The Kidnapped Robot Problem

Team Name: FPMM-2

Members & Assigned Tasks: Christos Mourouzi 25% - ultrasound, Diar Masri 25% - odometry, Nate Fear 25% - ultrasound, Graham Peden 25% -odometry.

All: Robot design, report, refinement.

Introduction

The kidnapped robot problem refers to a scenario in which a robot is translated and/or rotated within its environment by an external factor such as a person. This action causes the robot to lose its place within its map of the room thus preventing it from reaching its target. To solve this problem the robot must have measures in place which allow it to sense its environment and reason about its position within the room. This report details our approach to solving this problem.

Division of Labour

As a group, we paired off to model and produce sensor and drive functionality because it was difficult to all work with the robot simultaneously. Upon completion, we discussed the strengths and weakness of each of our code from the first assignment to democratically decide on whose code to use. We decided on using Diar's code since we had reasonable confidence in both the particle filter and path planning features. Once this decision was made it somewhat set the dynamic for the group since Diar became mostly responsible for implementing code changes as discussed by the group. For the vast majority of the time all group members worked together making improvements empirically, documenting progress and discussing changes to be made. The existing code already produced a significant number of graphs and these aided the group in debugging performance and understanding the model's behaviour.

Robot Design Process

When designing the robot we took inspiration from the robot gallery lecture material, this helped us to decide that a good approach would be to construct a two wheel drive robot with a sensor that could rotate and scan 360 degrees. In our first attempt the body of the robot was large, flimsy and the sensor was too high and far from the drive wheels. From this we observed that a rigid, compact robot with a sensor close to the drive wheels would give us less error. We decided to mount the ultrasound sensor just above the middle of the wall height to avoid loss of signal and tried using a castor wheel and taping the sensor cable to reduce snag and friction (Figure 1). We found the castor wheel caught on the gaps in the flooring so we switched to low-friction skis. The power block was also re-orientated from a horizontal position across the wheel axis to vertical one in order to reduce the robot's width and make corners easier to turn (Figure 2).

Figure 1 - Castor Wheel

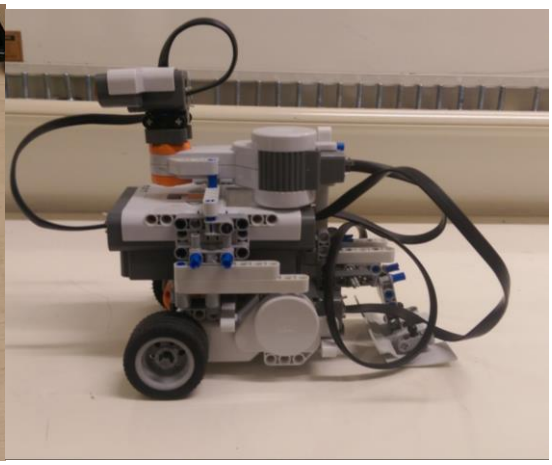


Figure 2 - Skis

Robot Final Build

The final build of our robot (Figure 2) consists of a front wheel drive where each tyred wheel is powered by an individual motor, these motors pull along skis for support. The power brick is mounted vertically on top of the motors and drive axis, with the skis protruding out behind the robot and power brick. A third motor is then mounted on top of the power brick with the point of rotation directly on top of the drive axis. The third motor is used to rotate an elevated ultrasound sensor. In the final build the sensor no longer required tape to reduce snagging and friction because moving the motor mount meant that more cable was freed up, allowing less resistance and removing the possibility of snagging.

Ultrasound Modelling

Inspection of the sensors properties was carried out by placing the robot in the arena and observing the outputs of the sensory data in various scenarios. We placed the robot perpendicular to a wall and compared distance estimates with real distances in range of 10 -70cm. The sensor provided distance estimates with average error of 2 cm which we considered precise enough. Nevertheless, we still discard readings with values greater than 120 which are not possible to take in the arena.

The biggest disadvantage of the ultrasound sensor occurs when measurements are taken with the sensor facing the wall at various oblique angles. In the scenarios where the measurements were not perpendicular to the wall the robot provided a larger estimate than the real distance for angles greater than 45 degrees. We tried to incorporate this knowledge into our code; however, it resulted in overall loss of information when we adapted the simulator readings because the simulation would estimate undesired angle of incidence.

Moreover, we decided that it is desirable to have more measurements from a single position since it is more precise to rotate only the sensor than the whole robot. Therefore, we connected the sensor head to the third motor. Whilst this robot's structure provides us with more scans some negative features suddenly occurred. It was noticed that the start position began to drift from the initial position, after multiple scans, which turned the sensor head by 360 degrees and then back to the start position. This resulted in the scans being rotated by an angle with respect to the robot

body which varied over time. This, in turn, degraded the accuracy of the result used in the particle filter calculation. One idea was to introduce a vertical pole into the field of view of the sensor. This could be used to calibrate the position and calculate a compensating factor for the scan rotation. An alternative idea was to use the touch sensor which could be pressed by a lever on the rotating sensor head, and so deduce its exact location. Later, neither of these ideas were deemed to necessary.

Localisation

In order to localise our robot we used a particle filter with 1000 particles; each particle was initialised in a random location and orientation within the map (Figure 3). In order to determine the location and orientation the real robot takes 8 scans in 45 degree increments and the particles also take the same scans in simulation. The result of each scan is the distance between the sensor and the boundaries of the map in each direction, the real robot's scan results are then compared with each particle's simulated scan results by using the Euclidean distance of each measurement vector. The particles are then weighted and particles with distance vectors most similar to the real robot are weighted the highest. The particles are then re-distributed within the map using roulette wheel approach (Figure 4), so that the number of particles in a certain position correlate with the weight of that position. This step is repeated many times (Figure 5) with the robot moving by a small distance (10 cm) and by a random angle until the particles all converge to a position and orientation with low variance within particle position distribution (Figure 6). In each step of the localisation 10% of the particles are randomly redistributed to allow relocalisation in case of error.

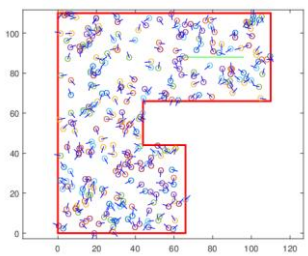


Figure 3 - Particle Initialisation

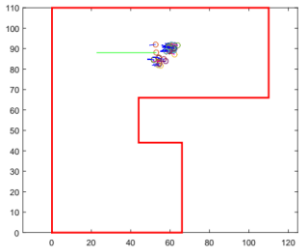


Figure 4 - Resampling at t

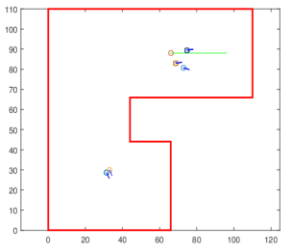


Figure 5 - Resampling at t+1

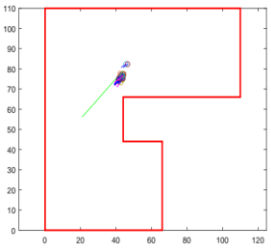


Figure 6 - Converged

The 8 scans work well for initial localisation; however, it started to introduce noise when localising while following a path. Therefore, we decided to use 8 scans for pure localisation and 4 scans for localisation when following a path in order to adjust our estimated position. Since the movement in the map is mostly perpendicular, this approach has produced an improvement in robot motion.

Navigation & Planning

A grid map representation is built as a search space for the path planning. The grid map is constructed from the initial map represented by the corners and allowed positions are represented as binary 1 and obstacles are represented as 0; however, it is necessary to take into account the non-zero real dimensions of the robot. Therefore, any obstacle in the grid map has to be inflated by the dimensions of the robot in order to prevent the planner from considering them.

A mathematical operation of erosion, with kernel size equal to dimensions of the robot is applied on the grid map with the results shown in Figures 6 and 7 (white represents allowed positions).

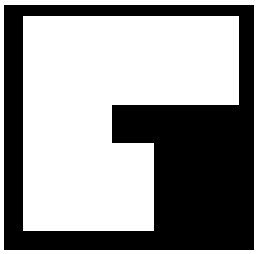


Figure 6 - Original Grid Map

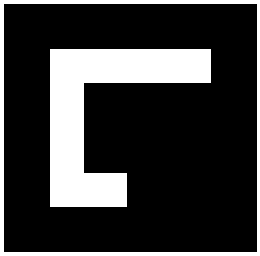


Figure 7 - Grid Map with Inflated walls

The A* path planning is processed over the inflated grid map. It was used both for its speed (when the target location is known) and global optimality. The path is then generated using an open list (to mark positions to be explored) and closed list (to mark positions already explored). The positions in the open list are sorted according to their cost **F** which is calculated using already travelled distance **G** and estimated distance to target **H** as

$$F = G + H$$

The estimated distance is based on heuristics which considers the Euclidean distance from the current position to the target. This allows us to explore the first positions that brings the robot closer to the target. We used an adjacency neighbourhood of 4 in position exploration, this meant that the output of the path planning has been split into segments. Making our paths suitable for smoothing using a Bressenham's line algorithm to make the route smoother where possible (Figure 7).

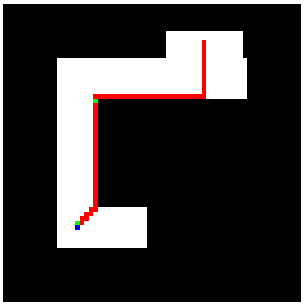


Figure 6 - A* plan output

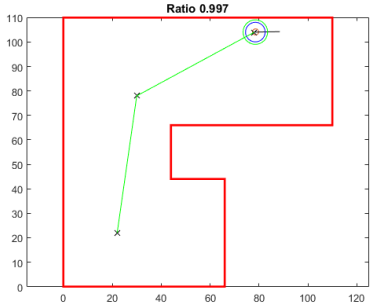
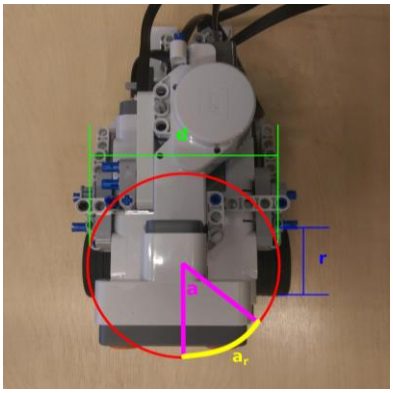


Figure 7 - Bressenham's line filter

Motion Modelling



The output from path planning is in the form of a sequence of motion commands which set the angle to turn and distance to travel. However, these sequences have to be converted into commands interpretable by motors which are controlled by specifying the angle which it should rotate. The conversion is based on real dimensions of the robot depicted in figure on the left. The measured dimensions are the radius of the wheel ***r*** and the distance between the two wheels ***d***. The application of simple differential drive allows the rotation of the motor ***w*** for forward movement to be calculated as

$$w = 360t/(4\pi)$$

Where ***t*** is the distance to be traveled. Similarly, the turn angle ***a*** which corresponds to the arc of length a_r converts to rotation of the motor ***w*** as

$$w = (d\pi)/(abs(a)180r).$$

The direction of the wheel turn is set with respect to the value of the angle $a \in [-\pi, \pi]$. The robot motion showed precision of ±1cm in forward movement, but less precision in turning this was mainly caused by imperfections in the arena floor.

Conclusion

Overall the team worked well together to produce a robot which for the most part gave us successful runs, where the robot was able to arrive within 10cm of the goal in well under 150 seconds. It was able to do this because the robot is able to deal with difficult sensor positions such as readings at 45 degree angles, relocalise if lost and route plan when the robot localises in the inflated map boundaries. Even in the cases it was unable to reach its target it was still able to beep after 40 cm of movement in a position where it has localised confidently in order to meet the secondary marking criteria. The failures are largely because the sensor is unable to take accurate readings when it is not perpendicular to the wall. This caused a lot of noise in the particle filter which made slightly similar locations appear identical causing the robot to crash and further propagate error because our robot could identify crashes or perform a reverse maneuver.