for Testing and Materials, "Practice for minimizing dosimetry errors in radiation hardness testing of silicon electronic devices using Cobalt-60," ASTM E1249-88, 1988.

[14] J. R. Schwank, M. R. Shaneyfelt, and P. E. Dodd, "Radiation Hardness Assurance Testing of Microelectronic Devices and Integrated Circuits: Radiation Environments, Physical Mechanisms, and Foundations for Hardness Assurance," *IEEE Trans. Nucl. Sci.*, vol. 60, no. 3, pp. 2074–2100, Jun. 2013.

[15] http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20080008301.pdf, "NASA Systems Engineering Handbook." 2007.

[16] M. J. Daigle, "A qualitative event-based approach to fault diagnosis of hybrid systems," Vanderbilt University, 2008.

[17] R. Ladbury, J. L. Gorelick, M. A. Xapsos, T. O'Connor, and S. Demosthenes, "A Bayesian treatment of risk for radiation hardness assurance," in *RADECS Proc.*, 2005.

[18] R. Ladbury, J. L. Gorelick, and S. S. McClure, "Statistical Model Selection for TID Hardness Assurance," *IEEE Trans. Nucl. Sci.*, vol. 56, no. 6, pp. 3354–3360, Dec. 2009.

[19] R. Ladbury and B. Triggs, "A Bayesian Approach for Total Ionizing Dose Hardness Assurance," *IEEE Trans. Nucl. Sci.*, vol. 58, no. 6, pp. 3004–3010, Dec. 2011.

[20] R. L. Ladbury and M. J. Campola, "Bayesian Methods for Bounding Single-Event Related Risk in Low-Cost Satellite Missions," *IEEE Trans. Nucl. Sci.*, vol. 60, no. 6, pp. 4464–4469, Dec. 2013.

[21] Z. Ghahramani, "An Introduction to Hidden Markov Models and Bayesian Networks," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 15, no. 1, pp. 9–42, 2001.

[22] D. J. Spiegelhalter, A. P. Dawid, S. L. Laauritzen, and R. G. Cowell, "Bayesian Analysis in Expert Systems," *Stat. Sci*, vol. 8, no. 3, pp. 219--247, 1993.

[23] G. Casella, *Statistical Inference*. Thomson Press, 2008.

[24] Lunn, David, Chris Jackson, Nicky Best, Andrew Thomas, and David Spiegelhalter, *The BUGS book: A practical introduction to Bayesian analysis.* CRC press, 2012.

[25] Gelman, Andrew, John B. Carlin, Hal S. Stern, and Donald B. Rubin, *Bayesian data aalysis*. Chapman & Hall/CRC, 2014.

[26] Carlin, Bradley P., and Thomas A. Louis, *Bayesian Methods for Data Analysis*. CRC Press, 2011.

[27] K. Murphy, "The Bayes Net Toolbox for Matlab," *Comput. Sci. Stat.*, vol. 33, no. 2, pp. 1024–1034, 2001.

[28] M. J. Druzdzel, "SMILE: Structural Modeling, Inference, and Learning Engine and GeNIe: a development environment for graphical decision-theoretic models," in *Aaai/Iaai*, 1999, pp. 902–903.

[29] A. Patil, D. Huard, and C. J. Fonnesbeck, "PyMC: Bayesian stochastic modeling in Python," *J. Stat. Softw.*, vol. 35, no. 4, p. 1, 2010.

[30] Spiegelhalster, D., "OpenBUGS user manual, version 3.0. 2." MRC Biostatistics Unit, Cambridge, 2007.

[31] D. J. Lunn, A. Thomas, N. Best, and D. Spiegelhalter, "WinBUGS-a Bayesian modeling framework: concepts, structure, and extensibility," *Stat. Comput.*, vol. 10, no. 4, pp. 325–337, 2000.

[32] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, "An introduction to MCMC for machine learning," *Mach. Learn.*, vol. 50, no. 1–2, pp. 5–43, 2003.

[33] H. Dezfuli, D. Kelly, C. Smith, K. Vedros, and W. Galyean, "Bayesian inference for NASA probabilistic risk and reliability analysis," 2009.

[34] M. Decréton, "Position sensing in nuclear remote operation," *Measurement*, vol. 15, no. 1, pp. 43–51, 1995.

[35] S. Kawatsuma, M. Fukushima, and T. Okada, "Emergency response by robots to Fukushima-Daiichi accident: summary and lessons learned," *Ind. Robot Int. J.*, vol. 39, no. 5, pp. 428–435, 2012.

[36] J. Schwank, "Total dose effects in MOS devices," in *IEEE NSREC Short Course*, 2002, pp. 1–123.

[37] H. Hughes and J. Benedetto, "Radiation effects and hardening of MOS technology: devices and circuits," *IEEE Trans. Nucl. Sci.*, vol. 50, no. 3, pp. 500–521, 2003.

[38] P. Bennett and L. Posey, "RHOBOT: Radiation hardened robotics," *SAND97-2405 Sandia Natl. Lab. Albuq. USA*, 1997.

[39] H. Igarashi, K. Kon, and F. Matsuno, "Evaluation of sensors for mobile robots based on irradiation experiment," in *System Integration (SII), 2012 IEEE/SICE International Symposium on*, 2012, pp. 517–522.

[40] J. Tan, B. Buttgen, and A. J. Theuwissen, "Analyzing the radiation degradation of 4-transistor deep submicron technology CMOS image sensors," *IEEE Sens. J.*, vol. 12, no. 6, pp. 2278–2286, 2012.

[41] L. A. Tambara, F. L. Kastensmidt, E. C. Junior, O. L. Gonçalez, T. R. Balen, P. C. de Aguirre, I. Arruego, and M. S. Lubaszewski, "TID in a Mixed-Signal System-on-Chip: Analog Components Analysis and Clock Frequency Influence in Propagation-Delay Degradation," in *2012 IEEE Radiation Effects Data Workshop*, 2012.

[42] R. Joshi, R. Daniels, M. Shoga, and M. Gauthier, "Radiation hardness evaluation of a class V 32-bit floating-point digital signal processor," in *IEEE Radiation Effects Data Workshop, 2005.*, 2005, pp. 70–78.

[43] R. W. Kingsbury, F. H. Schmidt, K. Cahoy, D. A. Sklair, W. J. Blackwell, I. Osarentin, and R. S. Legge Jr, "TID tolerance of popular cubesat components," 2013.

[44] J. Howard, M. A. Carts, R. Stattel, C. E. Rogers, T. L. Irwin, C. Dunsmore, J. A. Sciarini, and K. A. LaBel, "Total dose and single event effects testing of the Intel Pentium III (P3) and

AMD K7 microprocessors," in *Radiation Effects Data Workshop, 2001 IEEE*, 2001, pp. 38–47.

[45] J. Schwank, P. Winokur, P. McWhorter, F. Sexton, P. Dressendorfer, and D. Turpin, "Physical mechanisms contributing to device'rebound'," *IEEE Trans. Nucl. Sci.*, vol. 31, pp. 1434–1438, 1984.

[46] F. W. Sexton and J. R. Schwank, "Correlation of radiation effects in transistors and integrated circuits," *IEEE Trans. Nucl. Sci.*, vol. 32, no. 6, pp. 3975–3981, 1985.

[47] J. Schwank, F. Sexton, D. Fleetwood, M. Shaneyfelt, K. Hughes, and M. Rodgers, "Strategies for lot acceptance testing using CMOS transistors and ICs," *IEEE Trans. Nucl. Sci.*, vol. 36, no. 6, pp. 1971–1980, 1989.

[48] S. Rezgui, E. P. Wilcox, P. Lee, M. A. Carts, K. LaBel, V. Nguyen, N. Telecco, and J. McCollum, "Investigation of low dose rate and bias conditions on the total dose tolerance of a CMOS flash-based FPGA," *IEEE Trans. Nucl. Sci.*, vol. 1, no. 59, pp. 134–143, 2012.

[49] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott, "Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling," in *High-Performance Computer Architecture, 2002. Proceedings. Eighth International Symposium on*, 2002, pp. 29–40.

[50] ATMEL, "ATmega48PA/88PA/168PA/328P Datasheet." .

[51] H. Barnaby, "Total-ionizing-dose effects in modern CMOS technologies," *IEEE Trans. Nucl. Sci.*, vol. 53, no. 6, pp. 3103–3121, 2006.

[52] Microchip, "PIC16F631/677/685/687/689/690 Data Sheet." .

[53] Z. Diggins, N. Mahadevan, D. Herbison, E. Barth, and A. Witulski, "Impact of gamma radiation on range finding sensor performance," in *SENSORS, 2013 IEEE*, 2013, pp. 1–4.

[54] P. Gerrish, E. Herrmann, L. Tyler, and K. Walsh, "Challenges and constraints in designing implantable medical ICs," *IEEE Trans. Device Mater. Reliab.*, vol. 5, no. 3, pp. 435–444, 2005.

[55] G. J. Schlenvogt, H. J. Barnaby, J. Wilkinson, S. Morrison, and L. Tyler, "Simulation of TID effects in a high voltage ring oscillator," *IEEE Trans. Nucl. Sci.*, vol. 60, no. 6, pp. 4547–4554, 2013.

[56] ATMEL, "AVR998 Guide to IEC60730 Class B compliance with AVR microcontrollers." .

[57] M. J. Druzdzel, "SMILE: Structural Modeling, Inference, and Learning Engine and GeNIe: a development environment for graphical decision-theoretic models," in *AAAI/IAAI*, 1999, pp. 902–903.

[58] Z. Diggins, N. Mahadevan, D. Herbison, G. Karsai, E. Barth, R. Reed, R. Schrimpf, R. Weller, M. Alles, and A. Witulski, "Range-Finding Sensor Degradation in Gamma Radiation Environments," *IEEE Sens. J.*, pp. 1–1, 2014.

[59] Z. J. Diggins, N. Mahadevan, D. Herbison, G. Karsai, B. D. Sierawski, E. Barth, E. B. Pitt, R. A. Reed, R. D. Schrimpf, R. A. Weller, M. L. Alles, and A. Witulski, "Total-Ionizing-Dose Induced Timing Window Violations in CMOS Microcontrollers," *IEEE Trans. Nucl. Sci.*, vol. 61, no. 6, pp. 2979–2984, Dec. 2014.

[60] P. C. Adell, R. D. Schrimpf, W. T. Holman, J. L. Todd, S. Caveriviere, R. R. Cizmarik, and K. F. Galloway, "Total dose effects in a linear Voltage regulator," *IEEE Trans. Nucl. Sci.*, vol. 51, no. 6, pp. 3816–3821, Dec. 2004.

[61] D. M. Fleetwood and Eisen, H. A., "Total-Dose Radiation Hardness Assurance," *IEEE Trans. Nucl. Sci.*, vol. 50, no. No. 3, Jun. 2003.

[62] A. E. Gelfand, S. E. Hills, A. Racine-Poon, and A. F. M. Smith, "Illustration of Bayesian Inference in Normal Data Models Using Gibbs Sampling," *J. Am. Stat. Assoc.*, vol. 85, no. 412, p. 972, Dec. 1990.

[63] M. R. Shaneyfelt, J. R. Schwank, P. E. Dodd, and J. A. Felix, "Total Ionizing Dose and Single Event Effects Hardness Assurance Qualification Issues for Microelectronics," *IEEE Trans. Nucl. Sci.*, vol. 55, no. 4, pp. 1926–1946, Aug. 2008.

[64] Z. J. Diggins, N. Mahadevan, G. Karsai, D. Herbison, E. J. Barth, R. D. Schrimpf, B. D. Sierawski, E. B. Pitt, R. A. Weller, M. L. Alles, R. A. Reed, A. F. Witulski, "System Health Awareness in Total-Ionizing Dose Environments," *IEEE Trans. Nucl. Sci.*, vol. 62, no. 4, Aug. 2015.

[65] "Pololu 3pi User's Guide," https://www.pololu.com/docs/0J21.

[66] R. A. Reed, C. Poivey, P. W. Marshall, K. A. LaBel, C. J. Marshall, S. Kniffin, J. L. Barth, and C. Seidleck, "Assessing the impact of the space radiation environment on parametric degradation and single-event transients in optocouplers," *IEEE Trans. Nucl. Sci.*, vol. 48, no. 6, pp. 2202–2209, 2001.

[67] A. E. Brockwell, "Parallel Markov chain Monte Carlo simulation by pre-fetching," *J. Comput. Graph. Stat.*, vol. 15, no. 1, pp. 246–261, 2006.

[68] A. Lee, C. Yau, M. B. Giles, A. Doucet, and C. C. Holmes, "On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods," *J. Comput. Graph. Stat.*, vol. 19, no. 4, pp. 769–789, 2010.