Exercise 1 — Going for a walk

Intelligent Robotics

Oliver Quirke, Lee Lavery, Chris Brookes, Cyril Noel-Tagoe
{oxq161,lxl135,cmb086,cnn007}@cs.bham.ac.uk

1 Introduction

Our first exercise with the Pioneer was met with much frustration due to spending a large amount of time dealing with how the platform and sensors worked and how they returned data before and during our attempts at processing. We spent a large amount of time dealing with returning averaged sectors from the laser – effectively slicing the returned data into n sectors of θ degrees.

Once these data challenges were overcome, our overarching ideas for the robot's behaviour came to fruition and we were successful in achieving the exercise objectives. We also spent time looking at how the laser data is affected by the material it is exposed to, mostly driven by our negative experience with the glass walls in the Lower Ground floor area.

2 Random walk

2.1 Introduction

The idea behind this program was to enable the robot to navigate without a map and localisation in a random direction, whilst still avoiding obstacles (including people) for a prolonged period of time. Whilst we did not record any information quantitatively for this exercise (aside from threshold values), we shall list our qualitative results here verbatim.

2.2 Approach

We approached this in a similar vein to how we tackled this problem during first year in the Robot Programming module. The idea was to let the robot move forwards continuously until it reached (or sensed from a distance) an obstacle, and perform an evasive manoeuvre. We decided to start initially with a 180° turn so the robot would subsequently move in the opposing direction, away from the obstacle.

We were aware that this would become an issue should the robot be placed in a small area from which it would continually turn away from two obstacles (for example opposing walls) infinitely.

We also experimented with proportional speed control, such that the robot would accelerate in open areas, slowing down as it approached obstacles to enhance accuracy and control.

2.3 Results

We had success with the robot in the stage simulator, based mainly on the simplicity of the program. Spurred by our naïvety, we quickly attempted to test the program on the robot. We were also moderately successful with this, however, our chosen threshold value for triggering the evasive manoeuvre was initially too large, such that the robot would turn away from an obstacle whilst still being a considerable distance

from the obstacle.

We experimented with a variety of different values, ensuring someone was in place to stop the robot should it venture too close to the obstacle, eventually ending up with a compromise such that the robot would turn away whilst close enough to the obstacle for it to minimise stoppage due to temporary obstacles (e.g. people walking in front) and far enough away such that people were not hit by the robot, and the laptop did not scrape along the obstacle during the rotation.

We also initially had to define a lower bound for the threshold to ensure that noisy data was not taken into account, which would stop the robot erroneously. This was found using a combination of rviz and manual measurement in debugging with the robot. We calculated that the sensor was reliable above 0.02m.

2.4 Modifications

We also ended up altering the angle the robot rotated in its evasive manoeuvre such that it was random between 90 and 270 degrees. This meant that the robot would no longer become stuck between two opposing obstacles, ignoring a possible 'escape' route.

We left the robot running for a number of minutes at different speed values, and once the collision threshold had been adequately set, the robot did not require any human assistance, either in avoiding obstacles or moving around any blockages. Our proportional speed control also enabled a higher level of control when tested against a fixed speed; in addition, it made the experience of debugging the robot's behaviour more tolerable as it would cover more ground in the same amount of time when compared to a fixed speed robot.

3 Longest path

3.1 Initial attempt (Experiment 1)

3.1.1 Introduction

This program was seen as an extension to the previous part of the exercise, with the robot now attempting to find the longest route to an obstacle, effectively wanting to drive into the area with the most available open space. It was also crucial that the robot must avoid obstacles in its endeavour.

3.1.2 Approach

We completed a simulation in stage to ascertain the robot's sensitivity to distant obstacles. We found that the simulation platform ceased to read above ≈ 6.5 metres, but would report that value back as a maximum. We started our experimentation by implementing a local averaging algorithm, which functioned by looking at each of the laser readings (from 10–170°) and averaging alongside 10 values to the left and right of the target value (where one value represents 0.36° of scan). For example, to find the average for the central direction (0°), we averaged the values from $-3.6°$ to $+3.6°$ inclusively.

We then took the largest of each average value, which enabled us to deal with narrow deep routes — if only the maximum distance was taken directly from the raw data stream, the robot would

not take into account any obstacles around that path, and would potentially try to navigate to it through surrounding obstacles.

In addition to this path finding, we implemented a basic subsumption architecture to ensure the robot would not directly collide with any obstacles during testing. This was achieved by monitoring the lowest value in the scan — when said value dropped below a threshold, the robot would rotate 180° and continue its task.

3.1.3 Results

As with the previous exercise, our simulation in stage was successful. However, when we ran the robot in the real world, we found wildly different results. After much investigation, we discovered that the laser would return values of 0 for distances larger than it could measure. We were not previously aware of this limitation, due to the way in which stage handled maximal readings, as described previously.

3.2 Experiment 2

3.2.1 Introduction

Now aware of the real-world sensor limitations, we altered our approach as to how we detect the largest clear path to follow. This utilised the fact that the laser returns 0 ranges out of its capability.

3.2.2 Approach

We looked for clusters of 0 values from the sensor, as our findings in rviz and in debugging indicated these were the areas we were looking to travel to. We then calculated the largest cluster and took the central position of the cluster as being the largest path to follow — in most real-world cases this was not always the case, but due to the infinitely low resolution of the data, we were forced to use this as our target destination.

3.2.3 Results

We were unable to test this in stage initially due to the fact that the simulation did not return the 0 values we were receiving from the actual sensor. As the robot navigated towards the calculated location, as described above, we found the clusters got smaller as expected, and the perceived resolution of the data became higher, as the target zone became smaller. This, in practice, resulted in a more accurate navigation to the true largest distance.

We placed the robot in a number of compromising situations, such as in corners. Due to the accuracy of the laser sensor at smaller distances and it no longer returning 0 values, the robot was unable to complete its task successfully. We decided that we should alter the algorithm to take into account both 0 values and actual readings when the robot was in closer contact with an obstacle.

3.3 Experiment 3

3.3.1 Introduction

Having found benefits in the two earlier approaches regarding largest area detection, we set out to amalgamate the two methods in order to reduce the probability of failure.

3.3.2 Approach

The approach used in the first experiment was useful in determining the direction of the largest section, and was found to be applicable in use in the `stage` simulation, however the real-world sensor readings did not allow the use of this owing to the 0 issue as described earlier. The 0 cluster detection in experiment 2 was found to work well in the real-world, however when the robot became close to obstacles (resulting in a lack of 0 clusters) the approach fell apart and the robot behaved erratically. We very simply applied both solutions in turn, one being evaluated before the other in a standard `if-then-else` scenario.

3.3.3 Results

We could not run this in `stage`, so were forced to run this on the robot in the real-world. We attempted this experiment poorly and with haste; we initially found that the robot was crashing into obstacles and assigned blame to how cluttered the code had become over the many days of work. We decided to reimplement the path detection and obstacle avoidance methodologies in the last experiment to improve efficiency and comprehensibility.

We do, with hindsight, believe the path detection code to have been satisfactory; only the implementation and upkeep was not.

3.4 Experiment 4

3.4.1 Introduction

We focussed on refactoring and improving all areas of the program such that dealing with low-distance detection and the 0 case was handled simultaneously. The obstacle avoidance was also rewritten for improving the long-term success rate.

3.4.2 Approach

We rewrote the path detection to only deal with averages over n blocks, rather than dealing with each data point returned from the laser individually. We did some minor experimentation to determine the effective range of the laser sensor, finding that it tops out at ≈ 5.5m. We were then able to convert all 0 values to a value slightly greater than this, such that the averaging computed over the block would now take into account the zero values without the use of another method. We also improved our obstacle avoidance to infer a direction of a detected obstacle and turn directly away from that. Previously we had only turned 180° so this was vital in increasing the long-term performance of the robot.

3.4.3 Results

We found the path detection to be optimal at this stage, having improved on the code from the previous experiments. The robot was able to run effectively both in `stage` and in the real-world (with the simulation support returning due to the efficient handling of low-value detection and the 0 cases). The obstacle avoidance was also hugely improved and contributed to a much smoother-running robot. In running this experiment, we also tweaked block sizes for the data averaging, finding that smaller block sizes were less effective as the robot would try to nav-

igate through chair legs.

3.5 Conclusion and Modification

We ran the robot in its final state for a number of minutes multiple times. We worked on tweaking variables regarding angular velocity when approaching an obstacle to ensure the robot moved smoothly but safely away from obstacles. We also brought in the proportional control for speed regarding large open spaces as previously mentioned in the first task. This means that the robot will accelerate and travel more quickly in large open areas, slowing down when walls and obstacles start to appear.

4 Conclusion

We were ultimately successful in both tasks, with the robot being able to navigate autonomously long-term in the first task, and navigate with enhanced intelligence as to maximising its distance travelled between areas of the room, rather than potentially getting stuck in one area, in the second task.

We implemented a basic form of subsumption in taking control of the robot away from the intended direction of the path detection code when an obstacle is detected.

We utilised solely the Hokuyo laser range finder for the task, passing on the sonar whilst still resulting in stable reproducible behaviour.