NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

SENSOR FAILURE DETECTION THROUGH INTROSPECTION

by

Jeremy Smeltz
Andrew Valerius

June 2007

Thesis Co-Advisors: Craig Martell
Kevin Squire

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 2007	Master's Thesis

4. TITLE AND SUBTITLE Sensor Failure Detection through Introspection	5. FUNDING NUMBERS
6. AUTHOR(S) Jeremy Smeltz, Andrew Valerius	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited	

13. ABSTRACT (maximum 200 words)

The advancement of robot technology holds many opportunities for military applications. One area of research being done is simultaneous localization and mapping (SLAM). SLAM uses a robot's sensors to generate a map of the area while maintaining its current position within that map. SLAM research is built upon the assumption that all of the sensors are working correctly. Since field conditions are likely to cause erratic sensor function due to damage or inclement weather conditions, this assumption must be addressed.

The goal of our research is to discover methods of effectively performing self-diagnostic checks on robots to detect failures and malfunctions in sensors. There has been little work in the area of error detection in sensors, and what little work has been done has limited applications. This thesis will perform a series of experiments using a variety of different error detection techniques. It is our hope that the methods developed will prove to be applicable to a variety of real world systems.

14. SUBJECT TERMS Error Detection, Sensor Failure, Introspection, Probabilistic Modeling, Autonomous Robots	15. NUMBER OF PAGES 57
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

SENSOR FAILURE DETECTION THROUGH INTROSPECTION

Jeremy M. Smeltz
Ensign, United States Navy
B.S., Miami University, 2006

Andrew J. Valerius
Ensign, United States Navy
B.S., Marquette University, 2006

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 2007

Authors: Jeremy Smeltz

Andrew Valerius

Approved by: Craig Martell
Thesis Co-Advisor

Kevin Squire
Thesis Co-Advisor

Peter J. Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The advancement of robot technology holds many opportunities for military applications. One important area of research is simultaneous localization and mapping (SLAM). SLAM uses a robot's sensors to generate a map of the area while maintaining its current position within that map. SLAM research is built upon the assumption that all of the sensors are working correctly. Since field conditions are likely to cause erratic sensor function due to damage or inclement weather conditions, this assumption must be addressed.

The goal of our research is to discover methods of effectively performing self-diagnostic checks on robots to detect failures and malfunctions in sensors. There has been little work in the area of error detection in sensors, and what little work has been done has limited applications. This thesis will describe a series of experiments using a variety of different error detection techniques. It is our hope that the methods developed will prove to be applicable to a variety of real world systems.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

We would like to first and foremost thank our families for their unwavering support throughout this challenging year. We are also grateful to our Advisors, Craig Martell and Kevin Squire, for their insight and guidance. Without their help, we could not have produced this work.

We would also like to thank Capt. Eric Sjoberg, USMC for his immense contribution to our understanding of DP-SLAM and the underlying mathematics. The Autonomous System Lab as a whole was a great help in our research.

We appreciate the help received from MobileRobots. One employee, Zeb Dahl, corresponded with us and answered all of our questions about the onboard software and common problems experienced.

A special thanks to Chris Henning and the Engineering Support Team at SensComp Inc./EDP Company. They generously supplied us with 3 malfunctioning and 3 used sonar transducers, which allowed us to create much more realistic experiments and test scenarios.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The advancement of robot technology holds many opportunities for military applications. One important area of research is simultaneous localization and mapping (SLAM). SLAM uses a robot's sensors to generate a map of the area while maintaining its current position within that map. SLAM techniques have advanced within the last few years, but the research is built upon the assumption that all of the sensors are working correctly. If this assumption turns out to be false, as is likely in the field, the SLAM implementation will fail do to faulty readings taken as truth. Before the technology is robust enough to be used in the field, such assumptions will need to be addressed.

Another exciting area of research is autonomous planning and task management. The idea is to give a group of robots a goal and for them to achieve that goal autonomously. The mission would need to be divided into sub-goals, and perhaps even further subdivided from there. Then the sub-goals will be assigned to robots based on their capabilities to accomplish that objective. For example, if an object needs to be moved, then a robot with some sort of gripper might be assigned to the task. Thus, the need arises for each robot to be able to understand its capabilities and announce them to the rest of the group. Furthermore, inclement conditions such as a sandstorm or sensor failures may alter those capabilities. For the robot team to be effective, capability changes must be made known to the group leader and the tasks adjusted accordingly.

Both SLAM research and autonomous task management have the need for the robot to perform some sort of sensor diagnostics to perform effectively. Our research focuses on performing self-diagnostic checks on a robot's sensors in order to determine when a sensor is malfunctioning.

Each type of sensor, whether it is sonar, Infrared (IR), Laser Range Finders (LIDAR), or pressure sensors, will have a different model of correct functioning and failure states. Our research will specifically address issues for sonar transducers, but the general approach will be useful in similar research for other sensor types.

In the past, most applications have simply checked one sensor type against another and in the case of disagreement, take the reading of the more trusted sensor as truth. We have found no attempt in the past to determine which reading is actually the correct one. This method is insufficient for use in military applications because any of the sensors could be damaged by combat and activity in the field. When we take these systems out of the controlled laboratory environment, all sensors become suspicious to one degree or another; any one of them is susceptible to breakage. Sonar transducers in the field could also be damaged or destroyed by flying objects, explosions, or being covered by a foreign substance (such as mud). In this case we would notice a maximum reading for the single sensor in question. Sonar is very prone to faulty readings, especially in very chaotic environments.

A. OUR HARDWARE SETUP

We performed our experimentation with a MobileRobots Pioneer P3-DX robot. The implementation of our specific sonar system only returned the distance to the closest object in the sensor's detection area. Combined with the fact that each sensor had an arc of detection of 15° and very little overlap with its neighbors, any checking of a reading was extremely difficult. All the failure states that we were able to create in our experiments led to a sensor reading of 5000. While this made it impossible to determine the type of failure (which would be necessary for implementing corrective measures), it did simplify the testing for failures. According to MobileRobots, the three most common sources of failures are in one of three locations: the micro-controller, multiplexer board, or the sonar

transducers themselves. A failure in the micro-controller would be cause by a severing of the wires that transmit data either to or from the micro-controller, or from damage to the controller itself. This would be characterized by 5000 readings on all sonar sensors in the array. If the multiplexer board fails to feed data to the micro-controller due to loss of connection or damage to the board, we would notice 5000 readings on all transducers connected to that board (either all front or all rear sensors). The transducers themselves are the most likely to fail, simply due to the fact that the other possible failing components are protected from the elements by the robots body.

This final failure case, where it is only due to a single sensor failure, is by far the most difficult to detect without other sensors to check against. Seeing as this is also the most likely for systems in the field, it is important for this question to be addressed. One way to deal with this problem is for the system to learn what a 'normal' reading is, as compared to each of its neighboring sensors. For example a reading of 5000 (maximum value) on one sensor while its neighbors have low readings, although still possible. Through the collection of a vast amount of readings from varying situations in the normal world, we can develop probability models for the triples consisting of the readings from a sensor and the neighbor on each side of it. These models would vary greatly depending on the environment they were created in, such as a home, a warehouse, or a heavily wooded area. Due to this variation it would be important for the robot to learn this model in an environment that closely resembles the area it will be expected to operate in.

The rest of this thesis is organized as follows. In Chapter II, we discuss current state of research in the field of robotics. Chapter III will explain the experimental setup and results. Finally, Chapter IV will cover our conclusions and future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. RELATED WORK

There are many areas of research that have relevance to our work. This chapter will look at some of the major concepts in autonomous robotics that were necessary for our research. We will first look at ways of understanding a robot's behavior through models. We will then explore the way that researchers translate sensor readings into relevant data using occupancy grids. Research in the use of sonar for accurate mapping/navigation will follow. Finally, we will discuss other research in error detection of sonar sensors.

A. MOTION AND SENSOR MODELING

The development of motion and sensor models has and will continue to be very important to the field of robotics. Without accurate models, the system has no way of relating its actions and readings to real world behavior and distances. For example when a system moves itself forward what it believes to be one meter, it will actually move a distance close to that but not exactly the same. If the system designers do not compensate for these small differences in perceived and real world truths, these inaccuracies will build upon each other until it grows into a problem that cannot be ignored.

The importance of highly accurate motion and sensor models was realized early on in the development of the field. A number of papers and books are available on the subject with many solutions to the problem of how to develop models for a specific application.

1. Probabilistic Algorithms in Robotics

A robot is inherently uncertain about its perceptions and therefore what actions to take next. Sebastian Thrun argues that in the case of robot perception and control probabilistic approaches will always outperform non-probabilistic methods for real-world complex applications. The reasoning behind the whole

approach is that "a robot that carries a notion of its own uncertainty and that acts accordingly will do better than one that does not." [14:1] Thrun argues that Markov localization techniques are the best performers for global localization, as pioneered by many such as Simmons and Koenig at CMU. These systems require an amount of prior knowledge of the sensors and motion of the robot in order to know what levels of uncertainty to work into the model. Through some prior assessment of the systems, we are able to develop a robust model that will permit reliable performance in complex situations.

2. Principles of Robot Motion: Theory, Algorithms, and Implementations

According [3], the value of a sensor model depends on a number of factors including the type of sensor used, the environment, and how the environment is represented. The author discusses different considerations in making the model and how we need the model in order to determine $P(y \mid x)$, which is the probability of a reading *y* on the sensor given position *x*. This is similar to the approach we used for our failure detection, except our probability was conditioned on the readings of neighboring sensors, instead of the position of that sensor or the robot. They propose a sensor model that was "designed to capture the noise and error characteristics of many active range sensors." [3, 322] One first determines the different situations that can arise when obtaining sensor data, and develops distribution models for each of these separate situations. Once a specific reading is obtained, the likelihood can be calculated by combining the distributions of each of the separate possible situations. The situation that gives us the highest probability of the reading we obtained is taken as the true case. Through this use of prior readings and the knowledge they impart, we are able to obtain more accurate localization and mapping data. One must take care to obtain prior data with which to build the models from an environment sufficiently similar to that which the system will be operating in.

B. EVIDENCE GRIDS AND MAPPING

As mentioned above, one promising and interesting area of research is robotic mapping, and evidence grids are one standard approach to robotic mapping.

Building maps from sonar data has proven to be a complex problem. Early methods made quick decisions on the existence of an object based on sometimes questionable data. If the decisions proved to be correct, this system was effective. However, when an incorrect decision was made, the error pervaded corrupted future inferences. These methods proved to be inadequate. A major leap forward came in 1983, when the evidence grids were proposed [11]. They were intended to convert readings from inexpensive, wide angle sonar sensors into high-detail maps. Large strides were not made in mapping until the advent of this concept.

Since 1983, there have been many attempts at mapping with sensor data (see [2], [5], [8], [9], [12]). Some very promising work has been done recently by Eliazar and Parr [6]. We summarize some of this work below.

1. DP-SLAM 2.0

Extensive work has been done recently in the field of Simultaneous Localization and Mapping (SLAM). SLAM is a method of generating maps from sensor data without a map or an exact initial position. SLAM is difficult because it is solving two problems simultaneously. To make an accurate map, precise position estimates (localizations) are needed for the updates to be consistent. To perform localization effectively, one needs to have accurate maps. If either the map is known or the real position can be determined, then the other task becomes relatively straightforward. Eliazar and Parr wrote two papers in 2003 with a new approach to SLAM (called DP-SLAM). Their algorithm makes no

assumptions about pre-determined landmarks. DP-SLAM is designed for use with laser range finders because they are fairly inexpensive, have a narrow beam, and can be very accurate.

Particle filters are essential to the DP-SLAM algorithm. A particle filter is a sampled representation of a probability distribution. It maintains normalized, weighted set of sampled states, each one called a *particle*. In particle filter algorithms, the particles are taken through a transition and each new state is then weighted according to a quality measure. Finally, weights are normalized for the new set of states [6, 2].

This translates very well to localization with a known map. Given a map, a series of particles will be created throughout that map to represent possible locations of the robot. Initially, each particle will be equally weighted. The robot is given a move command and each particle has that move, or transition, applied. The move is determined by the robot's motion model and is represented by:

$$\begin{aligned} x_i &= a_x * x + b_x + N(0, \sigma_x) \\ y_i &= a_y * y + b_y + N(0, \sigma_y) \\ \theta_i &= a_\theta * \theta + b_\theta + N(0, \sigma_\theta) \end{aligned}$$

Equation 1

The *a* and *b* terms represent correction for the consistent errors in normal motion. *N(0, σ)* returns a random value from a normal distribution with mean 0 and standard deviation (*σ*) that is determined by the motion model. The robot then takes a series of observations. The observations are compared to each particle's simulated observation from the new location. For particle *i* this is represented as:

$$P_i = \prod_k p(\delta_{ik} \mid S_i, m)$$

Equation 2

where δ_{ik} is the difference between the expected and perceived distances for sensor *k* and particle *i.* Thus for each particle, as the discrepancy between the

real and simulated readings increases, its weighting is decreased. After moving through the map, a good set of particles will be distinguished by the high correlation to the actual location and orientation of the robot within the map.

Many SLAM techniques keep a single map and a set of proposed positions and orientations. DP-SLAM distinguishes itself from other SLAM methods because it also maintains many sets of possible maps for the particle filter. The DP-SLAM algorithm uses particle filters on all of the maintained maps, and determines probabilistically the map and particle that most likely represents the robot's orientation within that map.

Current DP-SLAM implementations assume the systems sensor suites to be functioning properly and its readings to be fairly accurate. Our research begins to ask how one can model and detect failures for a given sensor. If this approach proves successful future DP-SLAM implementations would be able to identify and compensate for broken sensors, making a much more robust mapping algorithm. Along the same goal, a similar approach could be used to model the accuracy of a given sensor reading, further improving the quality of generated maps.

C. OTHER RELEVANT WORK

Our search for research in the area of sensor failure detection returned very little, indicating that this field has not been thoroughly examined. Reference [1] approaches this problem in a novel way, although we were unable to apply any of the methods used. This is because they used two types of sensors and were able to do cross-echoes between nearby sonar sensors. Our failure to apply and of their methods was due to our lack of other sensor suites and the inability to do cross-echoes.

1. An Error Detection Model for Ultrasonic Sensor Evaluation on Autonomous Mobile Systems

Work by D. Bank [1] relates most to our objectives. Bank's research was specific to an environment different from ours. Despite these differences, our work was helped a great deal by his research. Bank's error detection model used a combination of different methods to determine whether a sensor is working correctly.

Bank's experimentation took place on an XR4000 robot. The sonar system actually contained two different sonar systems, each with 24 sensors. One set (made by Polaroid) had a detection cone of 30°, while the other (Bosch) had a detection range of 60°. The Polaroid system was mounted such that the angular displacement between each sensor was 15°. The Bosch system sensors were mounted directly above the Polaroid sensors, also at 15° displacement. For this model, however, only the Bosch sensors were used for ultrasonic readings. Laser range finders were also installed on the robot.

The first step in Bank's methodology was to create a "simulation model" using the laser range finders. The laser scan enables the distinction between planes, corners and edges defined by the paper as:

> A plane is represented by a line in the two-dimensional environment model. Lines are represented in Hesse's normal form.
>
> A corner is concave dihedral, and produces specular returns. Corners are represented as points in the two-dimensional model.
>
> An edge is a convex dihedral, and produces diffuse reflections. Like a corner, an edge is represented by a point in the two-dimensional model [1, 3].

The data from the laser finders was analyzed to make a rough map containing all of the planes and corners. Experiments using sonar sensors

indicated that detectable echoes generally came from planes and corners. Therefore the edges were ignored. This model then enabled predictions about the sonar data.

The next step in the process was comparing the ultrasonic sensor readings to the simulation. In their setup, each sonar sensor had one echo return and six cross-echoes. That is, the return emissions from one sensor were received by another up to six others. This produced a 7-by-24 matrix. They collapsed the 7-by-24 matrix into a 4-by-24 matrix by "[averaging] over the two-fold existing echo paths and returning an error code if the... readings were inconsistent [1,4]." The resulting matrix was then used with another 4-by-24 matrix containing the simulated echo returns. A new "simulation-based confidence matrix" containing values of 0 and 1 was derived from the two, based on whether the difference matrix was within the threshold of .1m.

The next stage of sensor evaluation was to process the real ultrasonic sensor readings. Given the width of the beam and the spacing between sensors, each sensor could potentially detect the adjacent three sensor's echoes on each side. This resulted in seven readings (or matrix entries) for each sonar sensor. Thus, each sensor had 51 overlaps of direct and cross echo paths. This was converted into a 51-by-24 "intersection matrix," with a 0 or 1 indicating whether an intercept point within the overlapping range did or did not exist. This matrix was collapsed into a 4-by-24 "intersection based confidence matrix," with each entry ranging between 0 and 1.

The final step of the error detection process was to average the simulation-based confidence matrix and the intersection based confidence matrix. The resulting matrix was normalized to [0,1]. If an element in the matrix became zero, then the system declared that sensor to be faulty.

Bank's methods appeared to be effective because he integrated multiple ways of testing the sensors. As previously mentioned, we were unable to

implement any of his ideas because of hardware limitations. Our work could possibly contribute to Bank's error detection model by providing additional input to the algorithm.

III. EXPERIMENTS AND RESULTS

Probabilistic methods have been effectively used with robots. When interacting with real environments, it is often impossible to model every feasible situation. Probabilities are then useful to make a "best guess" about the current conditions. We used this type of approach for our attempts at error-detection. This section will first detail the collection of all of the data. Next we will describe our five experiments which varied the ways we analyzed the data to detect failures. The end of this chapter will discuss the results of each experiment.

A. DATA COLLECTION

The first step in finding a method of sensor failure detection was to gather large amounts of sensor data from various situations. There were two different phases of data collection. The first phase involved gathering data where all of the sensors were working correctly. This was done in several ways. One way was to manually drive the Pioneer through a variety of situations, polling the sensors before and after each movement. To automatically collect data, we wrote an application that made the Pioneer turn in a random direction and move as far as it could without hitting an object. Again, sensor data was collected before and after each movement command. Table 1 shows generally the way that data was organized in a spreadsheet. Note that each row contains the readings from a poll at some instance of time. We also annotated in the tables that the data came from correctly working sensors.

	Sensor Number						
Sensor Poll	*0*	*1*	*2*	*3*	*4*	*5*	...
1	4153	1839	597	630	4294	2166	...
2	5000	1008	2451	1437	2329	2102	...
...	...	...	...	...	...	...	...

Table 1. Example Sensor Data

To prepare for future experiments, we also collected sets of more detailed information. In addition to sensor data, we also wanted actual distances of objects from the sensors. This was necessary because, although a sensor may be working correctly, inaccurate readings may result from beams deflecting off of walls and flat surfaces. Gathering this data involved placing the Pioneer in a number of known settings. For the first scenario we placed the Pioneer perpendicular to a wall. We determined R, which is the center of the arc formed by the sensors. We then measured the distance *d* from *R* to the wall. Finally, we measured the distance between R and each of the sonars, which was 15.9cm, or 159mm. The angle *a* for any given sensor was already known. Thus the distance *t* in millimeters from a sonar sensor to the wall was found by the equation $t = d\cos a - 159$. Once the robot's *d* was measured, we took five sets of readings and moved the robot forward 50mm. At each step the readings were taken and the *t* values recalculated.

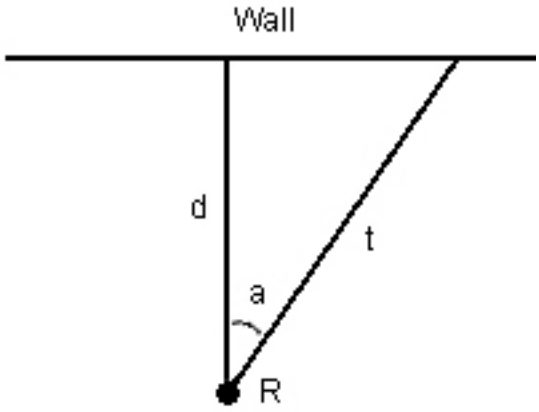

Figure 1. Collection Scenario 1

The second scenario for data collection was very similar to the first, except the Pioneer was not aligned perpendicularly to the wall. Instead, sensor #6 was facing the wall. The basic process was the same as before, and the resulting trigonometry did not change. See Figure 2. The dotted line represents the perspective of sensor #6.

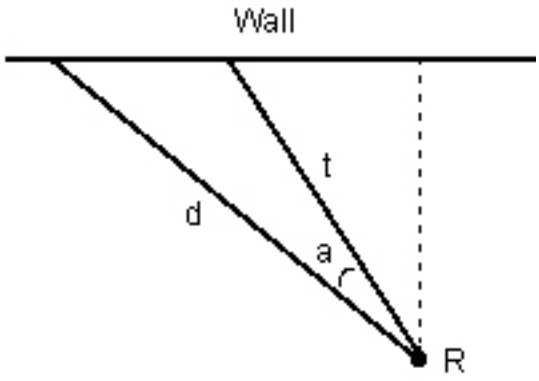

Figure 2. Collection Scenario 2

Both scenarios were repeated several times to efficiently collect the sensor and measured information in the known environment. These scenarios were also the primary method employed in collecting broken sensor data. The general process was repeated again, except with a broken sensor replacing a previously working one. The tables where this data was stored were marked as having faulty readings, allowing for future analysis.

B. EXPERIMENT 1: USING MODELS TO DETECT FAILURES

The first way we approached the problem of sensor failure was to use the model,

$$P(reading \mid l, r). \qquad \textbf{Equation 1}$$

That is, given the readings of the left and right sensors, how likely is the given reading? To perform this calculation, we needed to model the likelihood of a sensor's readings in relation the adjacent sensors.

Once the data was collected, it was compiled into a large dataset consisting of faulty and normal sensor data. We began to consider the data as many sets of triples. A triple represents readings taken from three adjacent

sensors. For example, one valid triple from Table 1 would be from the first poll on sensors #1, #2, and #3. The resulting triple in this example would be <1839, 597, 630>.

The next step was to find all the triples where the sensors had similar characteristics. To better comprehend the grouping of the triples, it is necessary to understand the general layout of the sonar sensors on the Pioneer P3-DX. Figure 3 shows the positioning and numbering of the Pioneer's sensors. The angle between sensor #0 and sensor #1 is 40°. This is the same as the relative angle between #6 and #7, #8 and #9, and as the angle between #13 and #15. All other adjacent sensors are separated by a 20° angle.

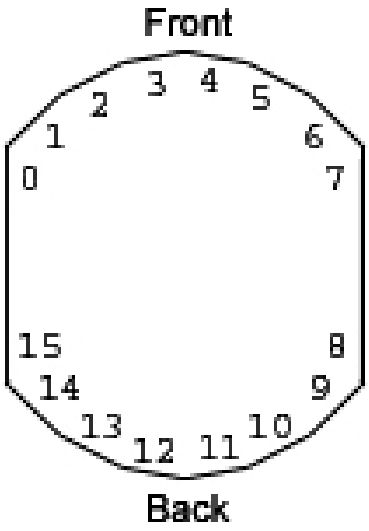

Figure 3. Sensor Positioning on Pioneer Robot

Using the above illustration, three resulting subsets can be determined. The sensor characteristics of interest are the relative angles between a set of three adjacent sensors. We will use the notation: [a, 0, b] where each number represents an angle offset from the second sensor. Enumerating through all possible groupings of three adjacent sensors revealed three general cases of sensor characteristics.

Case 1: [-40, 0, 20] or [-20, 0, 40]

Case 2: [-20, 0, 20]

Case 3: [0, 0, 40] or [-40, 0, 0]

The first case is where the triple contains a relative angle of 40° and a relative of 20°. All rows of triples from sensors #0, #1, and #2 fall into this category. The second and most prevalent case is where the triple contains two relative angles of 20°. Sensors #3, #4, and #5 belong in this subset. The final subset is where the triple contains a relative angle of 40° and a relative of 0°. This includes triples from sensors #6, #7 and #8.

After the data was collected and organized, we focused on the [-20, 0, 20] case and created a 3D plot of the triples that did not contain the value of 5000 for the middle sensor. This case was handled separately and will be explained later in the paper. The presence of clusters indicates some predictable relationships. The figures below show 3D and 2D views of the resulting graphs.

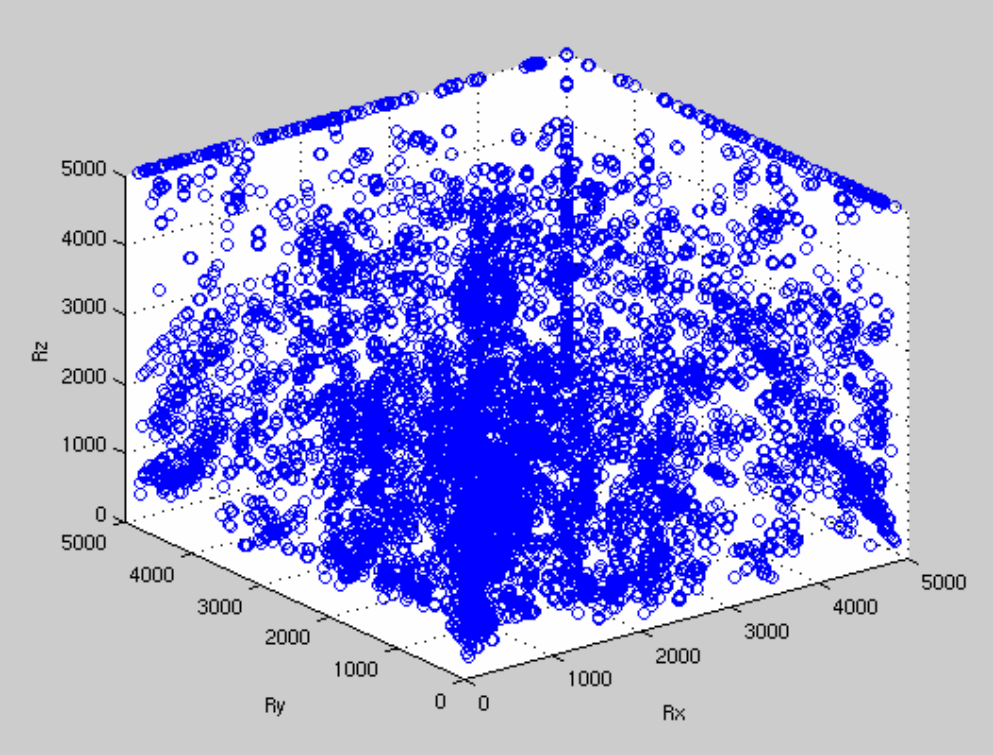

Figure 4. 3D Plot of [-20,0,20] Triples

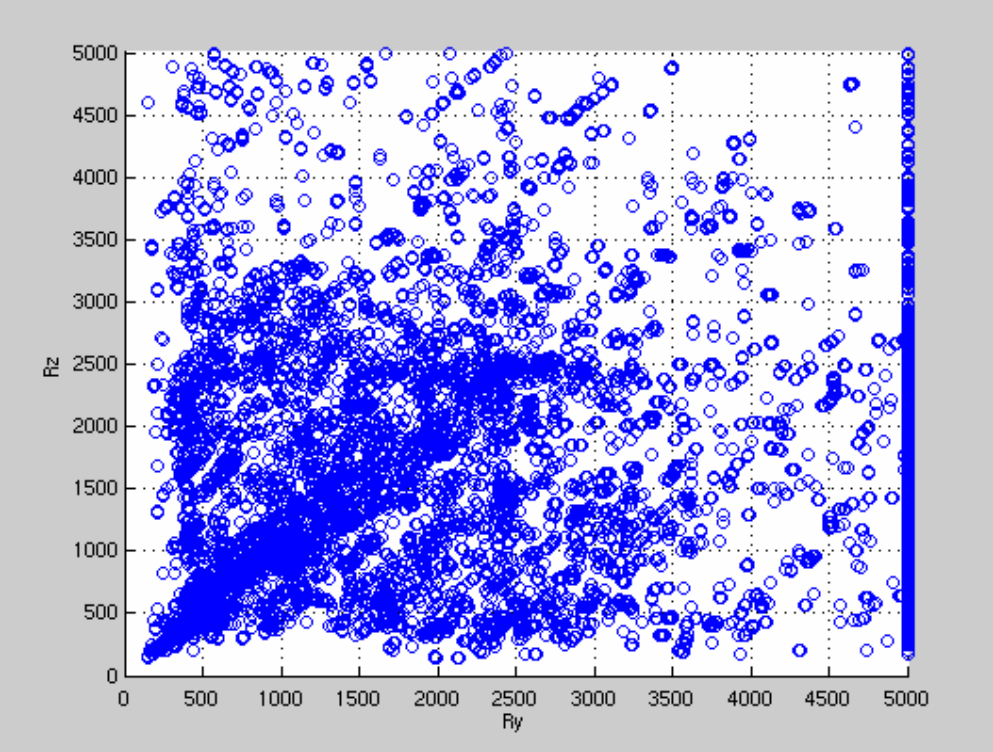

Figure 5. Z-Y Plot of [-20,0,20] Triples

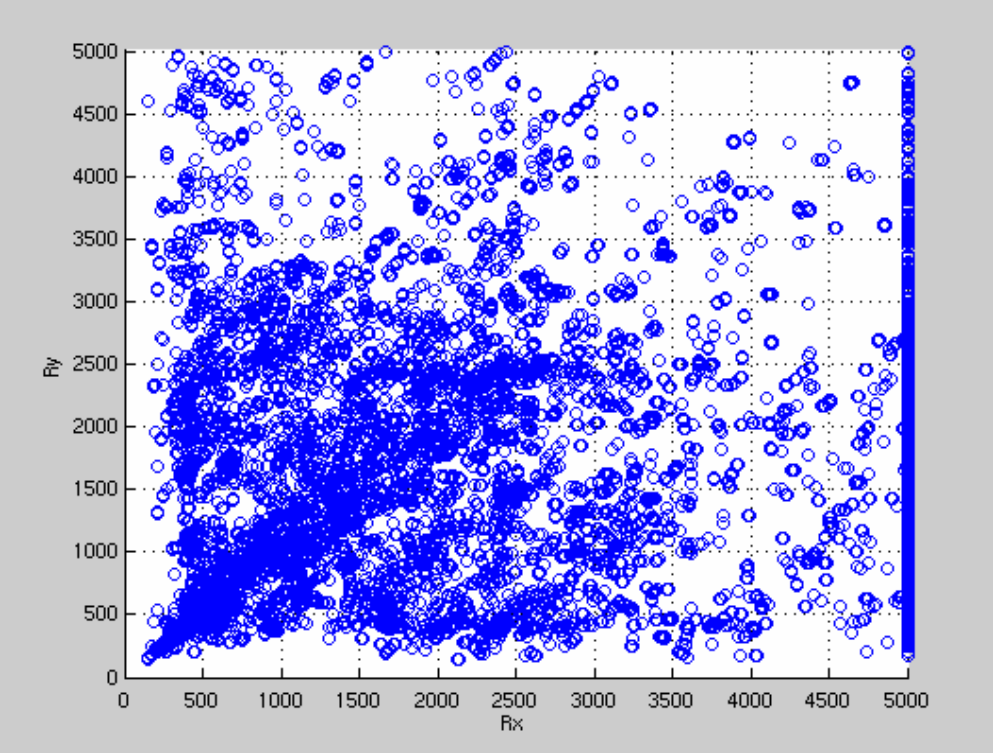

Figure 6. Y-X Plot of [-20,0,20] Triples

The data space was then divided into smaller sections of length 750. For example, one section was the block of *X* values ranging from 751-1500, and the *Y* values from 1501-2250. From each section we generated a histogram that showed the frequency of the middle sensor distance, given the parameter's of the other two. Figure 7 shows the frequency of Y at each reading where both the X and Z values range between 751-1500.

Once the histogram was created for each section, we compared their shapes against common distributions. In all cases, the quadruple-Gaussian distribution most accurately estimated the data. A quadruple-Gaussian has four sets of parameters. Each Gaussian had a mean, variance, and a weight. This "best-fit" distribution became the basis of the models.

To calculate the likelihood of a reading given the readings of its neighbors, we used the X and Z values to determine which model to use. The CDF for each Gaussian was calculated, multiplied by its corresponding weight, and added together. To calculate the CDF for each Gaussian, the following equation was used, where *r* = the reading of the sensor being examined.

$$F(x:\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}\int_{r-25}^{r+25}\exp\left(-\frac{(u-\mu)^2}{2\sigma^2}\right)du$$

Equation 2

The total cumulative probability of a reading was then equal to:

$$P(r) = F_1 * w_1 + F_2 * w_2 + F_3 * w_3 + F_4 * w_4$$

Equation 3

where *w* is the weighting of each Gaussian.

The final step in creating the algorithm was to discover the optimal threshold, above which we declare a reading to be accurate. This was done by experimenting with a variety of thresholds and calculating the precision and recall

for each. Precision, recall, and F-scores shall be explained in the results section, and the equations are in Appendix A. The results of this experiment are described at the end of the chapter.

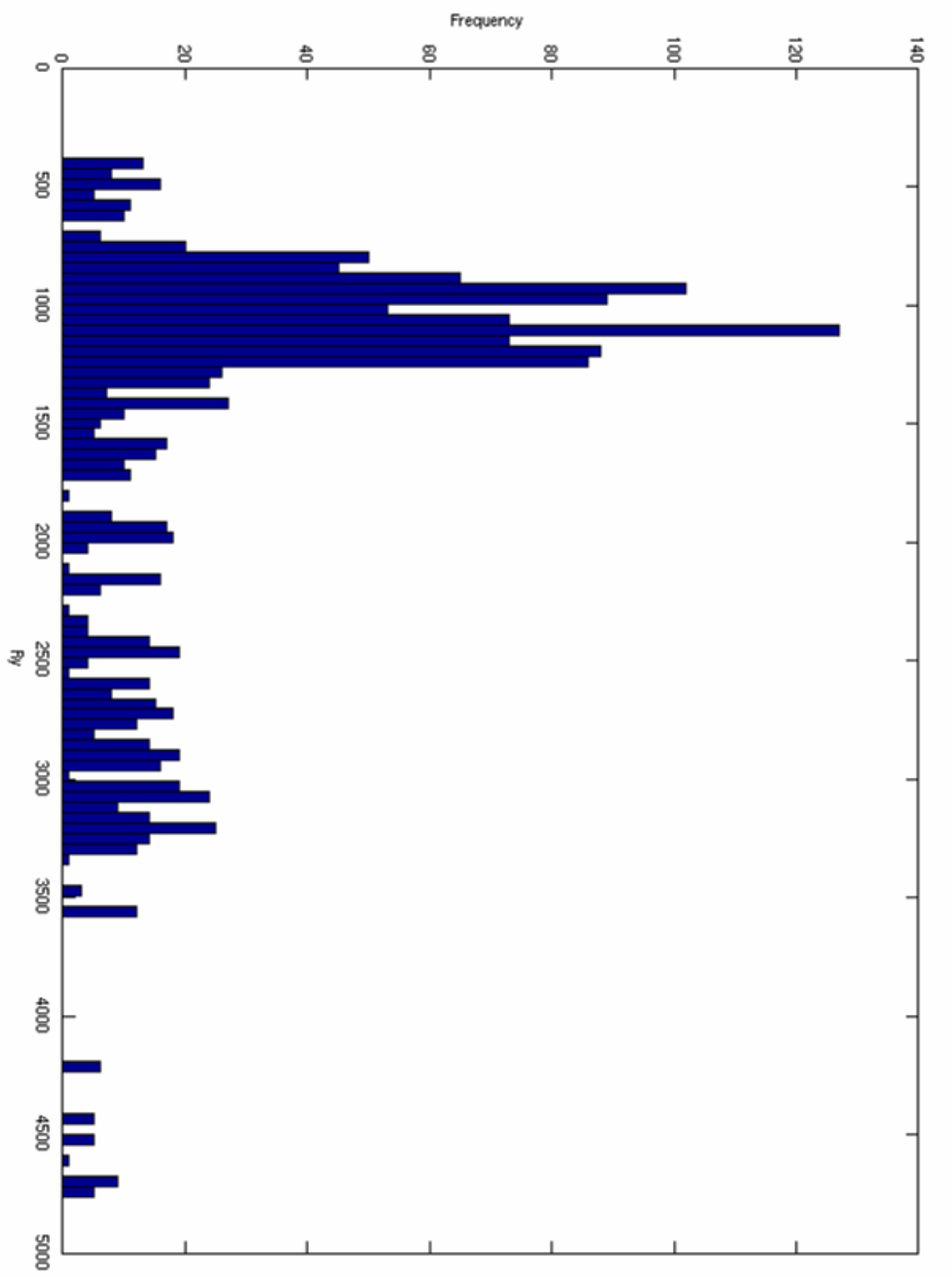

Figure 7. Example Histogram

C. EXPERIMENTS USING NAIVE BAYES CLASSIFIER

Our second attempt of the problem utilized a Naive Bayes Classifier, represented by the following equation:

$$P(C_i \mid l, r) = \frac{P(l, r \mid C_i)P(C_i)}{P(l, r)} \qquad \textbf{Equation 4}$$

Where C_i is a classification of a sensor (broken or working), and the terms *l* and *r* are the sensors to the left and right of the sensor being examined. The term "naive" refers to assumptions about independence. In our case, we assumed independence between the readings of *l* and *r*. This was a reasonable assumption because they face in different directions (40° difference) and the fact that the sonar will deflect off surfaces at any angle greater than 20° from the sensor. This assumption simplifies the equation to:

$$P(C_i \mid l, r) = \frac{P(l \mid C_i)P(r \mid C_i)P(C_i)}{P(l)P(r)} \qquad \textbf{Equation 5}$$

The advantage to using the Naive Bayes Classifier was to enable an argmax operation, which finds the element in *C* that produces the highest value. Additionally, since the numerator will be the same in both cases and we are looking for the higher of the two, the numerator can be ignored. This makes our evaluation function:

$$C = \arg\max_i P(l \mid C_i)P(r \mid C_i)P(C_i) \qquad \textbf{Equation 6}$$

where *i* has the value of "working" or "broken."

To further isolate the problem of detecting failures, we decided to examine only sensors that had readings of 5000. This was reasonable because in every failure state that we observed, the sonar sensor returned a 5000. This

simplifying assumption made it easier to derive the values on the right side of the equation. The value of each term was then found through table lookups. The following experiments varied the mix of broken/unbroken data as well as the combination of training/test data. The variance in methods resulted in differences in tables and results.

As mentioned above, our analysis was performed only on the 5000 readings. Therefore, all references to data hereafter implicitly refer to only the data with a 5000 for the center reading. Table 2 below gives information on the usable data used for experiments 2-5.

Description	Information
Working sensor data points	6795
Broken sensor data points	6350
Total data points	13145
Percentage of data that is working	.5169

Table 2. Information about the data

1. Experiment 2

The first step in *Experiment 2* was to divide the data into "training data" and "test data." Training data is used to create the tables, but the test data is processed by the resulting algorithm, allowing for evaluation. This set of experiments used a mix of 90:10 training to test data ratio. Therefore 90% of the data became the training data while the remainder was considered test. The broken and working data stayed separate. Both sets of training data triples were then split into two more groups: left sensor data and right sensor data. Since we ultimately needed probabilities, we performed conversions on the data. To do so we divided the range of the sensors (0,5000) into "bins" of size 500 each. So there was a bin from 0-499, one from 500-999, and so on. Each bin is intended

to hold a number representing the frequency that values within the range have been seen. The probability for any bin is found by dividing the frequency by the total number of data points.

Because the data was sparse, we next performed smoothing on the frequencies for each bin. We used Witten-Bell discounting to smooth our data because it is fairly easy to implement and tends to give better estimates than add-one smoothing. [7] details different types of smoothing including Witten-Bell and Good-Turing. An explanation of the Witten-Bell smoothing algorithm can be found in Appendix A.

The working-left, working-right, broken-left, and broken-right sets of data were all was processed separately in this way. By keeping the left and right data separate, this experiment operated under the assumption that the robot's environment was asymmetric.

At this point, we had all of the information we needed to solve *Equation 6.* Here's an example where *i* = working:

$P(l \mid W)$ - found in working-left table

$P(r \mid W)$ - found in working-right table

$P(W)$ - percentage of data that is working = .5169

To calculate *Equation 6* where *i* = broken, the tables containing the broken data would be used, and the value of $P(B)$ (proportion of broken sensor data) would be equal to .4031.

2. Experiment 3

This experiment was done similarly to *Experiment 2*, with the exception of the splitting of left and right data. This experiment made the assumption that the robot's environment is mostly symmetric. This was reasonable because the world is generally symmetric. Making this assumption had the benefit of increasing the amount of data used to calculate the probabilities and reducing the

complexity of solving *Equation 6*. Once the data was processed, there were only two tables: one containing working data and the other containing broken data. The values of $P(r | C_i)$ and $P(l | C_i)$ were then found from the same table.

3. Experiment 4

The basis of *Experiment 4* was to use leave-one-out cross-validation (LOOCV) in building our tables. Given *N* data points, LOOCV uses *N-1* of them as training data and the remaining data point as the test data. This process is repeated *N* times until all data points have been used as the test data exactly once. We used this method of partitioning because of few data points contained within the broken sensor data.

The broken and working data stayed separate, and again each set was split between left and right. The process of "binning" the data and calculating its probability remained the same as experiment two. The consequence for using this method was that we had to process the data for each entry. This resulted in generating new tables, performing Witten-Bell discounting on the data, and calculating the argmax 5929 times.

4. Experiment 5

Experiment 5 also used cross-validation. The difference between this experiment and the previous is that the left and right data was never separated. Like *Experiment 3*, we made the assumption of symmetry in Pioneer's environment. The data was otherwise processed in the same way as *Experiment 4.*

5. Experiments 6-9

These experiments were performed exactly in the same way as 2-5, using different sets of data. The results section will detail the differences and the reasoning behind them. Table 3 shows the set of data used for these experiments.

Description	Information
Working sensor data points	5224
Broken sensor data points	705
Total data points	5929
Percentage of data that is working	.8811

Table 3. Information about the data

D. EXPERIMENT RESULTS

To evaluate our results from these nine experiments, we used precision, recall, and the F-score as metrics. [7] contains the mathematical explanation of these terms, and the equations can be found in Appendix A. The precision tells us what percentage of sensors we labeled as working actually worked. The recall gives us what percentage of working sensors were identified as working. An F-score is used to find a balance between the two, where the ideal approaches 1. The baseline is found by declaring all sensors as working (recall = 1) and calculating the F-score. The results from all five experiments are listed below in Table 4. Appendix B contains the confusion matrices for each experiment.

Experiment #	Precision	Recall	F-score	Baseline
1	0.8262	0.5557	0.6645	0.9372
2	0.5167	1.0000	0.6814	0.6814
3	0.5167	1.0000	0.6814	0.6814
4	0.4882	0.824	0.6286	0.6814
5	0	0	0	0

Table 4. Experiment Results

Experiment 1 was our first pass at trying to detect sensor failures. Since many applications use models in this way, we thought it might be effective in this setting. Unfortunately, this was not the case. When a reading's probability was calculated, it was compared to the threshold. If the probability was above the

threshold, the algorithm declared the reading to be accurate. Otherwise it was assumed broken. Finding the best threshold involved testing various values between 1 and .001. Then for each value, the precision, recall, and F-score were calculated. The entry in Table 5shows the highest F-score we could obtain using this method. Even our highest F-score still had a value well below the baseline. The disparity between the two was a strong indicator that this method was inadequate for our problem.

The second experiment was motivated by the success of Bayesian methods in robotics. Although the algorithm is fairly simple, it often leads to promising results. In our case we assumed independence between the left and right sensor, which might not have produced the best results. The advantage was that there was less computation involved, and the assumption of independence was not unreasonable. *Experiment 2* used a 90/10 split on training/test data and assumed an asymmetric environment. As shown in Table 4, this experiment got fairly low results. *Experiment 3* was then just a variation on *Experiment 2.* By combining all the left and right sonar data, we had hoped the increase to the data would lead to more accurate predictions. This was not the case, however. The results were identical. This seemed to indicate that the robot's environment was in fact, symmetrical in this setting.

Experiments 4 and 5 paralleled *Experiments 2 and 3*. By using cross-validation we had hoped to get better predictions through the utilization. The opposite was true, however, and resulted in an even lower F-Score than *Experiments 2 or 3.* For *Experiment 5* we combined the left and right data in the same way as *Experiment 3.* This experiment produced the most peculiar results: all readings were mislabeled.

Experiments 2-5 all produced poor results. We concluded this to be because the broken data points and the working data points looked so similar, that we were not able to distinguish between the two. We thought the way to improve the experiments would be to force a difference in the probabilities between broken and working readings. Our next step was to perform

experiments in a controlled environment where failures would be more predictable. To do this we took our broken data from "Collection Scenario 2," seen in figure two of the data collection section. We followed the exact same steps for *Experiments 6-9* as we did *2-5* with this new data set. Table 5 shows the results of this new round of experiments.

Experiment #	Precision	Recall	F-score	Baseline
6	0.9652	0.9023	0.9327	0.9372
7	0.8818	1.0000	0.9372	0.9372
8	0.8639	.8564	0.8601	0.9368
9	0.8811	1.0000	0.9368	0.9368

Table 5. Experiment Results

Experiment 6 was by far the most successful. This was the Bayesian method without combining the left and right data. Although our F-score was slightly below baseline, it gave us great results due to the very high precision. *Experiment 8* gave good results but they were below that of *Experiment 6. Experiments 7 and 9* were unsuccessful and labeled all readings as failures. This indicated that the asymmetric view was in fact much more useful than the symmetric.

Since our results for *Experiments 6*, were so positive, we decided to run it again, substituting different values for the $P(C_i)$ prior. Table 6 below shows those results. Interestingly, the ratio of .16 yielded a slightly higher F-score than the baseline.

P(B)	Precision	Recall	F-score	Baseline	TW	FW	TB	FB
.15	0.96424	0.89527	0.92848	0.93716	701	26	79	82
.16	0.96962	0.91737	0.94277	0.93715	766	24	88	69
.17	0.9577	0.91779	0.93732	0.9372	815	36	83	73
.18	0.96067	0.90957	0.93443	0.93719	855	35	91	85
.19	0.96021	0.9244	0.94196	0.93718	917	38	95	75
.20	0.95556	0.90613	0.93019	0.93674	946	44	97	98
.21	0.96459	0.91887	0.94118	0.93681	1008	37	111	89
.22	0.96091	0.91993	0.93997	0.93681	1057	43	112	92
.23	0.95581	0.9184	0.93673	0.93682	1103	51	111	98
.24	0.96241	0.91939	0.94041	0.93682	1152	45	124	101
.25	0.96359	0.91194	0.93706	0.93687	1191	45	131	115

Table 6. Variations on Experiment 6

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CONCLUSION

The positive results we from *Experiments 6 and 8* were due to us forcing an artificial difference between the working data and the broken data. This is useful because it shows our methods can be effective if we can find a way to either create a difference in the normal and broken data or if we can isolate some difference between the two types of data. To create a difference we could create a test arena, where the maximum reading would be less than the maximum range of the sonar. That way, maximum readings would always be suspect and could create that difference needed to make our algorithms accurate. In order to detect failures, there needs to be a relationship between broken readings and the sensor's neighbors.

A. CONTRIBUTIONS

Although most of our experiments were not successful, we believe that we still made contributions to the largely unexplored field of sensor introspection.

1. Our work presented some methods that may not be effective, encouraging future researchers to explore other ways of solving this problem.

2. Our research indicated that until much progress is made, building and testing in regulated scenarios will give the most accurate results with a restrictive system such as this.

3. All of the Pioneer and Matlab code we have developed will be freely available to any researchers.

At the very least we hope that our work will help to either advance the field or to spark new ideas that will contribute towards more robust and self-aware robotic systems.

B. FUTURE WORK

Despite our extensive experimentation and analysis, there is much work left to be done. In the future, we would like to see the following research:

1. Performing our experiments with the Good-Turing smoothing algorithm, or other smoothing methods applied. Good-Turing might be a better choice in this context because it is a more effective method of smoothing for cases where the data is sparse or nonexistent data for specific species [7, 214-216].

2. Collecting much more data and performing the same experiments. Due to the erratic nature of the sonar readings, a much larger data set may yield better results.

3. Perform the analysis and similar experiments on a system with different properties. For example, we believe that if the sensors had greater ranges (perhaps 20m instead of 5m) patterns would emerge that would allow us to make distinctions between broken and working readings. Additionally, having cross-echoes from the sonars would also provide us with information to allow us to relate the left and right sensors to the center.

4. Use methods explored in this paper to develop techniques for rating readings. Since sonar easily deflects off of flat surface, it would be useful in situations such as SLAM to be able to detect when a sensor is not reading correctly, but not broken either. An algorithm could rate each reading, and the program that's using the readings would ignore all readings with values below a certain thresholds. This could lead to more accurate mapping in SLAM. This would also involve gathering much more data, as well as collecting actual distances with each sonar reading.

5. Develop a test procedure that is able to create a dependency between the left and right sensors. This would be used such that we can get more meaningful information by looking at a triple about the sensor being examined.

C. CONCLUSION

In conclusion, we have shown that this specific implementation of sonar does not provide enough information to detect errors correctly under normal conditions. We had assumed that the probability of a given reading would be different for a working and broken sensor. This generally proved to be incorrect. In experiments 6-9, which were the most successful, all failures occurred when the left and center gave readings of 5000 and the right neighbor gave readings in a specific lower range. Since all failure readings were of this type, the lookup table that was created for broken sensors had a much higher probability for this type of reading than any other. Thus, any similar readings in our test data were easily and correctly classified as broken. This indicates that for our methods to be successful a difference in probabilities between working and broken sensor readings must be found. When used under conditions that can isolate these differences, our methods are very effective in determining the state of sensors.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A

A. PRECISION, RECALL, AND F-SCORE

Precision: P = $\frac{TP}{TP + FP}$

Recall: R = $\frac{TP}{TP + FN}$

F-score: F = $\frac{2}{\frac{1}{P} + \frac{1}{R}}$

B. WITTEN-BELL SMOOTHING ALGORITHM

$$C_i^* = \begin{cases} \frac{T}{Z} * \frac{N}{N+T}, & if\ C_i = 0 \\ C_i * \frac{N}{N+T}, & if\ C_i > 0 \end{cases}$$

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B

		Predicted	
		Positive	**Negative**
Actual	**Positive**	290	232
	Negative	61	9

Table 7. Confusion Matrix for Experiment 1

		Predicted	
		Positive	**Negative**
Actual	**Positive**	679	0
	Negative	635	0

Table 8. Confusion Matrix for Experiment 2

		Predicted	
		Positive	**Negative**
Actual	**Positive**	679	0
	Negative	635	0

Table 9. Confusion Matrix for Experiment 3

		Predicted	
		Positive	**Negative**
Actual	**Positive**	5996	799
	Negative	6285	65

Table 10. Confusion Matrix for Experiment 4

		Predicted	
		Positive	**Negative**
Actual	**Positive**	0	6795
	Negative	6250	0

Table 11. Confusion Matrix for Experiment 5

		Predicted	
		Positive	**Negative**
Actual	**Positive**	471	51
	Negative	17	53

Table 12. Confusion Matrix for Experiment 6

		Predicted	
		Positive	**Negative**
Actual	**Positive**	522	0
	Negative	70	0

Table 13. Confusion Matrix for Experiment 7

		Predicted	
		Positive	**Negative**
Actual	**Positive**	4474	750
	Negative	705	0

Table 14. Confusion Matrix for Experiment 8

		Predicted	
		Positive	**Negative**
Actual	**Positive**	5224	0
	Negative	705	0

Table 15. Confusion Matrix for Experiment 9

LIST OF REFERENCES

[1] D. Bank, "An Error Detection Model for Ultrasonic Sensor Evaluation on Autonomous Mobile Systems," 2002.

[2] W. Burgard, D. Fox, H. Jans, C. Matenar and S. Thrun, "Sonar-based mapping with mobile robots using EM," in *Proc. 16th International Conf. on Machine Learning,* 1999, pp. 67-76.

[3] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations.* MIT Press, 2005, p. 550.

[4] A. Elfes, "Using Occupancy Grids for Mobile Robot Perception and Navigation," *Computer,* vol. 22, pp. 46-57, 1989.

[5] A. Eliazar and R. Parr. 2003, "DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks," November 2006, citeseer.ist.psu.edu/eliazar03dpslam.html.

[6] A. Eliazar and R. Parr. 2003, "DP-SLAM 2.0," November 2006, citeseer.ist.psu.edu/article/eliazar04dpslam.html.

[7] D. Jurafsky and J. H. Martin. 1st ed., vol. 1, Upper Saddle River, NJ: Prentice Hall, 2000, p. 934.

[8] F. Lu and E. Milios. 1997, "Globally consistent range scan alignment for environment mapping," November 2006, citeseer.ist.psu.edu/lu97globally.html.

[9] M. C. Martin and H. Moravec, "Robot evidence grids," 1996.

[10] H. Moravec, "The Stanford Cart and the CMU Rover," *Proceedings of the IEEE,* vol. 71, pp. 872-913, July 1983.

[11] H. Moravec and A. E. Elfes, "High resolution maps from wide angle sonar," in *Proceedings of the 1985 IEEE International Conference on Robotics and Automation,* 1985, pp. 138-145.

[12] D. Polani, B. Browning, A. Bonarini and K. Yoshida, *RoboCup 2003: Robot Soccer World Cup VII.* Germany: Springer-Verlag, 2004, p. 767.

[13] S. Thrun, "Probabilistic Algorithms in Robotics," *AI Magazine,* vol. 21, pp. 93-109, 2000.

[14] S. Thrun, W. Burgard and D. Fox, Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). MIT Press, 2005.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Craig Martell
   Naval Postgraduate School
   Monterey, California

4. Kevin Squire
   Naval Postgraduate School
   Monterey, California