Noushin Islam Student No. 201508438

Assignment 1 – COMP329

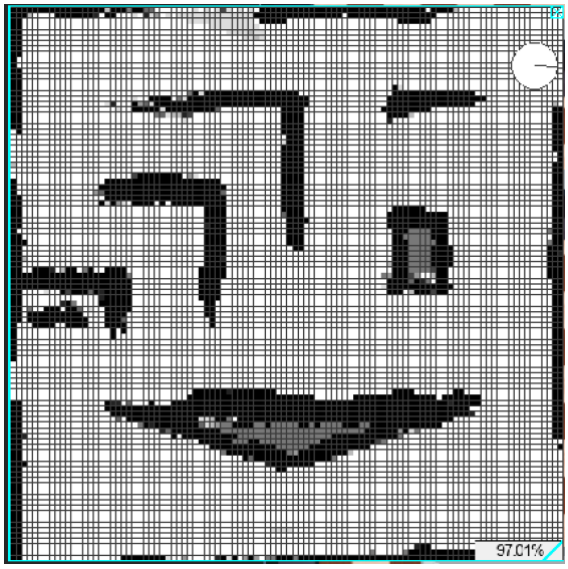
Report

Assignment Brief:

The aim of the programming assignment is to take a partially completed Webots project and extend the controller and navigation code included in the project to navigate around the arena, so that an Occupancy Grid map can be generated for that arena.

Result:

Final Occupancy Grid map of size 100x100 after 3:50 min of simulated time.



As seen above, all obstacles have been identified well except on (left side). Around 70% of the walls around the map have been identified. Almost no issues with prior probability and no gray areas.

Implementation:

Initially, I was planning to implement a wall following approach with PID controllers as taught in Lab5, however, for this project, I found it much easier and more efficient to implement a set of predefined waypoints.

Waypointing is a method of robot navigation where the robot is given a set of specific locations, or waypoints, to visit in a particular order. The wall following approach involves the robot following the walls of a space in order to explore it.

One reason why waypointing is better than wall following is that it allows the robot to have a more structured and planned approach to exploration. With waypointing, the robot can be programmed to visit specific locations in a predetermined order, which can help it to cover more ground and explore the area more thoroughly in less time. In contrast, wall following can be more random and may not allow the robot to cover as much ground or explore as effectively.

Additionally, waypointing can be more efficient in terms of energy consumption and computational resources, as it does not require the robot to constantly sense and react to its surroundings in the same way that wall following does.

To implement my solution, I first identified and marked a series of points on the map that I wanted the robot to explore. These points, or waypoints, were specified solely by their x and y-coordinates (theta was set to 0).

The process of guiding the robot through these waypoints is relatively straightforward:
First, the robot is set to move forward at a predetermined velocity towards its first waypoint. Before initiating any movement, the program calculates the time required for the robot to travel from its current position to the next waypoint. This allows the robot to determine when it has reached the next point in its path.
Next, the program calculates the angle that the robot needs to turn in order to face its next waypoint, as well as the time this rotation should take. Based on this information, the robot rotates around itself to face the next waypoint.
From this point, the process is repeated for each subsequent waypoint in the path. The program calculates the angle and distance that the robot needs to travel from its current position to the next waypoint, and the robot moves accordingly. This process is repeated until the last waypoint is reached, after which no further movement is required.

The code is thoroughly documented with clear and concise comments that explain the purpose and function of each part of the program.

Reflection:

At one point during the development of my code, I encountered a frustrating issue where my robot was turning the wrong angle after reaching each waypoint. Despite spending a significant amount of time attempting to identify the cause of this issue, I was unable to do so. It wasn't until later that I realized that I had been using the wrong calculation formula, which had caused me to waste valuable time. This experience taught me the importance of thorough research and careful attention to detail when coding.

It is important to note that my proposed solution is not a universal one. While it works well on the specific map provided for this assignment, it may not necessarily be effective on other maps. This is because the waypoints have been specifically set to avoid collisions with walls and other obstacles on the given map, and to ensure that the robot scans around certain objects multiple times to obtain better coverage and results on the occupancy grid. As a result, it is likely that the program would not perform as expected on different maps.