UNIVERSITY OF BRISTOL

ROBOTIC SYSTEMS

Authors:
Signe JENSEN
William MAPSTONE
Oliver PIKE
Edward MCLAUGHLIN

Supervisors:
Prof. J.GOUGH
Dr. J.YON
Dr. K.ALEMZADEH
Dr. M. PEEL

Unit Coordinator:
Dr. W. MAYOL-CUEVAS

The Kidnapped Robot Problem

A simulated and demonstrated solution

IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF ENGINEERING IN THE SUBJECT OF
MECHANICAL ENGINEERING

November 24, 2015

Table 1: Table to show the distribution of work

	Signe Jensen	William Mapstone	Oliver Pike	Edward McLaughlin
Localisation	40	15	15	30
Path Finding	40	20	20	20
Robot Calibration	15	40	30	15
Robot Actuation	15	35	25	25
Report	15	15	35	35
Total %	25	25	25	25

A brief sentence to describe individual roles

Signe Jensen: Coding the simulation (including own a-star code)

William Mapstone: Build, calibration and coding of robot

Edward McLaughlin: Coding localisation and helped robot coding

Oliver Pike: Coding path finding and robot calibration.

Contents

List of Figures

List of Tables

Abstract

This report covers our approach in finding a solution to the kidnapped robot problem. A MATLAB code was first used in order to create a simulated robot. This simulation was then tested using several different maps. Subsequently, a physical robot integrated with NXT software was built and calibrated for testing within a real physical map. The different sensors used were analysed and modeled in a way that took into account their inherent inaccuracies. The simulation proved capable of solving the problem for all of the maps with the different levels of noise. The average time for completion was 17s and the average distance from the target was 1.3cm. The physical robot was also successful in localising and finding its way to a target point, however it was not as fast and much less reliable than the simulation. This was due to the inaccuracies in the motors and sensors, as well as inherent difficulties of operating in a real environment.

1 Introduction

The fundamentals of practical robotics were set out by Norbert Wiener in 1948. He is credited with being the founder behind cybernetics. The term cybernetics is used rather loosely in present day technology, but is applicable when using a closed signaling loop. It wasn't until the later half of the 20th Century that fully autonomous robots were being used. In 1961, the Unimate was installed as the first digitally operated and programmable robot. Since then, robots have been implemented across all realms of technology with a vast range of capabilities.

The kidnapped robot problem involves an autonomous robot that is placed at a random location in a room, with the task of localising itself and navigating its way to a desired point without colliding with obstacles. It is most commonly used to re-localise a robot if it has had a catastrophe during its operation and does not know its location. The aim of the project was to solve this problem in a simulation using MATLAB, and then to adapt the simulation so that it was able to solve the problem in a real world situation. This was achieved using a probabilistic robotics method called Monte Carlo Localisation (MCL). This uses an algorithm that involves using a particle filter to estimate all the likely states that the robot could be in. The projects success is judged on the accuracy to which the robot can get to the target and the time it takes to get there.

2 Simulation

The simulation was programmed in MATLAB using a framework of functions and the predefined BotSim library. A visual example of the simulation is shown in the appendix, Fig 7.3.

2.1 Localisation

Particles were initially spread across the map using a uniform probability of landing at any position and in any orientation. The robot and all of the particles performed a number of scans

to calculate their proximity to a wall. These euclidean distances were then compared to one another and a percentage error calculation was performed to yield a weighting. The particle readings that best matched those of the robot were given a higher waiting and those that had a poor match were given a low weighting.

The program was run using weightings based on a Gaussian distribution. The particle weightings were normalised so that those with a very low weight were not disregard prematurely. Any particles that maintained a critically low weighting were removed and redistributed based on the weightings of the other remaining particles. The robot and all of the particles were then translated and rotated in an identical manner. At this point the convergence of the system was checked. If the standard deviation between the readings from the particle and the readings from the robot was less than a critical value then the system was deemed to have localised and the mean X and Y coordinates of the particles were taken to be an acceptable estimation of the absolute position of the robot. If the localisation criterion of the system was not met then all of the particles and the robot performed another scan and the loop was repeated.

A flow diagram of the locational process is shown in the appendix, Fig. 7.4.

2.2 Path Finding

As the boundaries of the map were known, once the robot was localised it could be translated towards a target point. The main objective of this process was to execute the journey in the most efficient manner possible. This involved taking the shortest path whilst avoiding any contact with the boundary of the map i.e. the wall.

The path finding method that was used in this project was related to an A* search algorithm that used multiple nodes spread across the map to find the most efficient feasible route to the target. Nodes that were adjacent to walls were recognised and therefore any path through them was avoided to prevent collisions.

3 Building, sensing and actuation of the robot

3.1 Building the robot

The walls of the arena that the robot was tested in were 20cm tall and thus the height of the robot and any added sensors were limited to being within this height. The way in which the robot moved around the map was of pivotal importance to the success of the solution. Ideally, the robot moved with as little inaccuracy as possible. These inaccuracies in movement presented themselves whenever the robot translated or rotated. In order to minimise them, the robot was built to be as robust and as compact as possible. To reduce complications, a front wheel drive axle and a rear caster wheel was used so that the point of rotation was in the middle of the front wheels. This point was also used as the point of reference for any translation of the robot. The selection of equipment in the box limited the design of the caster wheel but ultimately a slider design was used. This allowed the rear pivot point of the caster wheel to slide freely across the map surface.

Figure 3.1: The built robot

3.2 Sensor Modeling

Two different types of sensor were available; a touch sensor and an ultrasound scanner. As with all sensors, there are inherent inaccuracies that need to be understood and accounted for. Consequently a series of tests were carried out as a means of calibrating the sensors. In order to test how well the wheel tacholimit corresponded to the physical distance traveled, the robot was moved in a straight line multiple times over different distances and the measured values were recorded. The difference between the intended and absolute values was averaged and a correction factor was calculated as a means of calibrating the code with the real world. The results of this calibration can be seen in the table in Figure 7.1.

The use of the touch sensor was deemed as unnecessary due to the decision to never allow the robot to come into contact with the walls. Instead, the range of positions that the robot was allowed to explore within the map was reduced in order to cater for the length of the robot behind the point of rotation. For this reason the accuracy of the touch sensor was not tested.

The ultrasound scanner was attached to a motor to enable it to rotate. The subsequent rotation of the ultrasound scanner produced errors. This was analysed so that the rotation could be calibrated and any errors were then accounted for in the model. To do the analysis, a straight bar was attached to the front of the robot at its centre of rotation in order to create a reference for its orientation. A protractor was printed out and stuck to a table to provide a frame of reference. The robot was then repeatedly rotated through different angles and the absolute values were

compared with the input commands. A standard deviation was calculated from the result. This was then used to compensate for the under or over rotation of the motor in order to obtain more accurate real rotations. The results of this calibration can be seen in Figure 7.2.

3.3 Localisation

The robot learns of its position within the map by calculating the distances towards its surroundings much like the method in the simulation. To achieve this, an ultrasound scanner is used to make four scans in 90^{o} increments. Only four scans are taken due to the fact that the real robot takes too long to perform any more than this. The built robot localises itself in the same way that the simulation does by using a particle filter and matching it's ultrascan readings with those of the particles.

The robot can rotate using its wheels or by using a separate motor to rotate the sensor itself. It was decided that the optimal way of moving the sensor was attaching it to an axle linked with a separate motor. Therefore the robot would not have to re-orientate itself every time it wanted to take a scan. If rotation about the wheels were chosen, the uncertainty in the wheels would have meant the robot would not have returned to its original position or orientation, thus affecting the reliability of the readings. Instead the inaccuracy is restricted to the motor directly attached to the ultrasound scanner.

3.4 Path Finding

The navigation of the robot to the intended target point, once it had localised, worked in much the same way as the simulation. An A* algorithm was used and the most efficient path was found and followed. If the spread of particles during path finding became larger than a chosen standard deviation, the robot stopped, performed another scan and relocalised. Once this had occurred it continued towards the target point. The minimum distance that the robot could reach between itself and the wall was also increased to make sure that it did not collide with the wall as it rotated.

4 Troubleshooting

4.1 Collision Avoidance Pre-localisation

Avoiding collisions with the wall before localisation was crucial. If the motors were still turning and the robot was resisted by a wall, then the translation of the robot was not duly accounted for and its absolute position differed from where the particles believe it had moved. Within the simulation, if the robot measured a wall that was less than 20cm in front of it, it would take a full scan of its surroundings and travel in the direction of the largest distance. The code was then adapted for the real robot so that if it was very close to two walls, it would turn away from them rather than turning into the corner. Having measured the fact that it was close to a wall in

the forwards direction, the robot would then measure the distance to a wall at 90^{o} and 270^{o} to itself. By using an -1^{n} parameter in the code, the power value given to the two different wheels would be made either positive or negative and as a result the robot would turn away from the closest wall. This prevents the robot from getting caught in a corner.

4.2 Collision Avoidance Post-localisation

The robot is deliberately designed with two front wheels and a rear caster wheel so that it rotates about the front wheel axle. This makes it easier to control spatially and allows a specific reference point for translation and rotation. Bumping into the wall of the map is undesired when the robot moves. Therefore all movements are carried out with the consideration of a safety distance between the edge of the possible areas the robot could end up in and the wall. In the code this was achieved using a shrunk map whereby the coordinates of the map were reduced.

4.3 Dealing with cables

One problem that arose with the ultrasound scanning was the wrapping around of the cable that joins the NXT box to the sensor. If the rotation occurred in a continuous direction, then the cable would get tighter and eventually the cable would strangle the motor. The solution to this was instructing the robot to turn back 360 degrees every time it finished a scan. The error in this return was analysed and factored into the tacholimit used.

4.4 Caster Wheel Catching

The division of the tiles on the map floor create slight lips that are sometimes covered with masking tape and sometimes not. This can cause the rear caster wheel to catch as the robot rotates and therefore disturbs the localisation and navigation processes. The justification of this source of error in the map was given as an example of a 'real life problem' that the robot should be able to account for. To solve the issue, the rear caster wheel was given a tape covering to smooth out the edges so that it would pass over any gaps or lips without catching. The majority of the weight of the robot was placed as close to the front wheel axle as possible to reduce the traction between the caster wheel and the ground surface. To further assure that the robot is in the correct position, as soon as the variance of the particle positions is over a value of 4, the robot stops, takes another scan and relocalises itself.

4.5 Flipping the index after localisation

Certain areas of the map are relatively symmetrical. In order to ensure that the particles are in the correct orientation, for the first five iterations, half of the particles orientation were flipped before the next iteration of the code. When the robot then moved forward and took another reading, the correct particles would confirm their position and those that were wrong would be reassigned to the correct area of the map. Despite taking slightly longer, this check was deemed as a necessary alteration.

5 Results

Table 2 shows the results of the simulation. It indicates how long the robot took to complete the different maps and states the distance that the robot finished away from the intended target point. It also shows if the robot collided with any of the walls.

Table 2: Table to show the results of the simulation

Map	Noise	Time (s)	Distance (cm)	Collisions
1	1	4.654	0.284	0
1	2	4.050	1.827	0
1	3	3.671	0.964	0
1	4	4.767	1.016	0
2	1	11.055	0.487	0
2	2	9.336	1.222	0
2	3	9.494	2.270	0
2	4	10.494	0.461	0
3	1	17.798	0.746	0
3	2	15.832	0.863	0
3	3	14.552	1.484	0
3	4	12.130	2.190	0
4	1	34.646	1.829	0
4	2	52.393	1.673	0
4	3	32.636	1.141	0
4	4	33.787	1.784	0
Average %		16.956	1.265	0

6 Discussion

The code that was adapted for the simulation ran very quickly for all of the maps at all noise levels without colliding with the walls. This was achieved through thorough checking of the code and careful debugging to account for inaccuracies. Once the simulation was running correctly for all maps and noise levels, a successful conversion was made to apply the code to the real robot. The commands used to actuate the sensors and motors in the robot were substituted into the algorithm to solve the problem in a physical environment.

There were still errors within the real robot despite accounting for them as much as possible using calibrated standard deviations from the hardware performance. These can only be reduced from further tuning of the code to a level where the process of checking and confirming the position of the robot takes too long.

6.1 Future Improvement

The implementation of a differential drive would improve the path finding significantly. The robot in this project moves in straight lines and rotates in multiples of 90^{o} in order to make its way to the target. Applying differential drive would allow a curved path to be made and so the route taken to get to the target would be reduced.

There is a bluetooth function with the NXT software that enables the testing of the real robot without the main cable that attaches it to a computer. Having a wireless robot would prevent any dragging of the cable against the floor or walls of the map and would also make sure the ultrasound doesn't take a reading off of it. The kit received at the beginning of the task also contained various balls. Implementing these as a caster wheel contact with the ground would improve the movement of the robot as it rotated by reducing the friction with the surface and by allowing easy sliding over any undulation.

7 Appendix

Forward Movement			
Target Distance (cm)	**Distance Moved (cm)**	**Error (cm)**	**% Error**
6.6	6.5	0.10	1.52
6.6	6.6	0.00	0.00
6.6	6.4	0.20	3.03
6.6	6.7	0.10	1.52
6.6	6.6	0.00	0.00
6.6	6.6	0.00	0.00
6.6	6.6	0.00	0.00
6.6	6.7	0.10	1.52
6.6	6.6	0.00	0.00
6.6	6.5	0.10	1.52
6.6	6.6	0.00	0.00
13.2	13.3	0.11	0.80
13.2	13.1	0.09	0.72
13.2	13.2	0.01	0.04
13.2	13.1	0.09	0.72
13.2	13.2	0.01	0.04
13.2	13.2	0.01	0.04
13.2	13.3	0.11	0.80
13.2	13.2	0.01	0.04
13.2	13.4	0.21	1.56
13.2	13.2	0.01	0.04
26.4	26.6	0.21	0.80
26.4	26.4	0.01	0.04
26.4	26.6	0.21	0.80
26.4	26.5	0.11	0.42
26.4	26.7	0.31	1.18
26.4	26.5	0.11	0.42
26.4	26.4	0.01	0.04
26.4	26.6	0.21	0.80
26.4	26.6	0.21	0.80
26.4	26.6	0.21	0.80
Average:		**0.09**	**0.64**

Figure 7.1: Forward Movement Calibration

Turning Movement			
Target Angle (°)	**Angle Moved (°)**	**Error (°)**	**% Error**
45	50	5	11.1
45	47	2	4.4
45	48	3	6.7
45	49	4	8.9
45	50	5	11.1
45	47	2	4.4
90	90	0	0.0
90	85	5	5.6
90	89	1	1.1
90	90	0	0.0
90	90	0	0.0
90	88	2	2.2
90	87	3	3.3
90	89	1	1.1
90	85	5	5.6
90	82	8	8.9
180	175	5	2.8
180	175	5	2.8
180	179	1	0.6
180	178	2	1.1
180	175	5	2.8
180	170	10	5.6
180	170	10	5.6
180	178	2	1.1
180	180	0	0.0
Average % Error for 45° target			**2.8**
Average % Error for 90° target			**2.5**
Average % Error for 180° target			**7.8**

Figure 7.2: Rotational Movement Calibration

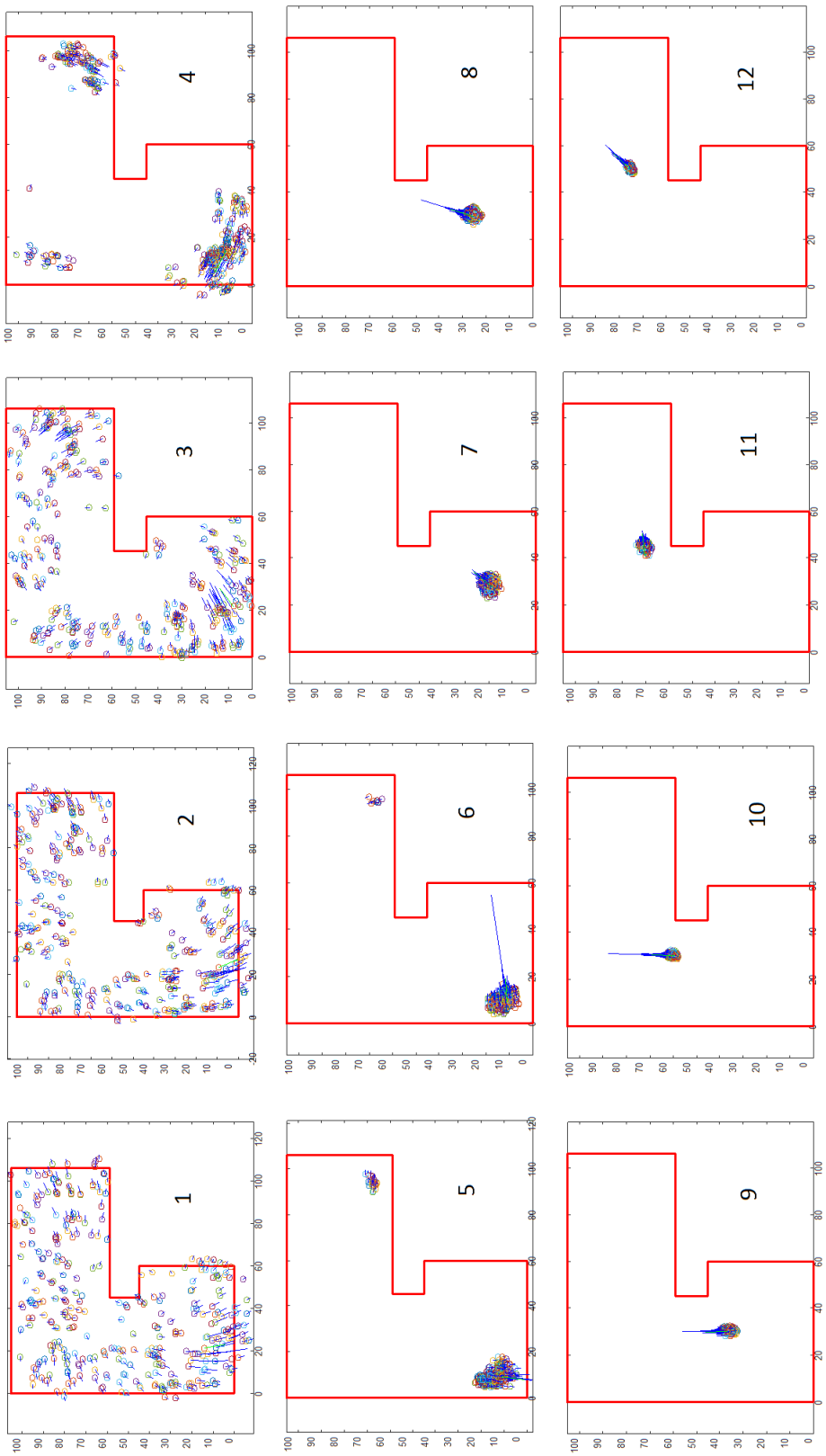

Figure 7.3: Visualisation of the simulation: localisation (1-8) and getting to target (9-12)

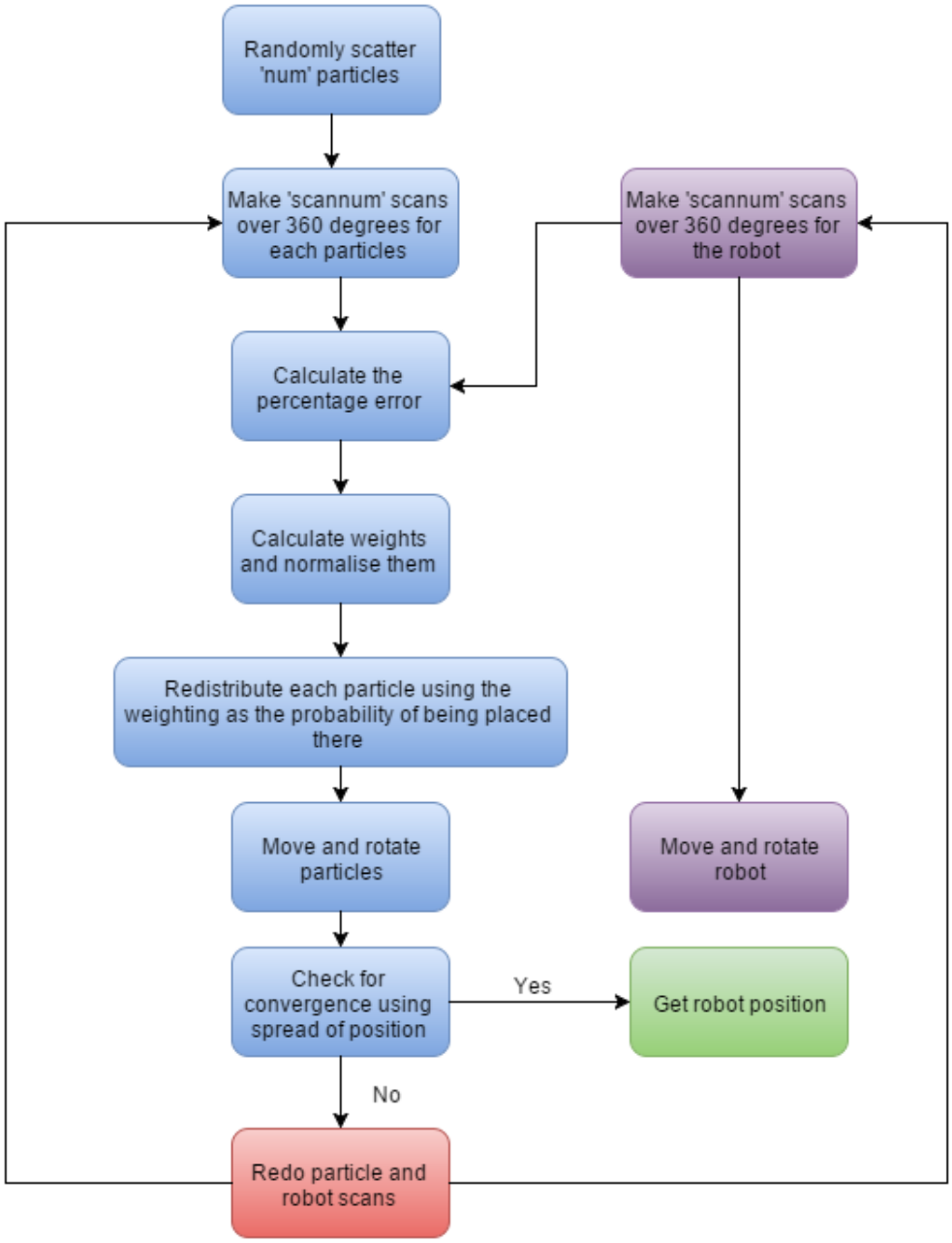

Figure 7.4: Localise Flow Diagram