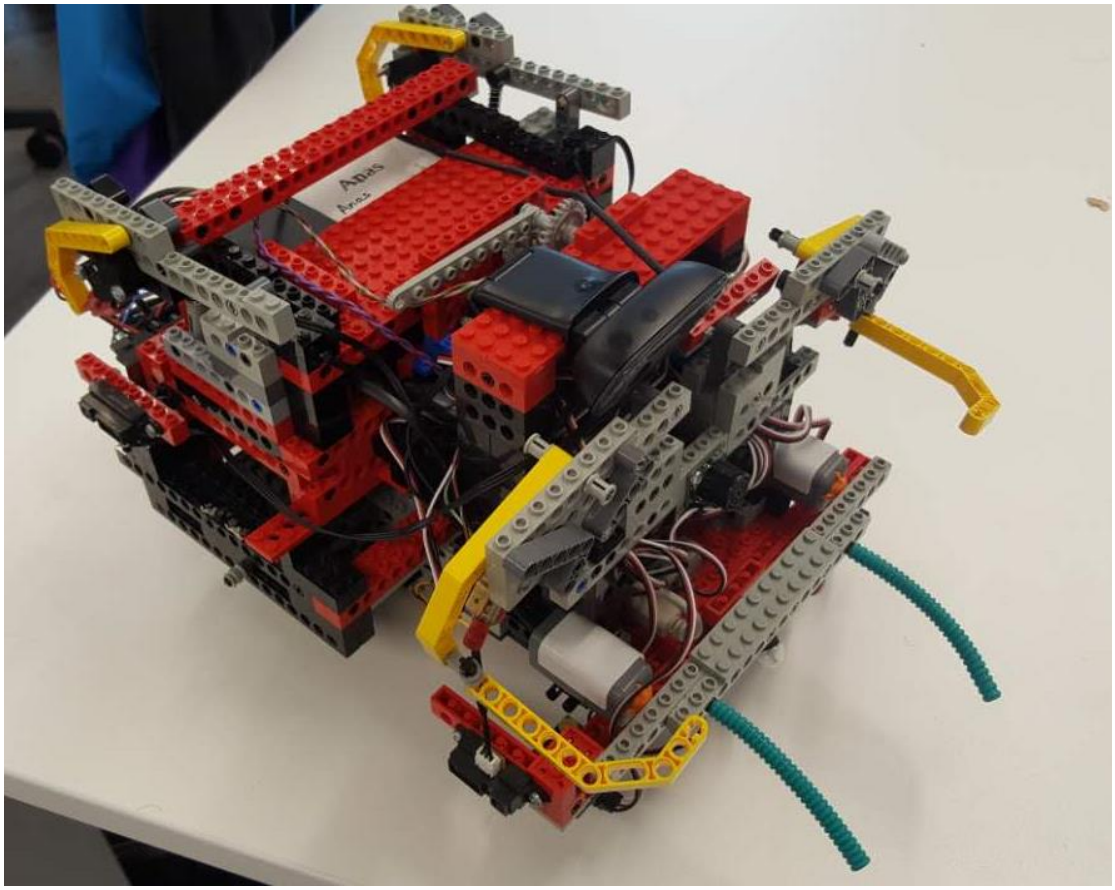
Robotics: Science and System

Practical Report

Grid Localisation and PID Control for Mobile Lego Robot

Jiale Lu
S1778365
November 20th, 2017



1. Abstract

This report is going to discuss how to implement position tracking and control algorithm in a known arena for mobile Lego robot by using one hall effect sensor, two IR sensors and one sonar sensor. The methodologies we will discuss are grid localisation and PID control. Gird localisation is used for tracking position while PID controls are used for planning the route of the robot. The report will describe how to build up this mobile robot by Lego blocks and how to optimize gird localisation algorithm and sensors' data. By using these two methodologies, our group have approached great performance in the course practical task. At the end of this report, the results and performance of the robot will be illustrated and discussed in detail.

2. Introduction

In the course Robotics: Science and System, we were required to build up a robot using Lego blocks and then implement a localisation algorithm for the robot so that the robot can search 3 specific points of interest (POIs) in the given arena autonomously. And then align the antenna to point towards the satellite based on the position of the robot and the given map. The satellite is located at a fixed position (-0.69, 0, 2.95) in the arena and the map and model of arena are illustrated on figure 1. The robot should be able to navigate around the arena without getting stuck all the time.

At the beginning of this task, the robot will be placed inside the starting zone which is marked by black square in the map with a known orientation. During the entire task, the robot is supposed to finish all the subtasks autonomously. The POIs are marked on the floor by reflective tapes as shown on figure 2. In addition, 4 light sensors, 4 micro switch sensors, 2 whisker sensors, 2 IR sensors and 1 sonar sensors were provided for measurements.

To solve this task, my group has tried to implement both particle filter localisation and grid localisation for position tracking. At last, we found that gird localisation is more efficient and robust in this particular application, so we decided to use gird localisation in the final demonstration. The gird localisation was implemented with the assistance of hall effect sensors, IR sensors and sonar sensor. Another important algorithm we used to complete the task is the PID control. Two IR sensors were attached on the right side of the robot for position and orientation PID controls so that the robot would move ahead straightly by following the wall. Although we have used PID control to make the robot move by following the wall, the robot may also probably get collision with the wall in some unexpected situations. Therefore, 4 micro switch sensors were used on 4 corners of the robot to avoid getting stuck by flat walls and 2 whisker sensors were connected with each other and placed on the front of the robot as a "bumper" to avoid getting stuck by non-flat wall like a triangular wall. At last, considering POIs are more reflective than the floor, we used 4 light sensors to detect the existence of POIs. And one 12-V bulb was placed in the middle of the sensors to enlarge the differences between the floor and POIs. After calibrating the threshold for all the light sensors, the performance of the light sensors is quite stable and robust.

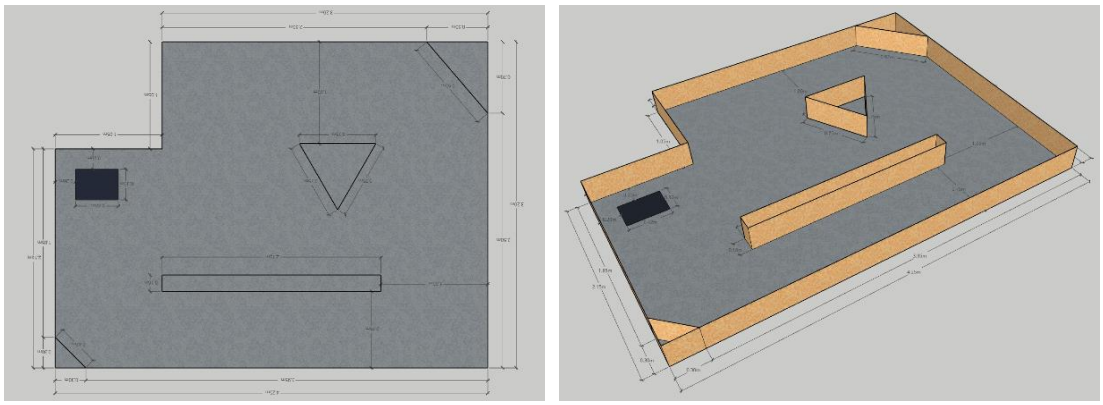


Figure 1. Arena map and mpdel

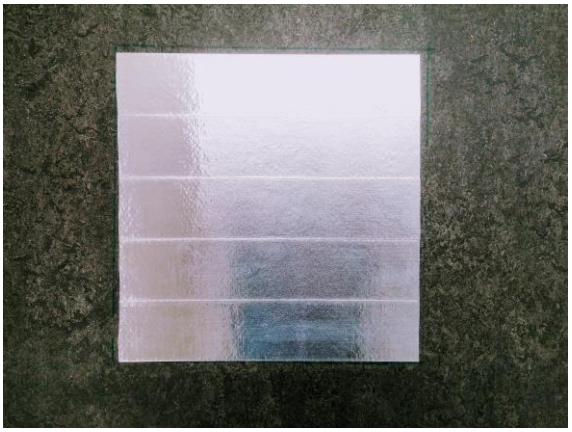


Figure 2. Point of interest

3. Methods

3.1 Mechanic design

3.1.1 Basic structure design

The robot was designed to have 3 wheels which include two driving wheels and one universal wheel. The universal wheel is on the front of the robot while the driving wheels are at the rear of the robot. The hall effect sensor was connected to right driving wheel through gear groups for movement measurement, the structure is demonstrated on figure 3. In order to have high control resolution, the driving wheels were gear down from 9 to 1 while the hall effect sensor was gear up from 1 to 9. And that means the hall effect sensor will rotate 81 circles if the right driving wheel has rotated one circle. This design has enabled the robot to rotate accurately with angle resolution less than 1 degree, but it was also the reason why our robot could not move ahead straightly. But we still kept this design at the end because we believed that the rotation accuracy was extremely important in this task.

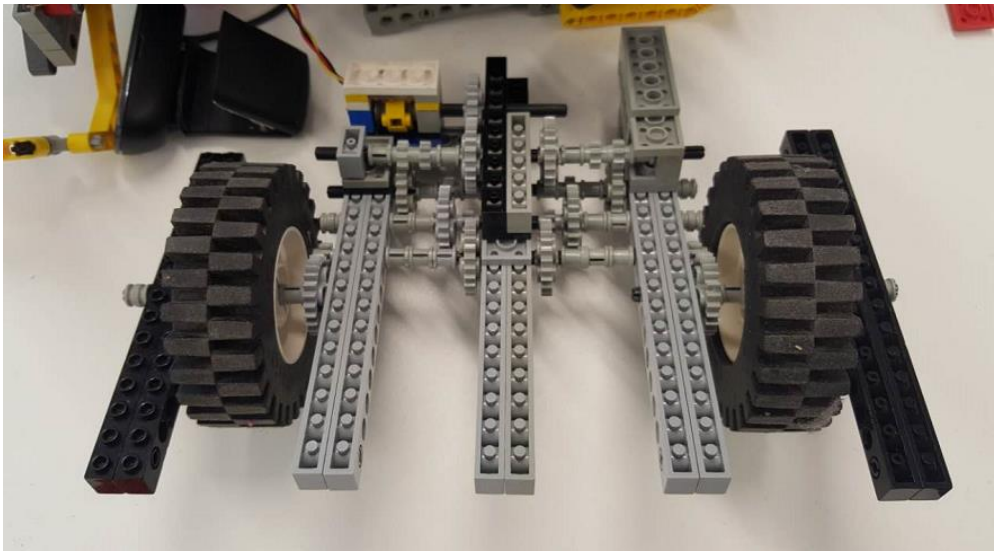


Figure 3. Gear groups for wheels and hall effect sensors

3.1.2 Obstacles avoiding mechanics design

The micro switch sensors were elaborately designed to detect the collisions from multiple directions. In this design (figure 4), a spring was used in the structure as a buffer to avoid hard collisions which may destroy the robot or burn the motors out. The whisker sensors attaching on the front of the robot were connected together by a bar to avoid non-flat wall collisions like the triangle walls which may not be detected by sonar and micro switch sensors.

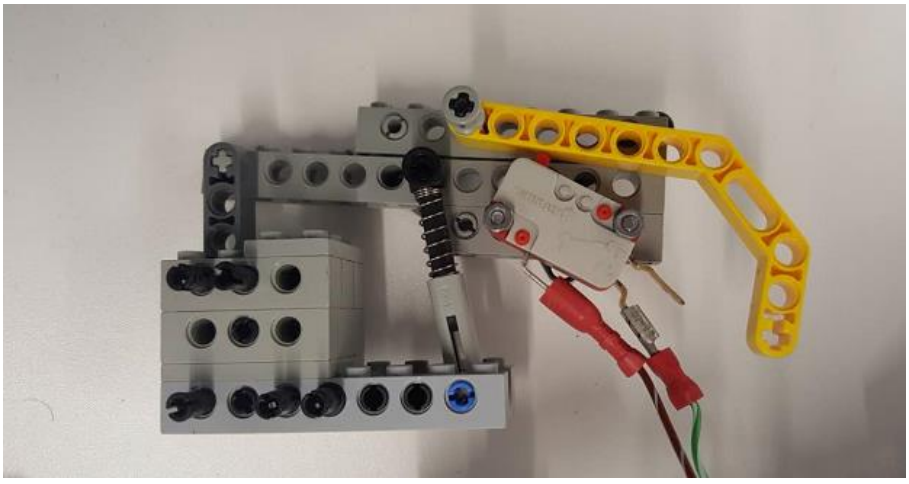


Figure 4. Obstacles avoiding structure using switch

3.1.3 POIs detection methods

Four light sensors were placed underneath front bumper, we have placed a light bulb in the centre of the light sensors to increase the differences between floor and POIs. As the lighting is not exactly the same for all the light sensors, we have also calibrated the thresholds of the light sensors.

3.2 Grid Localisation

As the map was already known, it would be easier to use grid localisation to implement position tracking. And it is also less expensive in computation compared with particle filter. We firstly converted the arena map into a discrete map with 32*42 grids which means one grid represents a $10cm^2$ square. And then we used movement measurement to generate new pobability position map and dictance sensors measurements to recover the errors of postiion. To simplyfier the algorithm, we have assumed the robot will only move in 4 directions which are 0, 90, 180 and 270 degrees.

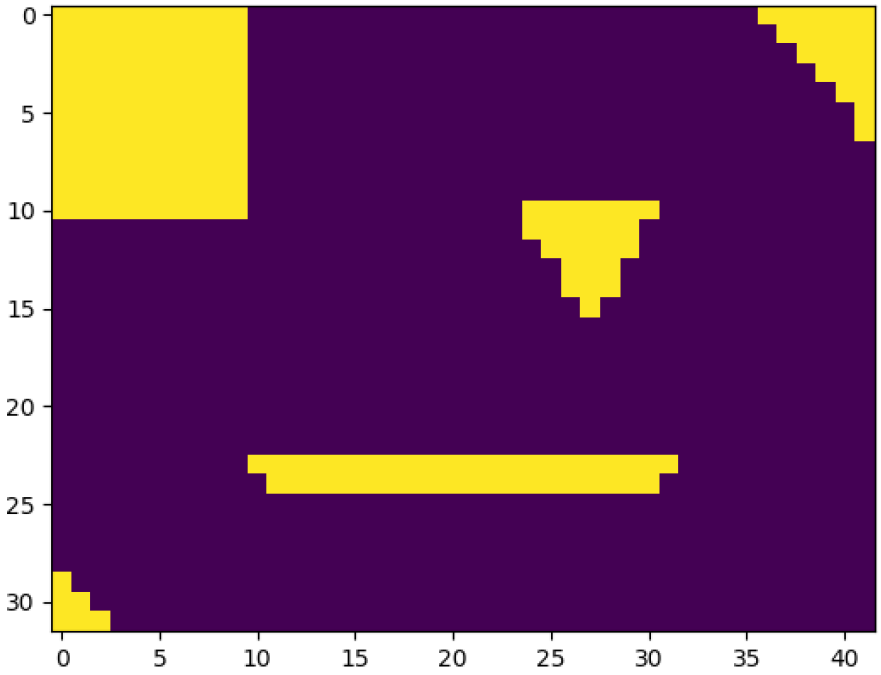


Figure 5. Grid map

3.2.1 Internal measurement

Measure the movement by hall effect sensor and then update the position map. Considering the hall effect sensor may be inaccurate and only have information of one dimension, the probable positions of robot should not be only on one grid otherwise there's no chances to recover it. The most straightforward way to implement it is to convolve the new position probability map with a Gaussian filter. The results were shown on figure 6.

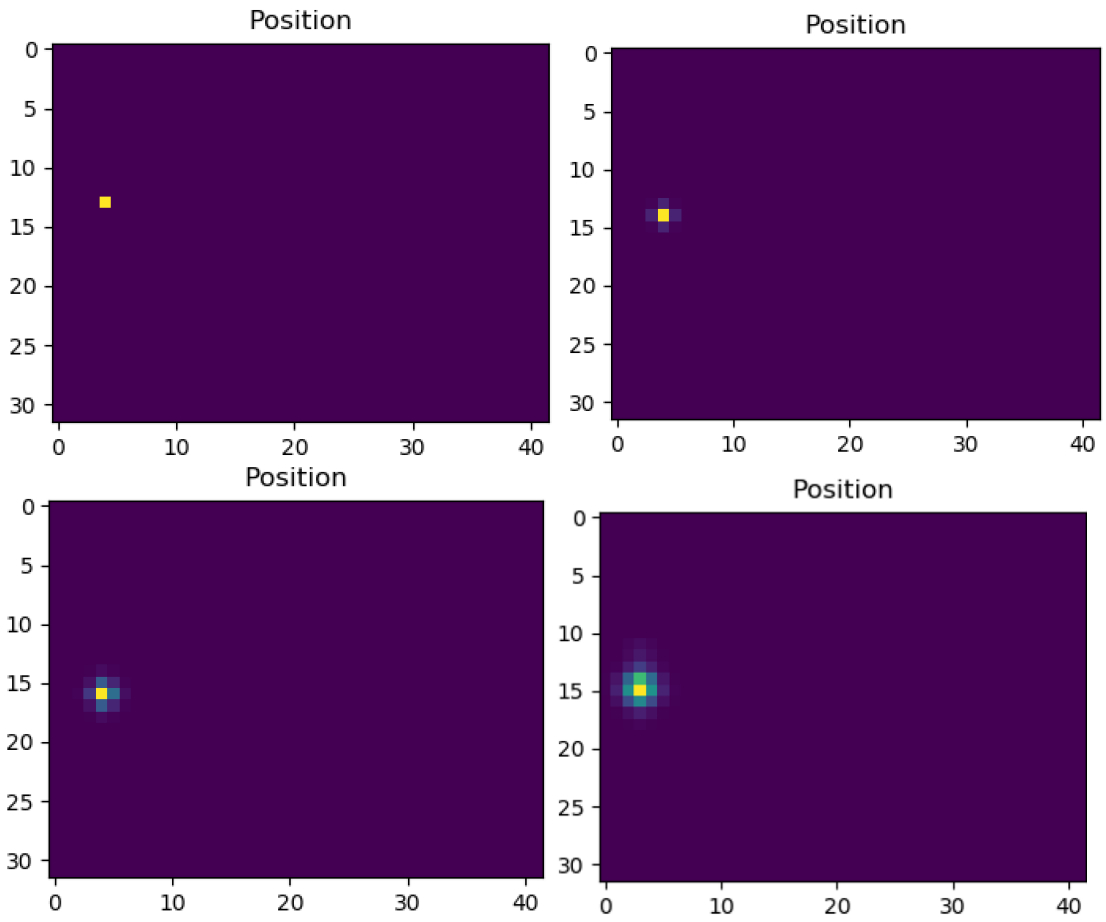


Figure 6. Position map

3.2.2 External measurement.

The external measurement was used to create measurement probability position map in order to fix the errors of the internal measurement. Two IR sensors were attached on the right side of the robot to measure the distances from the wall. We took the average of the IR sensors as the real distance from the centre of the robot to the wall. Also, there was a sonar sensor on the front of the robot to detect the distance from the front wall. However, we found that the sonar sensor may probably detect the side wall if the robot was not parallel with the wall. As a result, we only used sonar measurement on the external measurement probability map when the robot was parallel with the wall (the values of two IR sensors were similar). In addition, we also found that the values of the sonar fluctuated dramatically if the robot was detecting a sideling wall such as the walls on the top-right corner and bottom-left corner. Unfortunately, we were not able to fix this issue, so we decided not to use sonar only when the robot was moving along 0 degrees and 180 degrees. Figure 7 shows two example of measurement probability map.

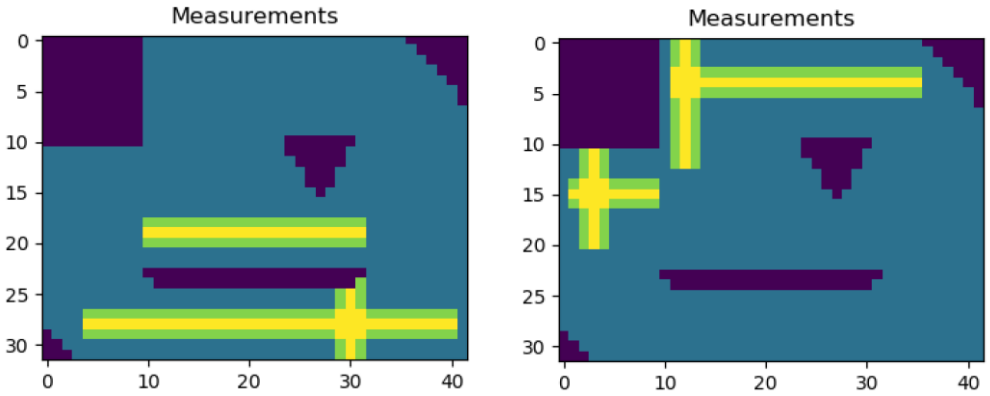


Figure 7. Measurement probability map

3.2.3 Algorithm optimization

As FitPC do not have extended library like Scipy, we had to implement the 2D convolution (used in generating new position map) by for loops. However, for loop in Python is extremely slow such that it took around 3 seconds to update a new position map. To improve the performance, if we refer to the formula of Gaussian equation (1), we will easily notice the separability of the Gaussian filter. Take a n*n image and a m*m Gaussian filter as an example. If we convolved the image by 2D Gaussian filter directly, the complexity of the computation would be $O(n^2m^2)$. But if we convolved the image by a 1D Gaussian filter in two dimensions separately, the complexity would reduce to $O(n^2m)$. At the same time, we can use the library function “numpy.correlate” and avoid using for loop. This optimization has improved the performance of the algorithm greatly and the FItPC was able to generate new position map quickly.

$$
\begin{aligned}
G_\sigma(x,y) &= \frac{1}{2\pi\sigma^2}\exp^{-\frac{x^2+y^2}{2\sigma^2}} \\
&= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)
\end{aligned}
\tag{1}
$$

3.3 PID control

As the robot was going to search on the arena automatically, there should be some rules for the robot to navigate around the entire map. The distance between the wall and robot is a great reference for navigation, so we decided to implement position PID control. Position PID control enabled the robot to move around the arena by following the wall with different distances. And after we have changed the target values of PID control after every circle, the route of robot was supposed to cover every corner of the map automatically. We used only one IR sensor on the right side to implement this algorithm at the very beginning, but we found that the robot was not able to return to the designed route quickly and it would move forward in a

random curve which is totally out of control. Under my considerations, the reasons for this problem are:

- The values of IR sensor will change if the robot is not parallel to the wall and the true distance the sensor measured in this case is shown on figure 8.
- Although the robot has return to the designed route, it will leave again as it is moving at a wrong direction.

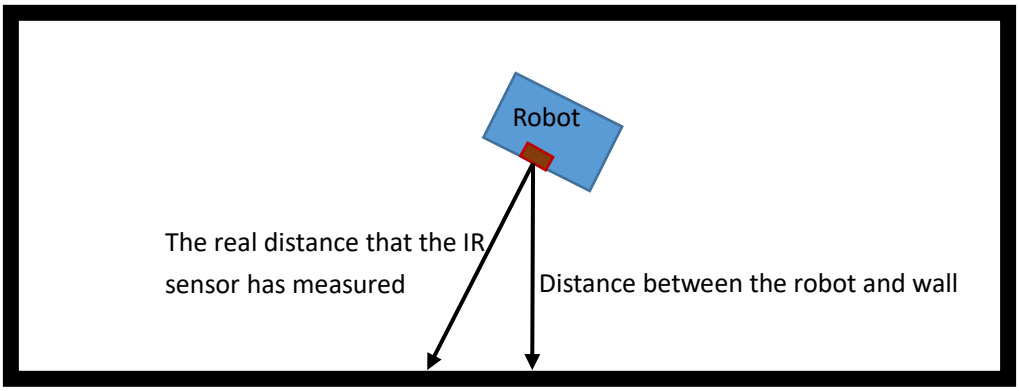


Figure 8. IR sensor measurement

Therefore, we included orientation PID control in the algorithm later to overcome these issues. The basic idea is that after the robot has almost returned to the designed route, an extra orientation control will be activated to force the robot rotate to the direction which is parallel to the wall. In this case, the robot will not leave the target route again once it has got back to the target route. The orientation PID control was implemented by taking the difference of 2 IR sensors as inputs because one IR sensor was on the front of the robot while the other one was at the rear of the robot. So, when the robot was parallel with the wall, the values of two IR sensors should be the same.

Another problem we found during the experiments was that if the robot was far away from the target route, the robot may end up rotating at the same position because the robot has rotated a large degree which may larger than 90 and could not detect the wall anymore. Thus, it was also necessary to limit moving orientation of the robot.

In conclusion, the algorithm for PID controls should be:

- Detect the distances form the wall by IR sensors, control the robot to move back to target route;
- Check if the robot has almost returned to the target position, if yes, go to step 4, if no, go to step 3;
- Check if the orientation (difference between two IR sensors) is larger than a threshold, if yes, stop PID position control and keep moving forward, and go to step 2, if no, go to step 1;
- Detect the distances form the wall by IR sensors, control the robot to be parallel to the wall. Go to step 1.

3.4 Calibration of sensors

In order to use the sensors' values to create a real-world coordinate measurement

map, we have calibrated all the sensors with the real world coordinate so that they return the real distances directly.

3.4.1 Calibration for hall effect

The method for calibrating hall effect sensor was straightforward. First, we calculate the ideal hall signal unit distance based on our design, which will be:

$$\text{Unit of hall} = 80*\pi / 81 / 2 \approx 0.16677$$

And then the robot was told to move forward for 1 meter. We measured the actually distance the robot has moved and then updated the coefficient.

3.4.2 Calibration for IR sensors

The calibration for IR sensors was relatively complicated and time-consuming. Firstly, we recorded the values of the IR sensors at different distances from the wall for both IR sensors. And then we fitted the points into a polynomial equation and got 2 different equations for IR sensors which are illustrated on figure 9 and 10.

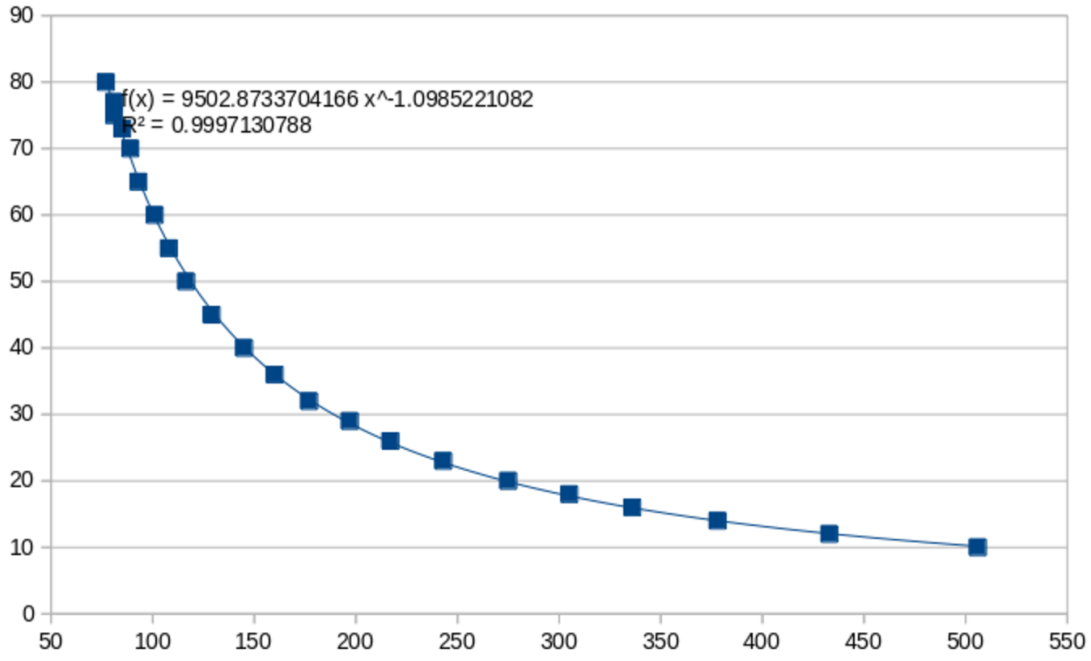


Figure 9. Fitted curve for front IR sensor

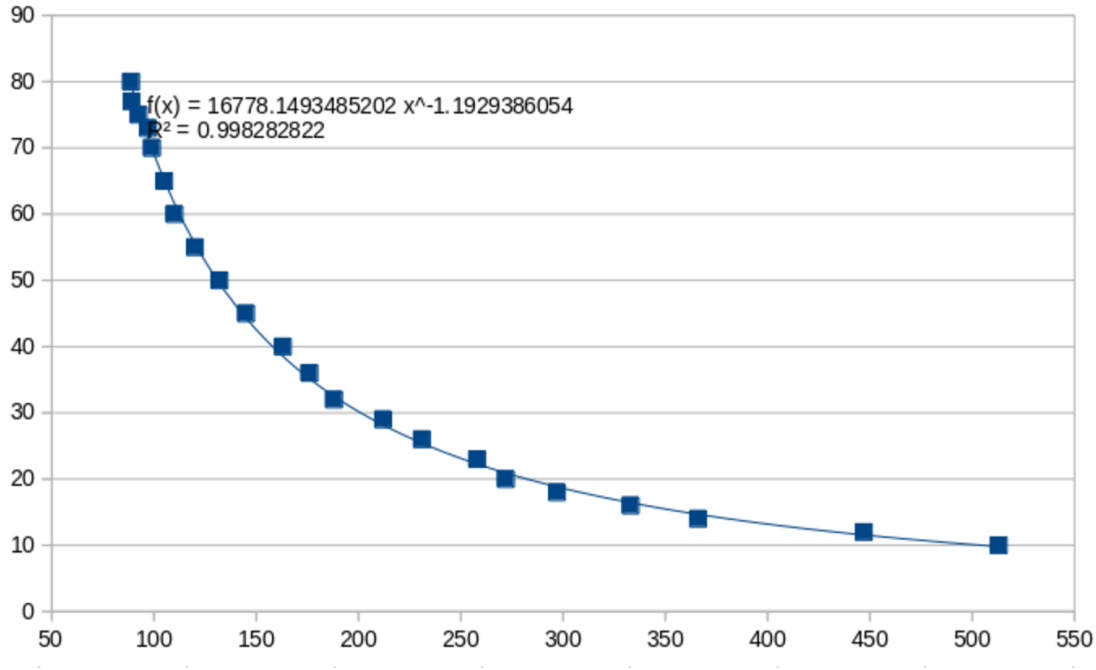


Figure 10. Fitted curve for rear IR sensor

3.4.3 Calibration for sonar sensors

The steps are exactly the same with previous one. In our case, we found that the sonar sensor's values are just three quarter of the real-world distance.

3.5 Sensor data preprocess

In fact, the distance sensors' values were not stable sometimes, especially when they were measuring far distances. So, we needed to preprocess the data before we used it for our algorithm. It is generally fair to consider most of the real-word noises as Gaussian noise. Therefore, the most straightforward idea to remove the noise is to apply a box filter to the raw data. However, in order to improve the stability, we had removed maximum values and minimum values before we applied the box filter. The overall algorithm will be:

- Collect 200 values from the sensors;
- Remove 60 maximum values and 60 minimum values;
- Take the average of the remaining values (same as using the box filter).

4. Results

The methodology we have used was robust as the position errors can be recovered by the external measurement probability map.

The simulation result which has taken noises in to consideration is shown on figure 11. The simulated robot was programed to move around the arena with some internal measurement errors but with accurate sensor measurements. The incontinuities of the route map have clearly shown that the position of the robot have been recovered by combining the new position probability map and the measurement probability map.

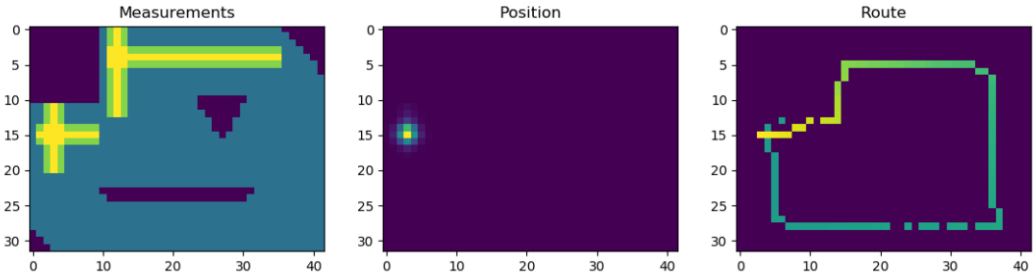


Figure 11. Simulation result

As for the real-world performance, after multiple optimizations, the robot was able to follow the wall stably and move forward straightly. It could detect all the three POIs in the first circle and then align the antenna to point towards the satellite accurately. During the final demonstration, our robot completed the whole task using around two and half minutes and obtained full points.

The strength of our robot

- First, the position of the robot will be really accurate while the robot is moving along 90 degrees and 270 degrees as the sonar sensor values can fixed the errors of the hall effect.
- Second, the robot can rotate accurately after we have gear up the hall effect sensors.
- Third, all of the values of our sensors are just real-world data, so we can use them directly to estimate the position of our robot.
- In addition, after we have preprocessed the sensor values, the final values we get are really stable compared with raw data, especially when the sensors are detecting far distances. The IR sensors now even can work at the distance of 1 meter which is out of the working range of the IR sensors.
- What's more, our robot is sensitive to any collision as the micro switches used on the corners can be trigged by tiny force and are sensitive at multiple directions.
- At last, the PID controls work quite stable and enable the robot to move straightly.

The shortcoming of our robot

- The position of the robot will have some small errors when it is moving along 0 degrees and 180 degrees. But the movement on these directions are relatively short and our hall effect sensor has high resolution so it is not a terrible issue.
- What's more, for some unknown reasons, the hall effect sensor can only work in one direction, so our robot can only rotate in one direction all the time. But it is not a horrible problem either as most of the time we just need to rotate in one direction. And we can rotate to any expected orientation as the robot has precise rotation accuracy. But it is true that it will take more time for our robot on rotation.

5. Discussion

The overall performance of our robot is great. The robot can follow the wall and move ahead straightly and stably most of the time. And it can return to the target route quickly. Also, the grid localisation works well and the robot can align the antenna to point towards the satellite accurately most of the time. In my view, the most successful elements of our approach are the calibrations of sensor values, the PID controls and the optimization of the convolution algorithm for gird localisation. The robot is less successful in measuring front distance which is limited by the characteristics of the sonar sensor and the robot cannot recover the position if the robot lost its moving orientation at some special case. Another issue can be improved is that the robot can only rotate in one direction. To improve the position tracking performance, a more robust grid localisation could be implemented such that the position errors can be recovered all the time. For example, it is possible to use computer vision and sonar sensor at the same time to predict the front distance.