Assignment 1 – Creating a 2D Map of an Unknown Environment

Introduction:

This report details the design and implementation of a mobile robot whose task was to create an accurate 2D map of an unknown environment. The selected mobile robot "PeopleBot" and its surrounding map, were created using the MobileSim simulator software. The behavior and mapping of the "PeopleBot" was implemented using the Aria C++ library in Visual Studio. The objective of the application was to design a C++ solution to allow the robot to navigate the created scene and generate a 2D map using the occupancy grid method. The solution was realized by taking elements of the work from previously completed labs and combining together to create the Occupancy Grid Map. To test the quality of the solution, a number of scenes (maps) were created using the Mapper3 software application and loaded within the MobileSim environment. The MobileSim environment contained the "PeopleBot" robot within each of the different maps, which in turn allowed for the generation of the corresponding Occupancy Grid maps.

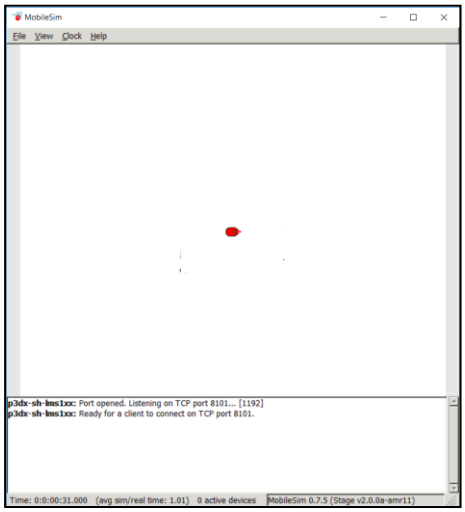

Figure 1 – *MobileSim Simulation Environment*

Map Construction Technique:

Maps 1 & 2 – Circular/Square Boundary Room:

- Several Mapper3 scenes were supplied to complete this assignment. For each of these maps, it was observed that the map was not centred about the origin. The first map that was created for testing in the MobileSim environment was a room with a circular boundary (Figure 2 below). The second map that was created for testing in the MobileSim environment was a room with a square boundary (Figure 2 below).

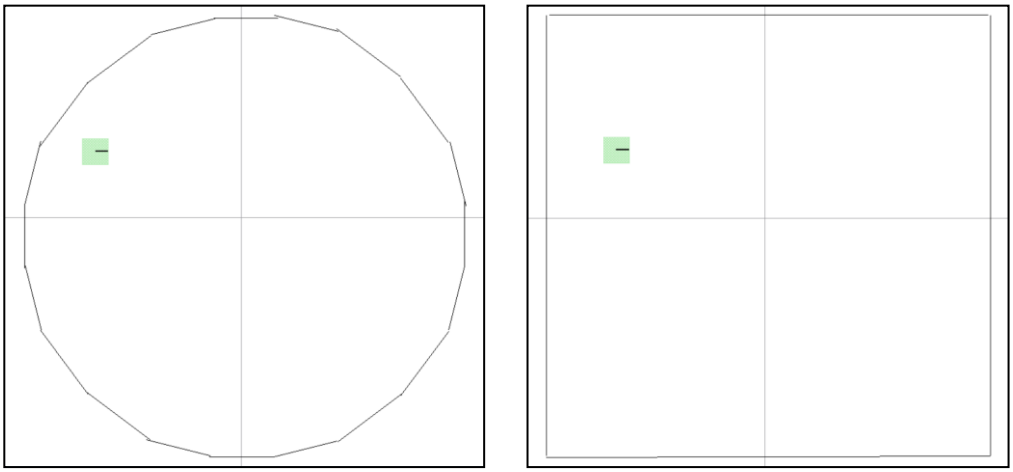

Figure 2 – *Circular & Square Boundary Maps – Centred on Origin*

Mapping Method:

- The strategy to solve this mapping application was to use two front ultrasonic sensors and two side ultrasonic sensors to provide distance feedback to the robot which could in turn be used to program the behavior of the robot in order to accurately perceive its environment. Figure 3 below, shows the four chosen ultrasonic sensors used to solve this application.
- There are multiple methods for solving a robot mapping application based on sensor feedback. The majority of these methods use the raw sensor data to map the surrounding environment. One of the biggest problems with these methods is that they are binary in nature i.e. point in space is occupied or not. Depending on the environment where the mapping robot is deployed, a number of factors can affect the result and cause inaccuracy about the mapped environment such as weather and object material properties.

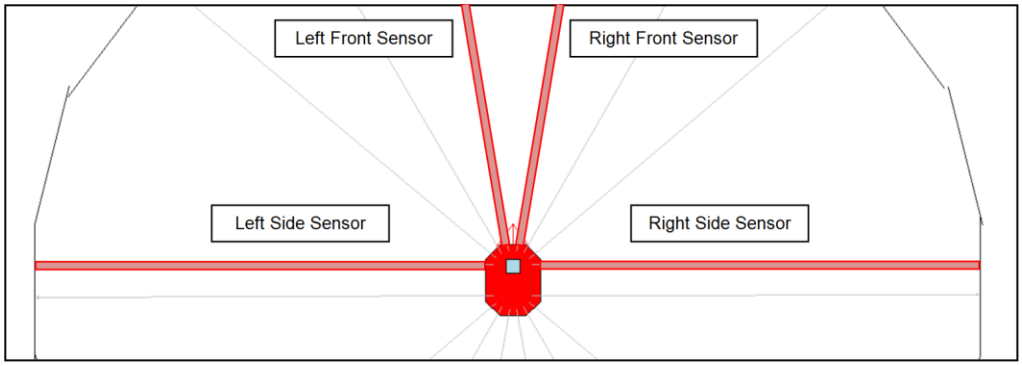

Figure 3 – *Sensor Model Strategy*

- The method chosen to solve this application was a probabilistic model. As this task involves the mapping of an ***unknown environment*** the factors mentioned above are present and are likely to cause issues with the generated map. As demonstrated in Figure 4 below, for each sensor reading there are three conclusions: 1. The probability that the area before the detected object is low, 2. The probability that the area around the detected object is high, 3. The probability that the area further than the detected object is unknown (currently). Using this method, a Sensor *"Sonar"* Model was created to get a Probability value for each sensor reading.
- As the Probability method above only applies to one sensor reading, the next step was to deploy this Probabilistic model iteratively over time to create an accurate 2D map. Bayes' Recursive Theorem was used to do this, by combining ***P(H|E$_t$)*** the probability that a grid cell is occupied given sensor reading, ***P(E$_t$|H)*** probability that sensor would return considered value if the grid cell was occupied and also storing the previous probability readings. The implemented Recursive version of the theorem is shown below:

$$P(H|E_t) = \frac{P(E_t|H) \times P(H|E_{t-1})}{P(E_t|H) \times P(H|E_{t-1}) + P(E_t|\neg H) \times P(\neg H|E_{t-1})}$$

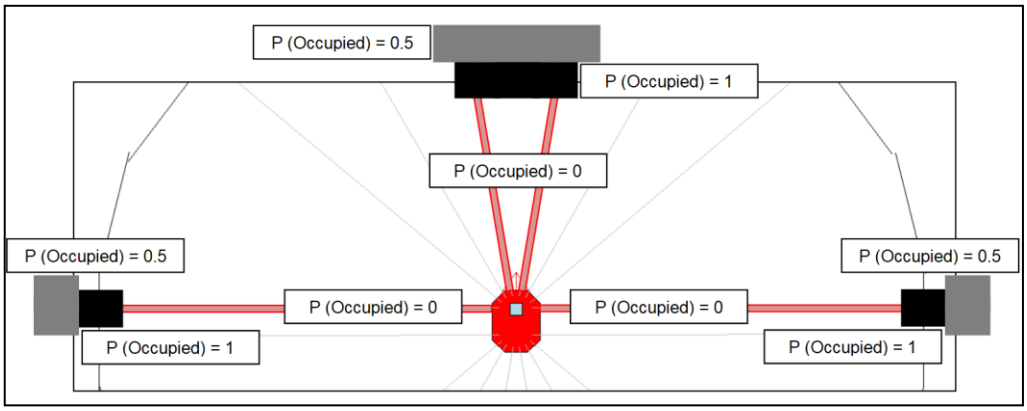

Figure 4 – *Probability Model – Occupancy Grid*

Software Implementation:

Initial Software Design:

The initial design software design for this application involved the implementation of a new ArAction (Aria function) to be performed by the robot. The concept behind this approach was that the solution would "modular" by nature meaning that the addition of further ArActions could easily be performed for each of the sensors or if there were several robots. The structure of the initial design was as follows:

- Main.cpp initializes all Aria components, executes ArActions (including new SonarModel ArAction) and displays graphical results.
- SonarModel.cpp gets all relevant Sensor and Robot values, rotates and translates values, determines Bounding Box, determines the Probability for that Sensor Model and updates the OccupancyGrid array using Bayes' Recursive Theorem.

Final Software Design:

The final design software design for this application involved the combination of all functionality within the main.cpp file. This approach was chosen as for the initial design, each time the ArAction was executed, its desiredState was reset meaning that the previous OccupancyGrid results were cleared and the recursive version of Bayes' Thereom could not be used. This method once developed was a lot clearer and the process of adding further Sensors to map the grid was straight forward. The structure of the initial design was as follows:

- Main.cpp initializes all Aria components, executes ArActions (including new SonarModel ArAction).
- While loop created gets all relevant Sensor(s) and Robot values, rotates and translates values, determines Bounding Box(s), determines the Probability for each Sensor Model and updates the OccupancyGrid array using Bayes' Recursive Theorem.
- Once the program has been executed greater than a set threshold, an OpenGL window was created to display the Occupancy Grid generated during the robots travel.
- **Used Libraries:** Eigen 3.3.7 – Matrix operations for rotating/translating points. SFML 2.5.1 – OpenGL graphics for plotting Occupancy Grid.

FIRST IMPLEMENTATION – PSEUDO CODE

```
Main.cpp

{
      Aria definitions
      Aria actions
      {
            **Execute "Sonar Model"
      }
      Open SFML window
      {
            **Update existing pixel colours within Bounding Box
      }
}


SonarModel.cpp

{
      ArAction definition "Sonar Model"
      Get Robot X,Y,Theta
      Sensor 1
      {
            Get Sensor X,Y,Theta
            Set working range
            Get Sensor Reading
            Rotate and Translate
      }

      Determine Sonar Cone Bounding Box
      Set initial probability for all cells = 0.5
      Store previous Occupancy Grid results

      While(true)
      {
            for Each Cell in Bounding Box
            {
                  **Bayes Theorem (Recursive form)

            }

            if(execution time > x)
            {
                  **Draw resulting Occupancy Grid
            }
      }


}
```

FINAL IMPLEMENTATION – PSEUDO CODE

```
Main.cpp

{
      Aria definitions
      Aria actions
      {
            **Execute "Sonar Model"
      }
      Open SFML window
      {
            **Update existing pixel colours within Bounding Box
      }

      ArAction definition "Sonar Model"
      Get Robot X,Y,Theta
      Sensor 1
      {
            Get Sensor X,Y,Theta
            Set working range
            Get Sensor Reading
            Rotate and Translate
      }
      Sensor 2
      {
            Get Sensor X,Y,Theta
            Set working range
            Get Sensor Reading
            Rotate and Translate
      }

      Determine Sonar Cone Bounding Box
      Set initial probability for all cells = 0.5
      Store previous Occupancy Grid results

      While(true)
      {
            for Each Cell in Bounding Box of Sensor 1
            {
                  **Bayes Theorem (Recursive form)

            }

            for Each Cell in Bounding Box of Sensor 2
            {
                  **Bayes Theorem (Recursive form)

            }

            if (execution time > x)
            {
                  **Draw resulting Occupancy Grid
            }
      }


}
```

Figure 3 – *Pseudo Code for First Implementation vs. Final Implementation*

Experimental Results:

Map 1 – Circular Boundary Room:

- For Map 1 (Circular Boundary Room), the robot was placed in a number of locations within the room and the program was executed multiple times. Due to the ArActions implemented in the program, the robot continuously followed the boundary which meant that the results would be a lot more desirable. This can be seen in Figure 4 below. On the left, is an early Occupancy Grid result from a test run. On the right, is the best result obtained from testing. The Z axis was scaled in order to help with the visualization of the results.
- *Room for Improvement:* In order to improve results further, the additional proximity sensors of the robot could be used to acquire more data points about the Map.

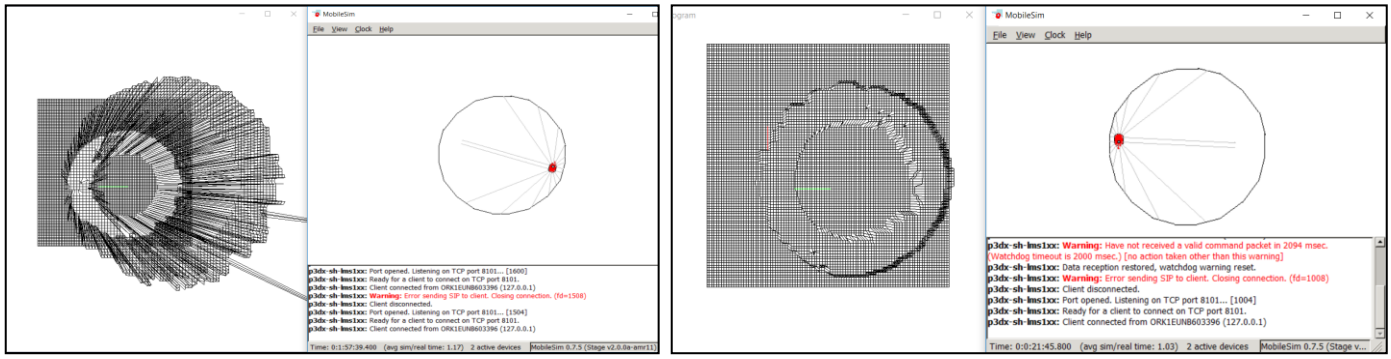

Figure 4 – *Occupancy Grid Map for Early Result vs. Best Result*

Map 2 – Square Boundary Room:

- For Map 2 (Square Boundary Room), the robot was placed in a number of locations within the room and the program was executed multiple times. Due to the ArActions implemented in the program, the robot continuously followed the boundary which meant that the results would be a lot more desirable. This can be seen in Figure 5 below. On the left, is an early Occupancy Grid result from a test run. On the right, is the best result obtained from testing. The Z axis was scaled in order to help with the visualization of the results.
- *Room for Improvement:* In order to improve results further, the additional proximity sensors of the robot could be used to acquire more data points about the Map.

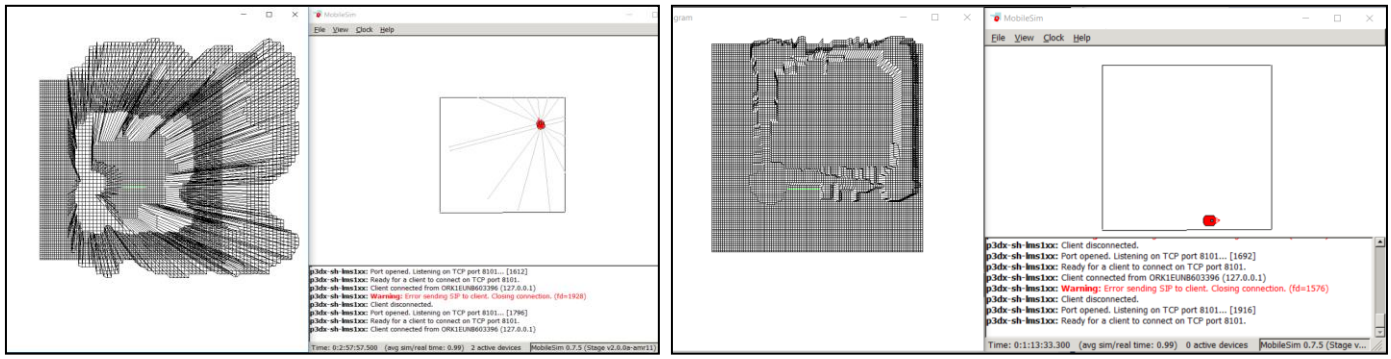

Figure 5 – *Occupancy Grid Map for Early Result vs. Best Result*

Conclusion/Learnings:

For the implementation of this robotic mapping application, the objective was to create an accurate 2D representation of the robots surrounding environment. As previously discussed, there are three main methods for completing this task. This could have been done by generating a set of 2D coordinates, generating a set of line segments using the RANSAC method or by generating an Occupancy Grid. The first two methods are both binary in nature and the accuracy of the generated maps are susceptible to other environmental factors such as weather and object material properties. For this reason, the method that was chosen to successfully create an accurate 2D map of the robots unknown surroundings was the Occupancy Grid method.

In order to implement the Occupancy Grid method, a Probabilistic Sensor Model was created. During execution, if the current grid cell was within the Bounding Box of the Sensor then its Probability was updated using the Sensor Model. This method was then repeated within a while loop to allow the robot and its sensors to generate a 2D array of probabilities. After the program executed for a set length of time, the resulting Occupancy Grid was then plotted using the OpenGL library.

From the testing and results stage of this implementation, it was clear that the generated Occupancy Grid maps gave an accurate representation of the unknown environment. The circular boundary map result was very accurate. The rectangular boundary map result was also accurate. Future works will allow for additional maps to be created with more difficult boundaries and also with obstacles contained within. In order to build on the current solution and accurately map more challenging environments, it is key to use more of the available ultrasonic sensors to generate the 2D Occupancy Grid map.

Bibliography:

1. Freiburg – AIS. Cyrill Stachniss – Robot Mapping Grid Maps. 2016. [ONLINE] Available at: https://ais.informatik.uni-freiburg.de/teaching/ws13/mapping/pdf/slam10-gridmaps.pdf. [Acessed 08 March 2020].

2. Penn Engineering – Robotic Mapping (Occupancy Grid Mapping, Handling Range Sensor). 2018 [ONLINE] Available at: https://www.youtube.com/watch?v=9w5_XPyDSlA. [Accessed 11 March 2020].

3. Alberto Elfes – Using Occupancy Grids for Mobile Robot Perception and Navigation. 1989 [PAPER] Carnegie Mellon University. IEEE ISSN 0018-9162.

4. Burguera, Antoni, Gonzales, Yolanda and Oliver, Gabriel, 2009. [PAPER] Sonar Sensor Models and Their Application to Mobile Robot Localization. Sensors 2009. ISSN 1424-8220, p.g. 10217-10243.

5. Souza, Anderson & Maia, Rosiery & Aroca, Rafael & Gonçalves, Luiz. 2013. [PAPER] Probabilistic robotic grid mapping based on occupancy and elevation information. 2013 16th International Conference on Advanced Robotics, ICAR 2013. 10.1109/ICAR.2013.6766467.

6. Meyer-Delius, Daniel, Maximilian Beinhofer and Wolfram Burgard. 2010 [PAPER] Occupancy Grid Models for Robot Mapping in Changing Environments. Proceedings of the 26th AAAI Conference on Artificial Intelligence.