# The Design of an Autonomous Robot for Exploration of a Simulated Collapsed Mineshaft

Scott Bailey, Matthew Hemming, Gary Hubley, Scott Campbell, MECH 6905 CAPSTONE Project

**This paper describes the design, implementation and experimental results of an autonomous robot as part of the MECH 6905 Capstone Project. The autonomous robot presented in this paper was designed with the intent to navigate and map an unknown area.**

## I. Introduction

Autonomous robots have numerous uses; however, these systems are particularly useful for applications that can be described as dull, dirty and dangerous. One examples includes the mapping and navigation of a collapsed mineshaft. In this instance the use of maned systems could pose a threat to the user and instead autonomous systems may be utilize. Scenario such as these, however, pose a significant challenge as the environment is unknown and thus a robotic system must self-navigate. It is this scenario that is explored in this paper. An autonomous robot was designed to navigate an unknown environment and accurately map objects of varying size and geometries.

## II. Robot Design

### A. Construction

The autonomous robot used for implementing the navigation and mapping algorithm was designed using the Lego Mindstorm EV3 platform. The stock build was used with select modifications to suit the application. The final build can be shown in Figure 1. This design was front wheel drive with an idler caster on the rear for support. In addition, an ultrasonic sensor was used for mapping and bump sensor was included to reduce potential collisions. Modifications from the stock build included:

- A gear train with ratio of 40:8 added to the ultrasonic rotation motor. This allowed for more accurate rotation of the sensor as slower speeds could be used without stalling the motor.
- A bump sensor linkage was used to allow the sensor to be activated by any impact to the front of the robot.
- A Raspberry Pi was included to allow for WIFI communication capabilities to the robot.
- Ev3dev, an MIT Open Source licensed kernel and development kit. Libraries from Ev3dev were also employed for interfacing with the robot's sensors and motors using Python.

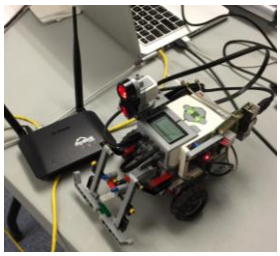


Figure 1 Lego Mindstorm EV3 with Modifications

### B. Sensor Model

Sensor measurement experiments were conducted to determine the behavior of the ultrasonic sensor. To determine the relationship between distance and sensor readings testing was performed at six-inch increments between 5-40 inches with data shown in Figure 2. This distance was chosen as it was approximately half of the testing environment and the maximum required distance for the robot to observe. Testing determined that the measurement values was linear with a relationship of 1 EV3 sensor unit equaling 39.06 inches.

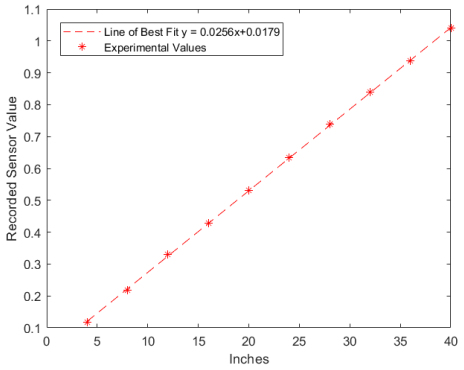



Figure 2 EV3 Ultrasonic Measurements vs. Distance

Next testing was performed to determine the beam angle of the ultrasonic sensor. This would be a crucial element if sensor data was to be properly interpreted. Testing was carried out on a grid with points located at vary angles and distances from the sensor. Objects were then moved into sight of the sensor a specific radius until the sensor could be detected as shown in Figure 3. For this test 100 sensors measurements were collected with an acceptance rate of 80% begin considered found. Results from this test are shown in Table 1&2.

Table 1 Ultrasonic Beam Angle Experiments (Left)

| Distance (Inches) | Cut off Angle (Deg) | Measurement Success |
|---|---|---|
| 10 | 36 | 84% |
| 20 | 39 | 100% |
| 30 | 38 | 100% |
| 48 | 42 | 84% |

Table 2 Ultrasonic Beam Angle Experiments (Right)

| Distance (Inches) | Cut off Angle (Deg) | Measurement Success |
|---|---|---|
| 10 | 40 | 100% |
| 20 | 47.5 | 100% |
| 30 | 40 | 88% |
| 48 | 40 | 84% |

Figure 3 Experimental Sensor Testing

From these experiments the following was concluded:

1) A total beam angle of up to 80 degrees was determined for the sensor at select distances
2) The object closes to the sensor were more often identified by the sensor than values further
3) Objects of tested materials included, plastic, cardboard and glass all appeared to be well received by the sensor
4) Objects of varying geometries did have an impact on the percentage of sensor readings and distance measurements
5) As sensor measurements were taken near corners no significant impacts on measurements were noticed

Based on these finding a method was determined for locating objects. A shown in Figure 4 measurements were taken, however, based on these measurements the EV3 sensor could not precisely identify the location of an object. To mitigate this shortcoming arcs were drawn with a radius equal to recorded measurement and with a beam angle of 60. These arcs would then be written into the occupancy grid by adding a "+INCREASE_AMOUNT" grid value at all points along the arc, all points within the arc were similarly decreased by a "-DECREASE_AMOUNT", such that the ratios between these two values could be tuned to accurately represent how much certainty should be added or deducted based on single readings from the sonar. This process would then be repeated for each angular increment to cover every direction around the robot. Hot spots would thus arise where overlaps in arcs occurred more frequently, implying the likelihood that the source of the sonar return was at the point of intersection of the two arcs.

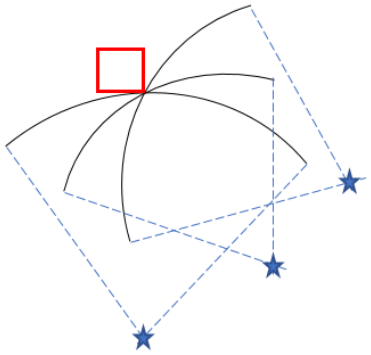


Figure 4 Multiple Sensor Measurements for Object Mapping

With this knowledge of the ultrasonic sensor an algorithm was created to determine if a wall would be identified so that arcs would only be drawn for non-wall objects to reduce false readings. This was achieved through a wall finding algorithm. Based on the location the wall intercepts could be located and then the distance could be found. It was discovered, however, that depending on the angle and distance from the wall. For this a lookup table was created and is shown in Table 3.

Table 3 Lookup Table for Determining Wall Measurements

| | 0 | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|---|
| 0.5 | 0.1700 | 0.1734 | 0.1752 | 0.1950 | 0.3245 | 2.5500 |
| 1 | 0.3210 | 0.3210 | 0.3259 | 0.3291 | 0.3869 | 2.5500 |
| 1.5 | 0.4740 | 0.4707 | 0.4861 | 0.5069 | 0.5370 | 2.5400 |
| 2 | 0.6254 | 0.6341 | 0.6389 | 0.6467 | 0.6676 | |
| 2.5 | 0.7739 | 0.7727 | 0.7759 | 0.8391 | | |
| 3 | 0.9304 | 0.9727 | 0.9976 | | | |
| 3.5 | 1.0771 | 1.1117 | | | | |
| 4 | 1.2229 | | | | | |

Using these findings, the following algorithm workflow was used:

1) 50 measurements are taken at each location to reduce possible outliers. This is achieved by finding standard deviation of the measurements and values outside of one standard deviation are removed. If multiple readings are observed a clustering algorithm is used and multiple arcs are drawn
2) Test is performed to determine if measurement is a wall and if so it is ignored and no arc is drawn
3) If measurement is accepted an addition of one is added into the occupancy in the grid.
4) Through multiple measurements from varying locations, objects are located from the intersection of drawn arcs
5) Post processing filtering is used to determine which values in the table should be considered objects. As the number of sensor measurements increased the total sum in a grid must increase for an object to be considered.

## III. Path Planning and Navigation

Because the environment was known to be approximate rectangular it was determined that the robot would attempt to navigate to the four corners of the environment and take readings in each corner. This method would increase the overall space seen by the robot while also reducing the number of turns required and thus reduce localization errors. The initial path planning was done by checking if the robot could travel to the next way point by analyzing the path on the occupancy grid. If the path was clear, the robot would begin its journey. If the path was not clear, it would analyze a counterclockwise and then clockwise path in 15-degree increments until it found a path. Localization was accomplished using odometry measurements. Furthermore, while not implemented, a Kalman Filter was also development to improve measurement readings and is discussed in this section. In the event that the occupancy grid was not correctly updated, and the robot bumped into an object that triggered the bump sensor, the robot would enter recovery mode.

### *A. Odometry*

Built-in wheel encoder odometry with one-degree wheel rotation resolution allowed the motion of the robot's tires to be used in determining an estimate of the location and heading angle of the center of the robot's axle. A PID controller was used to reduced localization error.

#### *Calculating turns*

To calculate turn angles a desired angle was given to the vehicle and the rotation required for that wheel was calculated using Equation 1. For the given encoders one tick related to one degree.

$$\beta = \frac{\theta * a}{r} \quad (1)$$

θ = angle of the robot
β = angle of the wheel
a = half the length of the robot's axel
r = radius of robot wheel

#### *Going straight*

In order to move straight the number of rotations of the wheel required for a specific distance was calculated. This was then given to the robot until the desired distance had been reached.

### *B. Kalman filter*

By scanning 360 degrees with the sonar the robot could detect and generate up to four perpendicular rays to any unobstructed walls by finding the angles of local minimum distance to the walls of the course. Using the minimum range and angle to the walls we can infer the co-ordinates of robot. Figure 5 shows a state transition diagram for generating measurement updates as inputs to the Kalman filter. Note that in this project, the Kalman filter was completed, but the parameters defining the distribution of the measurement and motion models have yet to be determined. Moreover, the Kalman filter was simplified through the use of a 1D Kalman filter instead of 2D. This allowed for the variance to be more easily determined without requiring a covariance values between x and y direction. This was assumed to be appropriate because the movement was either in the purely x or y directions unless obstacles were encountered. For these scenarios, however, a 2D approach was not used.

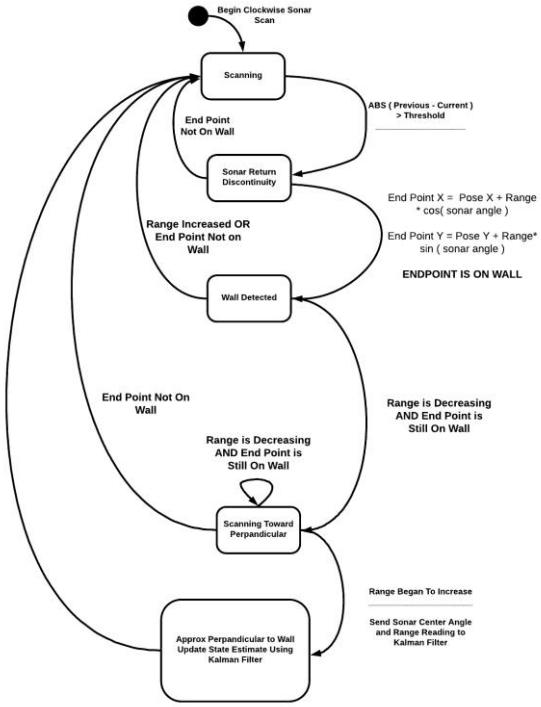



Figure 5 Kalman Filter Work Flow

### *C. Recovery mode*

After the robot has bumped into an object, it immediately stops and the front facing edge is marked blocked on the occupancy grid. The robot then reverses six inches and begins a path planning sequence similar to the initial path planning sequence. The difference being that the robot only checks 16 inches in the direction of the clockwise or counterclockwise angle. Once a 16-inch path is confirmed, the robot will drive 16 inches and check the path to the way point as if nothing had happened previously.

## IV. Methodology

Given the number of unknowns and testing of both mapping and navigation techniques a modular coding approach was taken. This allowed the team to easily modify and test specific aspects of the code. This was reflected in the design as the team work took incremental steps towards the final algorithm. One key aspect of the algorithm was the implementation of occupancy grid and is discussed in this section.

### A. *Occupancy Grid*

By maintaining an occupancy grid, the robot can determine whether it needs to adjust its course to avoid colliding with an obstacle. By incrementally taking sonar readings over 360°, the distances returned by the sonar can be used to infer the relative distance and angles to obstacles near the robot. As discussed in previous sections of this report, the ultrasonic sonar has been characterized to observe returns from a field of view of 60°. Therefore, a large amount of work done to determine how to mitigate the effect of marking entire 60° arcs as occupied.

Three major methodologies were tested and implemented to optimize the tradeoff between reducing false positives for obstacle detection but ensuring the safety of the robot during its mission.

1) Bi-directional sweep patterns to detect discontinuities in the sonar returns allowing the robot to infer the relative angle and distances to the edges of obstacles. By comparing the sonar return value with the previous sonar return, discontinuities will result in large positive or negative differences. With the knowledge of the direction of the rotation of the sonar we can focus the cells to increase and decrease to the angle width of the sonar stepping angle as we know the edge of the obstacle is on one extremity of the beam width or the other.
2) Favoring disproof of obstacles over detections. While generating the occupancy grid, the inferred detection locations along the 60° arc set by the returned sonar range are incremented by an arbitrary weight value. However, the remaining the cells between the end of the arc and the sonar can be assumed not to have any objects and are therefore decremented by a larger value than the arbitrary incrementation weight. This is since the location along the return arc that contained the detected object is less certain than the odds that another object may be within the sonar detection range and not be detected.
3) Failsafe collision detection; if the occupancy grid fails to develop accurately and the robot collides with an object, it is important that the robot have a reactive action planned to stop and retreat from the obstruction, but also that it updates the occupancy grid. The robot will increase the weight of all cells along the front bumper of the robot to a maximum value that represents a near certainty of an obstacle in this location. As the sonar failed to detect the obstruction in previous scans, we mark these cells with high enough weights that they persist even if they are not detected on subsequent scans.

In addition to the three methodologies employed to improve the accuracy of the occupancy grid, there were two important parameters which were defined by weighing the performance (time) cost against the reduction in error. The first criterion was the size of the occupancy grid cells, and the second was the angular stepping resolution to use during the sonar scan. By decreasing the size of each grid cell, the location of marks in the cells that contained the objects causing sonar returns because more accurate. Furthermore, reducing the angle change in the sonar between readings allowed for the edges of obstacles to more accurately be returned. However, each of these parameters come at a noticeable cost in scanning time. As the step width is widened, we of course need to do less iterations of the algorithm to cover the 360°. As the cells grow larger, we reduce computation on a per sonar return basis as there are less cells to collect for the distribution of positive or negative changes to the belief that an obstacle lies within them.

## V. Experimental Results

### A. *Preliminary Testing*

With both the path planning and sensor models determined they were combined and tested. Prior to final testing small scaled testing was performed to determine accuracy of the generated algorithm. In order to do this small a reference space was set up as shown in Figure 6. This was then tested using the previously discussed methods. The results of this method are shown in Figure 7a-c. In addition, results with vary occupancy grid resolution are shown. It can be shown that as predicted increasing the resolution of the grids helps improve the accuracy of the generated map.

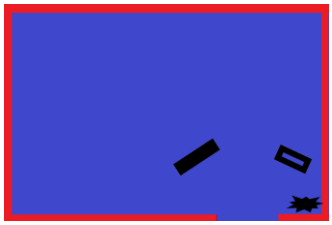
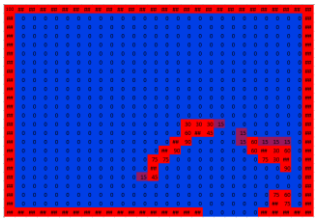
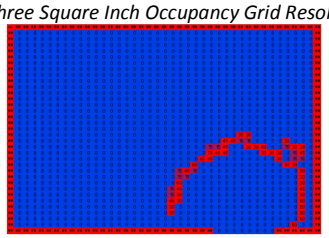


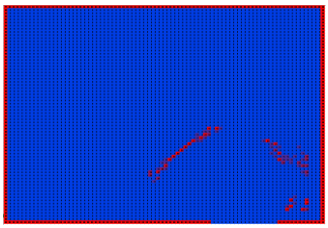
Figure 6 Experimental Test Grid



*a – Three Square Inch Occupancy Grid Resolution*



*b – Two Square Inch Occupancy Grid Resolution*



*c – One Square Inch Occupancy Grid Resolution*

Figure 7 a-c Occupancy Grid from Experimental Testing

### *B. Final Results Using Test Environment*

With preliminary testing performed a full-scale test was completed in the test environment. This environment featured a 69.5'' x 81.5'' rectangular space with six objects of varying geometries, material and size and is shown in Figure 8. The robot entered in the bottom right corner and navigated around the environment mapping the objects using the previously discussed algorithm.

Figure 8 Final Test Environment

With this testing perform an occupancy grid was drawn using the techniques discussed previously. This map is shown in Figure 9. From these results a strong agreement with Figure 8 is shown. This testing demonstrated the effectiveness of the team's approach and its ability to accurately identify the location of objects.

While the algorithm can identify object locations, Figure 9 shows that challenges arose when working to determine the geometry of objects. As shown by the two vertical wooden blocks along the right wall only their corners were identified. This is a due to the fact that throughout the motion of the robot through the grid the corner of these objects almost always remain the closest to the robot and are therefore more likely to be identified. In contrast, however, the orange cylinder in the center was more accurately predicted as measurements were taken from numerous vantage points around the object. It is also important to note that the smaller glass block was less accurately predicted. This is likely due to the material and size of the object.

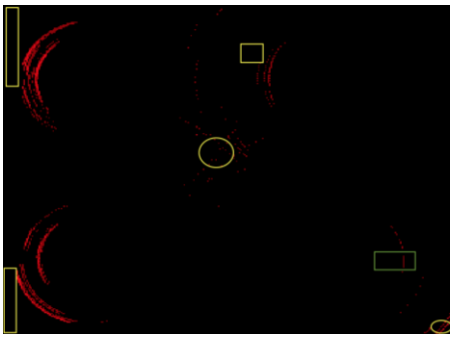


Figure 9 Occupancy Grid of Final Test Environment

## VI. LIMITATIONS AND FUTURE WORK

From testing it was shown that the occupancy grid could be drawn with sufficient accuracy. It was demonstrated that the implemented dead reckoning method proved to provide sufficient accuracy of location. Moreover, suggestions have been made for future improvements using the proposed Kalman Filter. In addition, the method used for interpreting ultrasonic sensor readings and drawing occupancy grid has proven to be a robust method that can account for the shortcomings of the EV3 hardware.

Despite the success of the implemented algorithms improvements are suggested for future iterations of the autonomous vehicle and include:

1) Further development of the motion and sensor models to parameterize the noise variance in each. This information could then be used to implement a Kalman filter.
2) Develop further filtering algorithm to account for the fact that using the current motion scheme objects in corners or along wall edges are mapped less often and are therefore given a lesser weight.