ROBO7001 - Assignment 1

Daniel Addo 19175148
Methuselah Singh 19188409
Sam Trowbridge 19182799
Word Count: 3006

MSc Artificial Intelligence

Contents

1 Introduction

The concept of autonomous navigation has gained traction in recent years predominantly in the field of robotics and various transportation vehicles; this is essentially when a robot moves or travels around an environment in order to complete a particular task. To be more specific, autonomous navigation achieves the concept of a robot travelling around an uncertain environment without the need or interception of an external operator.

A prime example of an intelligent autonomous agent that presents itself is the Anki Cozmo robot, which consists of sensors and actuators such as a camera, cliff sensor and a lifter that allows it to locate and lift cubes in an environment respectively. This enables the robot to be placed or used in uncertain environments where it contains little to no knowledge.

The upcoming task requires a python implementation of a "search and rescue" scenario using Cozmo. The team will implement the sensor models to be used by the robot to detect the cubes. But first we must explain the difference between a sensor model and a sensor. Sensor Models can be described as a summary of the whole sensoring procedure, it describes what the sensors provide and how this information is either limited or enhanced by the environment and other various sensors in order to increase the robustness [1]. However, a sensor reading can be explained as the output from a sensor after it has detected and translated a level of intensity from the environment. The output is then used to compute a probability density function from the sensor model in the form of the mean and variance of our sensor readings.

2 Sensor Model

2.1 Initial Sensor Experiments

Cozmo has two main sensors that have been tested for the purpose of this report: proximity sensor underneath (cliff sensor), and the camera in front. The robot itself comes with three identifiable cubes, and can detect walls based on print out images. These being additional objects for the purpose of the experiments and can come in two different sizes: 200mm and 220mm. To test the limitations of the sensors, a number of real world experiments were conducted.

Cube and wall objects were utilized primarily for the front facing camera. A hypothesis that the sensor readings would increase in accuracy the closer the object, was discussed. Hence, an optimal field of view for the sensor, such that there are distances where it will detect objects with greater accuracy is considered. Therefore, it is expected the sensor will react in a similar manner when detecting cubes and walls.

Cozmo's starting point on the table grid map for all experiments is (100,100), taking this as the origin. All static experiments were run for one hundred frames, and log files were saved using the "monitor_save_all.py" file.

2.1.1 Experiment 1: Cube Distance

The cube was positioned at the furthest point on the table, and slowly moved towards Cozmo in a straight line. At the point where the sensor detected the cube, a real world measurement was taken using a ruler.

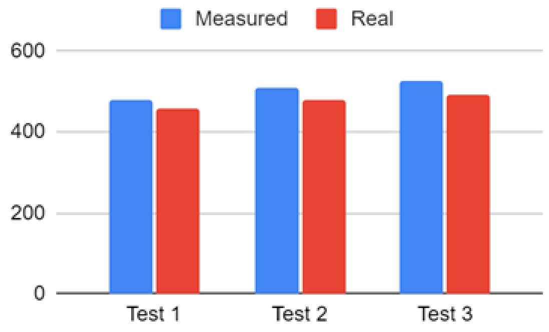


Figure 1: Graphical demonstration of measured distances for cube and actual distances.

It appeared that there were fluctuations at the point of detection, but on average the sensor measured distance was 504.95mm, and the average real life measured distance was 478mm.

2.1.2 Experiment 2: Wall Distance

This was identical to experiment 1 using a wall object.

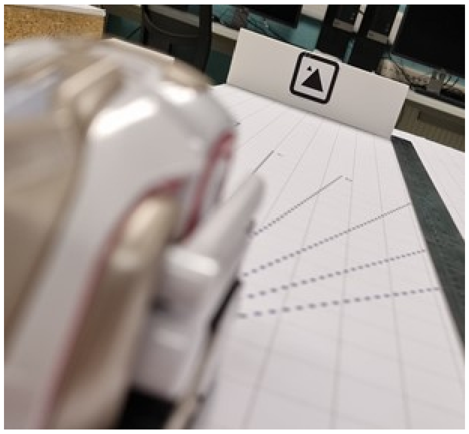


Figure 2: Image of wall from Cozmo's view.

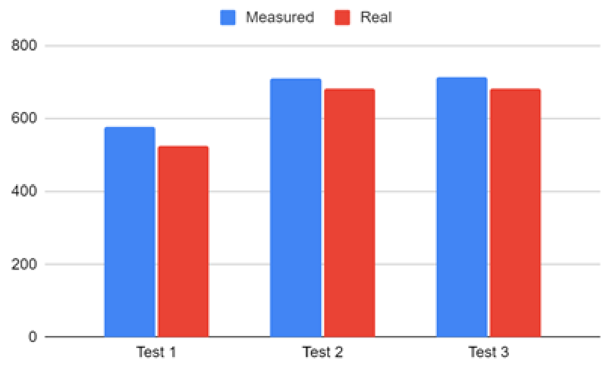


Figure 3: Graphical demonstration of measured distances for wall and actual distances.

Our initial prediction was that the distance of detection would be similar to that of the cube, however this was not the case. The experiments demonstrate that the average measured distance of the wall according to the robot sensor was 667.48mm while the real world distance was 629.67mm. One reason as to why this is the case may be that the logo printed on the walls is larger than that on the cubes, and is printed on a clear white background, making it easier to locate.

2.1.3 Experiment 3: Rotate wall until Cozmo does not detect

This experiment was designed to establish how distorted the wall images can be for them to still be detected. The wall was placed at a set distance from Cozmo and was manually rotated until no longer detected. A real world measurement was taken using a protractor: 64.43°, or 1.12 radians.

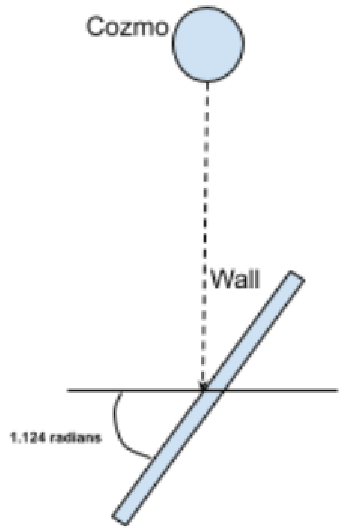


Figure 4: The angle of the wall at which it is no longer detectable.

2.1.4 Experiment 4: Cube at increasing angles with constance distance (295mm)

The cube was placed at 295mm from Cozmo, firstly at a straight line from the robot. The cube was then measured at 10°, 20°, 30° and 40°, which is essentially altering the y coordinate of the cube. The results indicate that the maximum angle from the central line of vision is around 25°, creaing a cone shaped field of view for Cozmo.

2.1.5 Experiment 5: Cliff Height

This experiment was designed to find the threshold of when Cozmo detects a cliff based on the height. At the 30mm point, Cozmo was able to detect the cliff, therefore his threshold for cliff detection based on height would be between 20mm - 30mm.

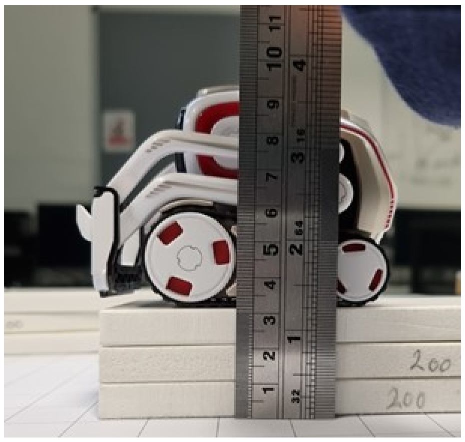


Figure 5: The cliff height at which a cliff is detected.

2.1.6 Experiment 6: Cliff Gap

The approximate real world size of the cliff sensor on Cozmo, is around 3mm. In this experiment, the gap between two cliffs is increased to the point of detection; tests were conducted at 3mm, 4mm, and 5mm gaps. At 3mm no cliff was detected despite being the same size as the sensor; the minimum width of a gap was found to be 5mm.

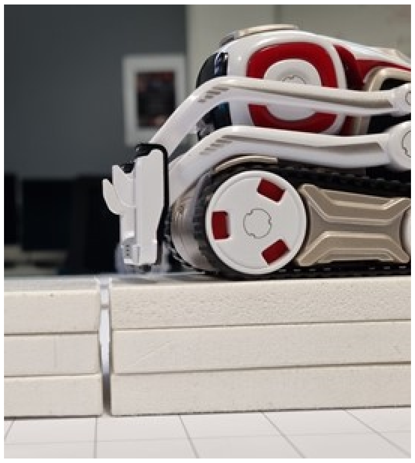


Figure 6: The cliff gap at which a cliff is detected.

2.2 Analysis of Experiments

By analysing the data, a “cone” of view can be established for the robot. This vision cone is the area at which objects are detectable, taking into consideration the blind spot right in front of the camera, where the objects are too close to the sensor.

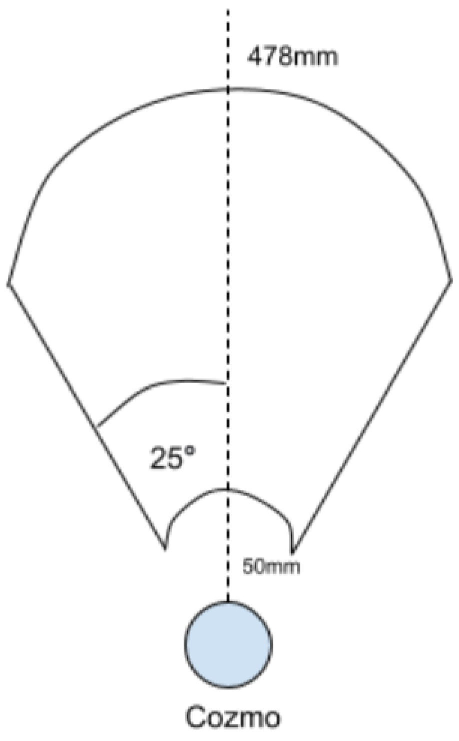


Figure 7: Cozmo’s vision cone.

Further examination of the data output by the sensor demonstrates that there is greater variation of the measurements at larger distances from Cozmo. From 100mm to 450mm, at equal 50mm intervals, Gaussian distributions were created. These intervals were chosen due to them being the boundaries of the vision cone.

The mean and variance that are used for these distributions relates to the average distance recorded. As expected, the variance is greater at distances further from Cozmo, and a much smaller variance the closer the object moves to Cozmo. Furthermore, this correlates with the probability that the cube is visible as seen in Figure 8. This Figure demonstrates how a sharp increase of probability occurs as the cube is closer, and then as it moves towards the extent

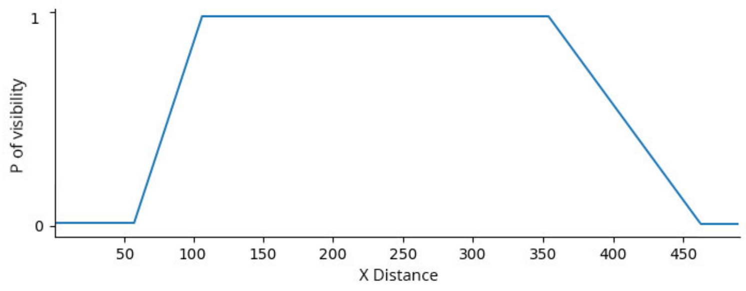


Figure 8: This figure demonstrates the probability of visibility for a cube as the distance increases.

of Cozmos vision cone the probability decreases. It is worth noting that the probability will never actually be 0 or 1, as there is always a small probability that it is not correct.

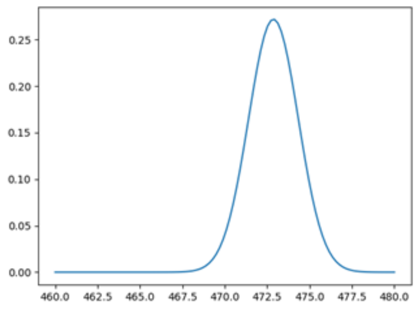


Figure 9: Mean of distances against variance at 450mm.

The linear plot of mean distance against variance (Figure 12) demonstrates the decrease in variance as the cube is moved closer to Cozmo. Each experiment ran for the same number of frames, however it is worth noting that at the 450mm distance not every single frame detected a cube. Therefore, distribution at this distance has a smaller dataset than the others, which could be why there is a slightly better variance. Lastly, Cozmo sensor readings were at least 12cm off from his actual position due to internal Cozmo data gathering errors.

2.3 Approach to Sensor Model and Implementation

Our sensor model is designed such that it takes in the measured readings of the object from the robot, as well as the true distance, as defined by the user. The robot's position is also passed into the model. Using the robot position

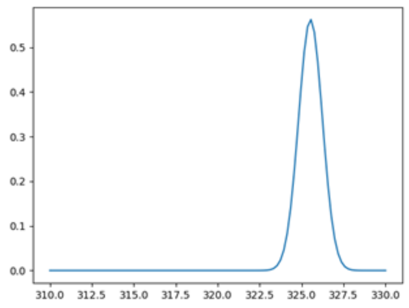


Figure 10: Mean of distances against variance at 300mm.

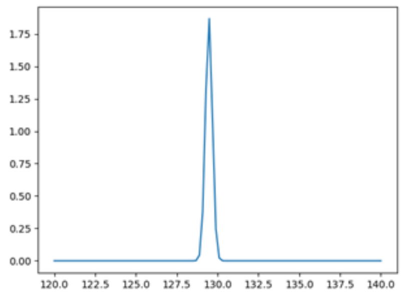


Figure 11: Mean of distances against variance at 100mm.

and the measured position of the object, we work out euclidean distance using Pythagoras's theorem. If Cozmo is given coordinates (x1,y1), and the cube is given (x2,y2), the following equation can be used to work out the distance:

$$Distance = \sqrt{((x2 - x1)^2 + (y2 - y2)^2))}$$

A variation of the tangent function, arctan2, is used to calculate the angle theta θ. This is similar to arctangent, however it takes two arguments, allowing it to be used on a euclidean plane. It is important to ensure that this function is implemented as it allows us to determine the actual coordinate; the euclidean distance calculated using the above formula is not enough to correctly locate the cube. The same distance would hold true for a cube placed at a coordinate with -θ from Cozmo.

$$Angle = arctan2(y2 - y1, x2 - x1)$$

Figure 13 represents Cozmo with the circle at (x1,y1) and the cube with the square at (x2,y2). The larger dashed arrow represents the direction that Cozmo

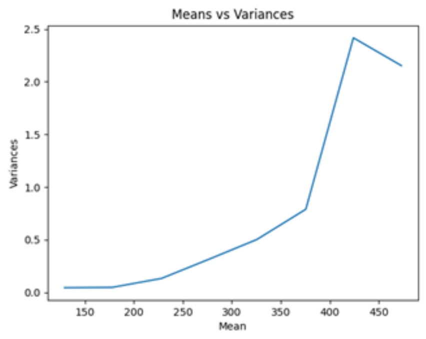


Figure 12: Plot of means against variance at our chosen intervals.

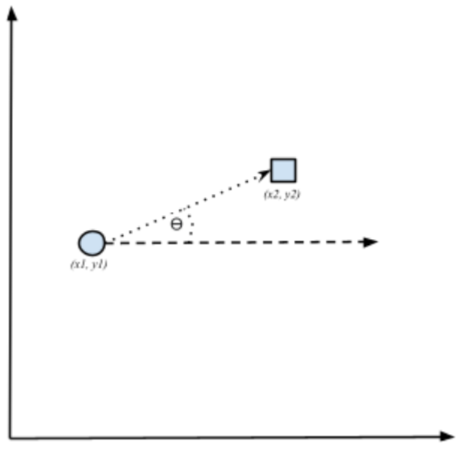


Figure 13: Cozmo's position from cube.

is facing, the cube being placed at an angle θ from this central line of vision.

The mean distance and angle is calculated and compared to the true values which are given to the robot. Having established "waves" of Gaussian distributions using our experiments, i.e. the 50mm intervals, we determine which of these eight waves the cube is closest to. For each frame, the mean and variance is updated using the newly processed distance.

Following on from this, the probability of the measured reading being correct is calculated; this is done by taking the difference between the true and our calculated position, and dividing by the true. This step is essentially finding out how close the sensor output is to the true value. The probability for both the distance and the angle are calculated; considering Bayes theorem, two values are then multiplied together. The sensor model then returns the probability, updated variance, and updated mean to be used to plot a Gaussian distribution[2].

3 Mapping

Mapping is the act of creating a model of the local environment, this is a vital system for any autonomous or remotely operated machine. Customarily, this also provides a visual aid to make the information more legible for people to understand. In addition, this provides the ability to observe how effective the current mapping system is and its accuracy. The mapping process predominantly uses the machine's sensors to acquire information about the environment surrounding it and record possible highlights or obstacles. This data can then be used for multiple actions, such as establishing landmarks, to assist navigation and by finding obstacles that may impede progress towards a particular objective. It can also be used to find a mission objective or pathway that's safe to transverse. The sensor model is applied to this data beforehand to process the probability of reliable and accurate information.

The most common forms of maps consist of: geometric/topographic, topological and schematic. Geometric maps display detailed information regarding objects, buildings and surfaces within an area maintaining the distances between these objects. Topological maps focus on critical locations, landmarks and routes, and often distorting the distances in between them. Schematic maps provide what could be considered a framework of an area with little detail and distorted distances. Hybrids are also a possibility by combining multiple mapping methods. This project needs to represent how accurate Cozmo's sensor readings are and display objects found within an environment, meaning a geometric map is more applicable. The chosen method of visualising this was an occupancy grid map, "Occupancy grid maps address the problem of generating consistent maps from noisy and uncertain measurement data, under the assumption that the robot pose is known [3]".

3.1 Map Designs

3.1.1 Splitting Maps into Tiers

Various topographical maps were first designed in lucidchart and then implemented in our python code. The notion behind the maps was to split the level of difficulty into 5 tiers;

- Tier 1 - Testing the sensor model and mapping algorithm.
- Tier 2 - Sensing more than one Cube.
- Tier 3 - Sensing Cubes in respect to Distance.
- Tier 4 - Sensing Cubes in respect to Obstruction.
- Tier 5 - Sensing Cubes in respect to perception bias.

Each tier consists of up to two maps, this will allow different comparisons, which can derive clear points, deviations and different complications as listed above.

3.1.2 Tiers Breakdown

Tier 1 - Testing the sensor model

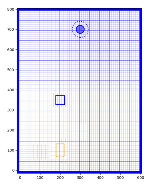


Figure 14: Tier 1 map

Before the Cozmo robot is tested on more than one cube simultaneously, it is placed in a grid with one cube to measure its efficiency, mapping algorithm and sensor model, respectively.

3.1.3 Tier 2 - Sensing more than one cube

The concept for this tier is to improve the efficiency of the sensor model gradually. For example, in the testing stage above, the Cozmo robot only detected one cube. This tier will increase the level of difficulty and include an extra cube with two walls. The idea behind this tier is to analyse any changes in the robot's behaviour with regard to new inclusions.

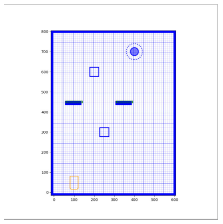


Figure 15: Tier 2 map 1

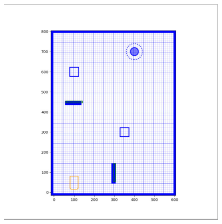


Figure 16: Tier 2 map 2

3.1.4 Tier 3 - Sensing cubes in respect to Distance

The idea behind this tier is to distance the cubes as far away from each other as possible. The map is constructed in this manner to begin testing the efficiency of the sensor model, with Cozmo placed in between the cubes. The maps are designed to observe how the sensor model can depict the distance between Cozmo and the cube. This will essentially begin testing Cozmos efficiency in sensing cubes regardless of distance. This will also show how effective our mapping implementation is at showing different levels of probability distribution. [h]

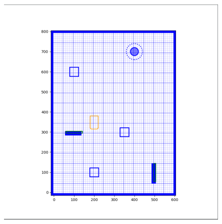


Figure 17: Tier 3 map 1

3.1.5 Tier 4 - Sensing Cubes concerning Obstruction

The concept of this tier is to test the sensors of Cozmo concerning obstruction. Therefore, the map is constructed to use the walls to cover the target position from visual sensors of Cozmo. This is done to test the efficiency of the wall

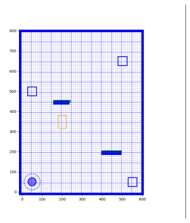


Figure 18: Tier 3 map 2

sensor model and its ability to spot a wall and manoeuvre its way around to reach the target position. These maps were designed with future functionality and objectives in mind.

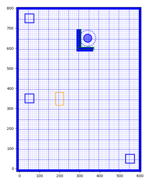


Figure 19: Tier 4 map 1

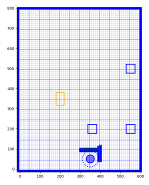


Figure 20: Tier 4 map 2

3.1.6 Tier 5 - Sensing Cubes concerning Perception bias

The primary notion behind tier 5 is the opposite of the "Tier 2" above. The team instead placed the cubes quite close to each other. This was implemented to test Cozmo's sensors concerning perception and a case of perception bias or ambiguity. To be more specific, the purpose of constructing this map is to prevent Cozmo from assuming that cubes close to each other are indistinguishable even when in different positions or angles. This serves as a final test as to how the developed functionalities can cope with a confined space and busy environment.

3.1.7 Benefits and Drawbacks

Designing and implementing various maps and ranking them in terms of difficulty enables a precise analysis of how Cozmo reacts to new changes and adapts to them accordingly. On the other hand, in terms of a drawback, the cube's orientation concerning the position of an approaching robot could be an issue.

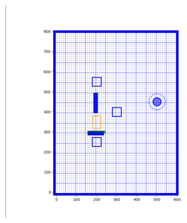


Figure 21: Tier 5 map 1

For example, in a particular scenario where Cozmo approaches a cube at an angle where it senses more than one symbol on the cube, the robot could find difficulties clarifying the cube's orientation and registering it as one cube.

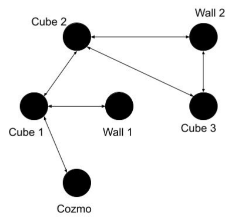
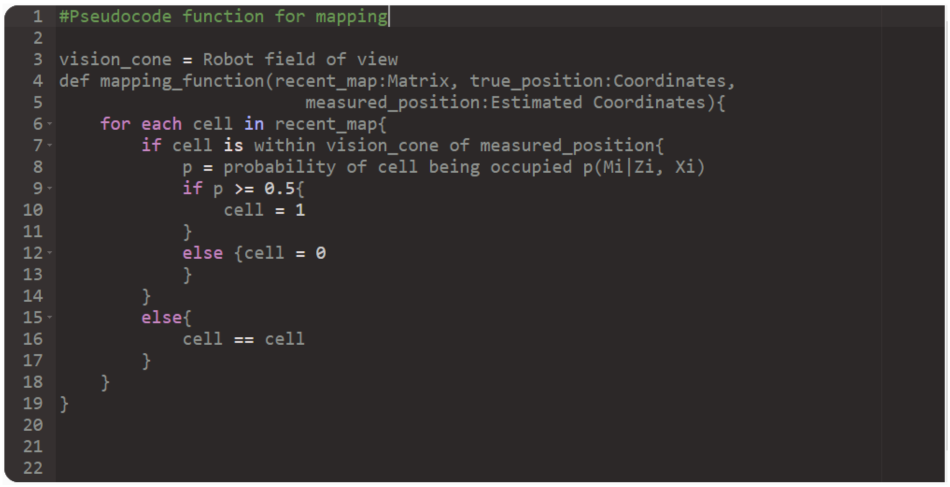


Figure 22: Graphical representation of topological mapping.

3.2 Implement Occupancy Grid Mapping

Due to time constraints, the mapping algorithm has not been implemented. However, this section will explain how we plan to achieve this. A primary base map must first be provided for an occupancy grid map, which has already been assembled in the base files provided. What is required is to create a posterior update function of objects and obstacles within the established map, obstacles consist of walls and a cliff, whereas objects are Cozmo cubes. "Gold standard of any occupancy grid mapping algorithm is to calculate the posterior over maps given the data" [3].The number and position of obstacles and objects varies depending on the maps' tier level previously constructed to test Cozmo's capabilities.

The pseudo-code below shows that the grid map update function needs the true object position, measured position, and most recently updated map. Next is to loop through all cells in the map and within this loop if the map cell is within the measured position including all visible cells in the distance prior to the object. The probability algorithm must be executed. For each cell within Cozmo's range of vision, this algorithm will calculate the probability of it being occupied. Depending on which of the two the probability is closest, the cell will then be updated with a 1 or a 0, 1 for occupied and 0 for unoccupied. If a cell is not in the cone of vision, it will remain the same value previously recorded. Once this loop is finished, the new matrix of 0's and 1's is returned to update our map and visual aids.



```
#Pseudocode function for mapping

vision_cone = Robot field of view
def mapping_function(recent_map:Matrix, true_position:Coordinates,
                        measured_position:Estimated Coordinates){
    for each cell in recent_map{
        if cell is within vision_cone of measured_position{
            p = probability of cell being occupied p(Mi|Zi, Xi)
            if p >= 0.5{
                cell = 1
            }
            else {cell = 0
            }
        }
        else{
            cell == cell
        }
    }
}
```

Figure 23: Pseudo code of Mapping Algorithm

However, despite the sensors, sensor module, and mapping algorithm, maximum accuracy is unachievable. Therefore as shown in an example below, what is returned is a distribution of an area in that it's likely an object exists. Also, from what was discovered during the sensor experiments, the size of the distribution area would highly depend on the object's location. The closer to the edge of Cozmo's cone of vision boundaries, the measurement data acquired possesses higher error ratings increasing the noise, therefore the area of object probability will be larger. On the other hand, as Cozmo gets closer and the object moves towards the centre of its vision cone, the data will be more accurate, producing a smaller probability area. Despite these issues and Cozmo's inaccurate sensory data, occupancy grid mapping is an effective mapping method for this experiment, because it takes place in a controlled linear environment and occupancy grid mapping is relatively simple to implement; furthermore it's particularly efficient at handling noisy, uncertain measurement data.

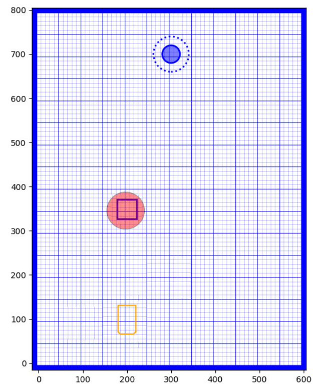


Figure 24: Object Area Distribution

4 Conclusion

After conducting initial experiments on Cozmo, we understood the boundaries around the sensors and developed a rough estimation of the visual abilities. These also provided us with adequate datasets to build a plot for object visibility over distance.

We developed a working understanding of the source code provided, fixing minor bugs that appeared, and established where our functions and code would sit. Additionally, we spent a significant amount of time developing knowledge and understanding of the sensor, probabilistic sensor models, and how these relate to robotic mapping. After building this knowledge base, a plan of action was agreed on how to approach the implementation.

Throughout this assignment, we developed extensive knowledge on how inaccurate sensors can be in certain situations, and methods that can be used to account for this. In addition, we have learned various ways of mapping an environment, including detecting objects and obstacles that may be present. As a result, we have a foundational understanding of building upon other robotics and autonomous system applications.

Overall, the report demonstrates a working sensor model using Gaussian distributions to provide an appropriate probabilistic model. Our intention is to use this model and our mapping plan to allow Cozmo to navigate an environment. However, due to time constraints, this implementation was not completed. If possible to reapproach this assignment, more time would be allocated to understanding the specification and expectations. Also, greater research could be conducted regarding the methods of implementation, as this was the most time consuming section.

5 Reference

References

[1] Durrant-Whyte H.F. *Sensor Models and Multisensor Integration*. Springer, 1990. DOI: `https://doi.org/10.1007/978-1-4613-8997-2_7`.

[2] Siegwart, R, Nourbakhsh, IR, and Scaramuzza, D. *Introduction to Autonomous Mobile Robots*. The MIT Press, 2011.

[3] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.