Robotics System CW2 Report

Team Name: **BSHL-1**

Abhishekh Jay Baskaran
StudentID:1758429
25%

Anjie Song
StudentID: 1705056
25%

Xiang He
StudentID: 1723303
25%

Xing Li
StudentID: 1727134
25%

Department of Computer Science

University of Bristol

1 Introduction

As part of the coursework, we have implemented a real robot and equipped it with the functionality for localisation and path planning. Previously, we were provided with a simulated robot that managed to navigate a virtual map while dealing with pre-defined noise. In this situation, a real robot has been built and tested on a map in the real world. The map dimensions were constant and were included in our calculations for the localisation, path planning. We utilised the Particle filters method for the localisation of the real robot and the "Shortest Path Faster Algorithm" was used for the path planning procedure.

We included several features in the code base to tackle the map symmetry problem. Other modifications include the development of an API to deal with the turning, movement and ultrasound scanning functionality. The development process lasted for a long period of time, which includes almost two weeks for testing, analysing and adjusting the robot to the real world map.

2 Related Works

2.1 Particle Filter Localisation

Particles filter is a very robust algorithm that can be utilised for robot localisation as it functions extremely well in different maps. When provided with a predefined layout of the surroundings, the algorithm calculates the orientation and position of the robot. This method works through the use of virtual particles that are randomly generated in the map and uses a Gaussian Distribution to give every particle a unique weight which depends on the difference between the sensor values of the real robot.

We scanned 30 directions in the simulation stage, but due to the time limit in practical testing, the result of this trade-off is that we limit the scanning to only 12 directions. For each particle, we calculated the average value of the square of the error with real robot's scanning values in each corresponding direction, followed by the application of normalisation and sorting methods. During the resampling phase, 90% of the particles are assigned to the position of the particle with the highest probability in the previous round, based on the order of sorting. The rest of the particles are randomly distributed on the map for robustness. We also apply a Gaussian Distribution when assigning one particle to a specific position and angle, and the parameters of this Gaussian Distribution get smaller as the round grows. The prior includes the knowledge that we believe the localisation would be accurate and we hope to be more precise gradually. When we choose a direction to move, we set a parameter to control the probability of the robot choosing the longest direction(scan), this is used to avoid the symmetry problem. This trick in combination with another algorithm used in path planning would successfully solve the symmetry problem. All particles should move in the same pattern as the robot, any particles moving-out are resampled randomly in the map. This process runs continuously until it converges or reaches the threshold of the max running limit.

2.2 Path Planning

Before path planning, we use the geometry method to reduce the size of map for the safety of collision volume 1. The robot would try to find the path in the reduced map which keeps a safe distance from the original map in each point of the edges.

Path planning is required to define the shortest, most optimal path to the target location when placed in a map with complex geometry. When provided with a start and target coordinate the path planning algorithm generates a virtual map that is generally made of nodes and edges. We utilised the "Shortest Path Faster Algorithm" for this purpose which computes the shortest path in a weighted directed graph.

A prior knowledge is that the shortest path must be on the edge or through corner point, so we use the corner points in the reduced map to build an adjacency matrix to apply SPFA. Generally, this algorithm applies a dynamic approximation from target to all other points with relaxation algorithm. Its structure is close to the breadth-first search. A circular FIFO queue is maintained for saving the points waiting for optimised.

One trick needs to be mentioned here is we firstly record the longest path among the minimum distance of all points. And we would move our robot through this path, if our robot falls into the

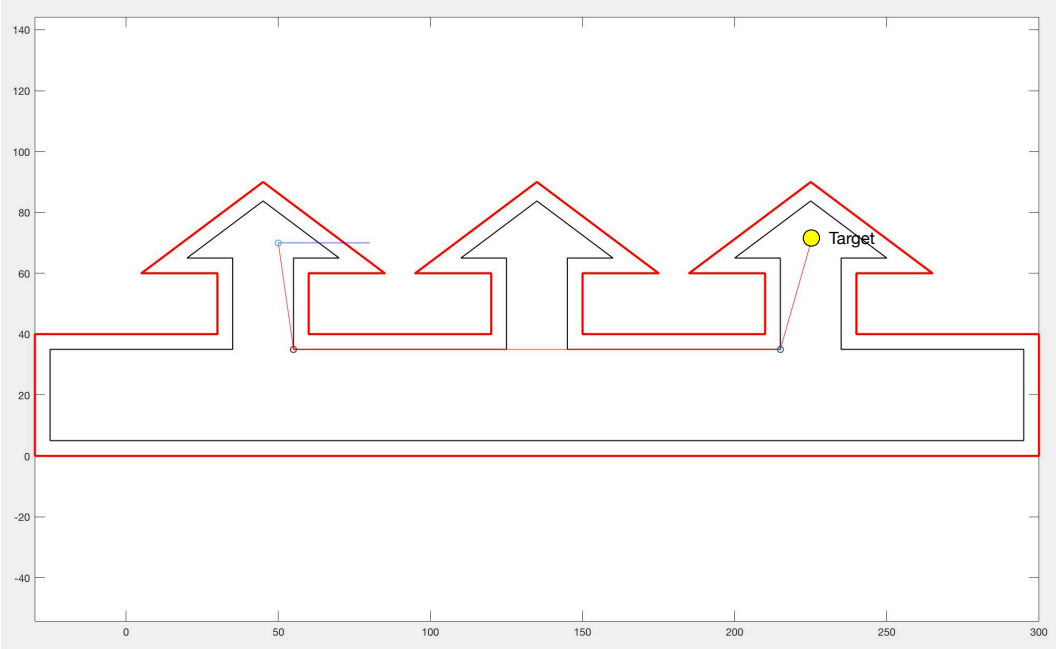

Figure 1: The red boundary is the original map, the black boundary is reduced map. The planning path is also illustrated in map.

symmetry problem, the collision would happen without doubts. In this case, we could re-run out localisation.

After the first trick, now we could apply normal SPFA and the robot should move to the target ideally.

3 Design and Algorithms

3.1 Design of LEGO robot

In the beginning of coursework, scattered components were given which include joints, brick and sensors which means we have to design the robot by ourselves. There are four objectives that we follow to design our robot

1. The robot need to be as small as possible. Since the test area is very small and the motion of the robot is not accuracy, a small robot have high robust to turn itself and low risk to collide wall and turning point.
2. The axis of rotation of the robot need to be close to the centre of it. To make the real robot adjust the interfaces of simulation robots in software layer, we need to design a robot whose moving and turning are similar to BotSim which is a point in the simulation map.
3. The height of the robot should be low to adjust the height of the wall in the map.
4. The axis of rotation of the ultra scan sensor need also to draw near to the centre of the robot which is also aiming to match the BotSim.

Thus, we build our robot following all above, and Figure 2 present the outward of our robot. The robot adopts caterpillar band to move and turn. The benefit of it is that the centre of rotation is the centre of caterpillar band which is also the centre of the robot in our design. The ultra scan sensor and one motor are placed on the top of the brick to rotate and scan the map.

3.2 Collision prediction

Although the robot is able to scan the distance which it faces up to and move forward, it is possible to collide wall. For example, when the robot is near the turning point and the straight scanning

Figure 2: The outward of the robot

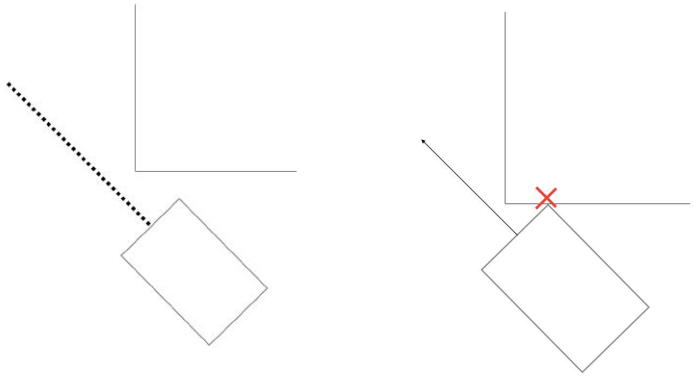

Figure 3: The collision in the corner

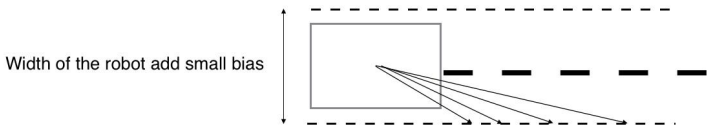

Figure 4: The robot will scan the distances and compare with the distances shown in the arrow

distance tells to the robot that is enough to move forward, the width of the robot will collide the corner. Figure 3 shows this situation. Thus, the robot needs to predict collision before moving.
We solve this problem by a simple mathematical approach. When the robot decides which direction it will move, it scans area from left 45° to right 45° with 5° as step. In each step, the scan value is compared with the minimum distance that the robot will not collide the wall which is given by

$$d = \frac{\frac{width+bias}{2}}{\sin \alpha}$$

where α is the angle between the scan line and the line which the robot faces to. Figure 4 gives this demonstration of the solution.If any of the scan distance is less than d, the robot will collide the wall if it move straight forward. And if every scan distance is greater than respective d, it means there is enough space for the robot to move forward.

3.3 Regression for Error Correcting

Since the moving and turning of the robot are both have significant errors, we want to figure out the projection from the real moving distance to an input value, thus to correct the error.

The projection of moving is nearly linear, so we choose linear regression to approximate the real projection function. And the approximation result is shown in Figure 5. With this function, the error in moving can be eliminated properly.

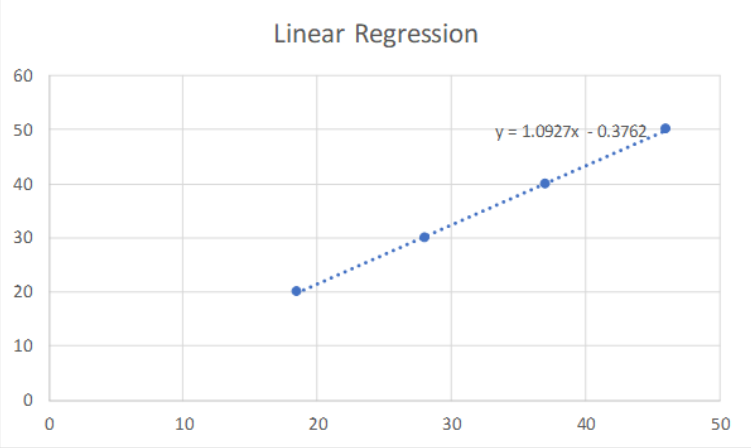

Figure 5: Linear Regression for Moving Distance

And for turning, it is hard to set up a projection since it is a stochastic error. And the parameters for turning also depends on the battery power state. Thus, we set up the parameters to a value which make sure the effect of the error is minimised.

In addition, the scanning of the sensor raised some problems. Every scanning of the sensor does not exactly lead towards the direction we input in the code. Therefore, the sensor cannot return back to the start position with a 360-degree rotation. To address this problem, we read the rotation information from the robot to get the position of the sensor after scanning, and then use this position as the input value for returning rotation. It is worth to mention that the sensor values have 1 to 3 cm errors while we test the sensor from 5 to 100 cm. It is acceptable considering the map size is 110 X 110. However, the sensor value will be influenced by the reflection problem, especially when the robot faces a wall at various oblique angles. Under this circumstance, sensor value of the real robot can be very unstable and should not be trusted. To address it, we want to incorporate this knowledge into the code. But the result is undesirable.

Consequently, we think that having more measurements on a single position can give us more precise sensor values. That is why we connect the sensor to a motor. And the ultra-scan number is set to 12 to make a balance between precision and time limitation.

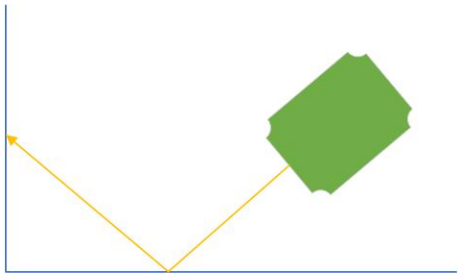

Figure 6: Reflection Problem

4 Conclusion and Future Work

Overall, we have worked well together as a team to design and construct a robot with solid localisation and path planning algorithms. The robot works well in most scenarios, manages to locate the target and navigate to that location successfully. However, since the map has some similar areas,

there is a slight probability that the robot can localise to the wrong position. These are mostly due to the provision of wrong sensor values which are extremely hard to deal with. And this limitation deserves further exploration in the future.

The localisation method being utilised in this project, actually has many limitations when we apply it to the case of real-world applications. For instance, there is the requirement for knowledge of the map characteristics as a prior to the localisation process. An advanced approach is to use 'Simultaneous Localisation And Mapping' (SLAM), which can construct and update a map of an unknown environment while simultaneously keeping tracking of the robot's location within it. When the real environment becomes more complex, we can include some computer vision techniques to assist in the map construction process. 3D reconstruction techniques can be performed on 2D images to generate a 3 dimensional model of the environment. This 3D model can assist in computing the distances from the robot to all objects in the surroundings. However, these techniques are quite complex and unrealistic for this project but could prove extremely useful in a real world application.

More stable and robust sensors can be employed, for instance, a laser sensor can be equipped to detect long-range presence of objects. The laser sensor would provide better results due to the speed of light being much faster than ultrasound. Another benefit is that laser beams travel in straight lines and do not reflect off the walls. This solves the reflection problem while producing a perfect measurement with a very minimal margin for error as compared to the ultrasound sensor.