Probabilistic Robotics Coursework Assignment

Module UFMFNF-15-3

Student name: Wesley Freeman

Student number: 15019249

Tutor: Dr. Martin Pearson

BEng (Hons) Robotics

April 2018

Wordcount: 2,487

Tyson ED209 Robotic Cleaning Machine Design Proposal

Abstract

This document presents the basis of a potential design for a new product based on improving the capabilities of the existing ED-208 cleaning machine.

Solutions are discussed for providing the desired new capabilities of autonomous roaming, localisation, mapping and location recovery.

Close attention is paid to minimising complexity and maximising cost effectiveness throughout.

We outline the software algorithms needed and demonstrate their potential effectiveness in a simulation of the robot.

Changes to the hardware are proposed, along with easily installed modifications to a warehouse space to support the new capabilities.

Contents

1 Requirements Analysis

1.1 Autonomous Roaming

"The platform should be able to move freely around the warehouse without the need of a pre-programmed path or remote control."

To achieve this the robot needs some code for obstacle avoidance and exploration behaviours, and to be able to sense obstacles in its environment.

For this and later requirements, several sensor options are considered:

1.1.1 Sensors

Bump Sensors

Simple microswitches behind bumper rails. Probably not a desirable solution for this heavy machine. Although the command to stop is issued instantaneously, it takes time to lose momentum.

Simple rangefinders

The common HC-SR04 ultrasound rangefinder costs ~£3 and detects objects in a 15-degree cone from a few cm to 4 metres, giving approximate bearing and accurate range. Multiple sonars can be arranged in an array to cover larger angles.

Advanced rangefinders

A LIDAR scanner provides accurate bearing and range over 360 degrees. 2D LIDARs are now quite affordable. The RPLIDAR A2 at ~£270 has a range of 6m, resolution of 0.9 degrees and produces 400 samples per revolution. An excellent option but perhaps overkill for the application.

Cameras & Computer Vision

Cameras can detect obstacles and motion using Structure from Motion (SfM) and Visual Odometry (VO) algorithms. Prices vary from the £22 Raspberry Pi camera to the £450 ZED and beyond. Monocular VO is harder to achieve than by stereo disparity. There are specialised RGB+Depth cameras like the £150 Orbbec Astra which use structured light and similar techniques to extract range information. The necessary algorithms are computationally demanding.

The option I have chosen to use is an array of cheap, simple ultrasound rangefinders.

1.2 Localisation

"The robot should send regular reports (via radio) of its current pose in the environment."

For the robot to report its location it needs a map of the warehouse and a method of keeping track of its own motion. The problem, as will be demonstrated in section 2.1, is that keeping track by relying on odometry alone is impossible. Motors, floor surfaces, sensors etc. inevitably introduce errors which compound over time. This must be corrected by measuring features in the environment and comparing to the record of them on the map.

The 2D position (x, y) and orientation (theta), together called the 'pose', should be represented as a probability, which defines the uncertainty. There are several methods for approximating Bayesian Probability theory, split into Parametric and non-parametric representations of probability distribution.

Parametric methods use Gaussians which take the form of 2 numbers per dimension: a mean, which represents the position, and a variance which defines a falloff in certainty around the mean.

The Kalman Filter and its variants work by combining Gaussians representing different pose hypotheses into a new Gaussian that acts as an average with a lower uncertainty than the inputs.

A real probability distribution can have more than one apparently valid hypothesis, in very different parts of the map. Gaussians cannot represent this.

The non-parametric Histogram Filter quantises pose space into a grid, assigning a single probability figure to each cell. More common is the Particle Filter, which maintains many non-probabilistic pose hypotheses. The overall shape and density of the cloud of particles forms an approximation of a probability distribution. It can cover multiple distinct areas. Particles are spawned randomly over the space. The better a particle matches the evidence from sensors the better 'importance factor' it has and the more likely it is to survive.

Common maps are feature-based or grid-based.

A feature map is an abstracted list of landmarks with their observed features and location. This is well suited to cameras and visual feature extraction. Only the landmarks are recorded and nothing about what might exist between them.

A grid map is like a conventional map with all locations represented. The cells are marked occupied or unoccupied based on sensor data. There is no attempt to distinguish individual objects. This is well suited to rangefinder sensors which report a simple reading/no-reading at some range.

Feature maps are useful for localisation – if we can identify features, we can look up their known location and work out where we must be based on our range and bearing relative to them. Occupancy grid maps are less suited to this, but are excellent for route planning, as we know where all the obstacles are and the grid representation lends itself to path-finding algorithms. A feature map cannot be used for this.

It becomes clear that the best map to use for localisation is a feature-based one, and it follows that a parametric, Kalman filter type algorithm should be used with it. The task remains of detecting landmarks.

Using cameras to extract visual features from the environment is difficult. With warehouses they tend to consist of many near-identical aisles with few unique features.

To get around this we introduce visual features which are unmistakable and reliable, called fiducial markers. They are QR-code-like symbols printed and stuck to walls, shelves, etc. They could be placed on the ceiling and the camera pointed upwards, allowing them to be viewed from more angles, though this may not be practical. Their locations and IDs are recorded in a list which forms the map.

Fiducial markers are also used to solve the problem of getting a robot to line up and dock with its charging station.

Location could be reported by having the robot publish a web page of this and other stats over wifi. It could also send alerts by email if there is a problem.

1.3 Mapping

"The platform should be able to generate its own map of a new warehouse, such that there is no need to upload a new map for each building it will operate in."

It is possible to build sophisticated particle cloud maps from cameras to show photographic detail in a 3D voxel model. These are useful where an immersive simulation of the real location is desired, but such a map would be redundant for this application.

The suitable solution is a 2D occupancy grid map built using rangefinder data. This shows minimal data important for the robot's navigation, and is human-readable.

1.4 Localisation Recovery

"The platform should still be able to re-establish a good estimate of its location if it were to be switched off, moved to another part of the warehouse, and switched back on"

This 'kidnapped robot' problem can be solved by particle filter based localisation configured to always maintain some pose hypotheses scattered throughout the map. Once kidnapped, the main hypothesis confidence will fall while the one closest to the true location rises and spawns new hypotheses, as its sensor evidence matches the map better than the original.

In our case it is much simpler to deal with. The fiducial based localisation system suggested in 1.2 naturally solves the problem, as discussed in 2.2.4.

2 Proof of Concept

2.1 Simulation of Existing System

The current system was modelled in the Player-Stage simulation system.

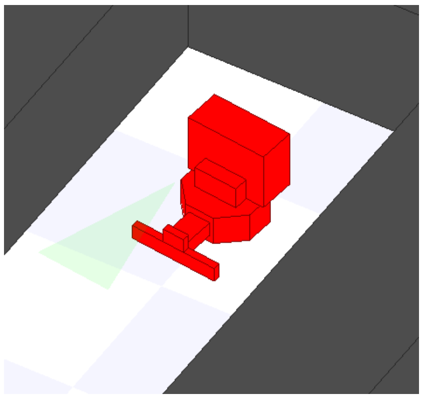

Figure 1: ED-208 modelled in Player-Stage

A warehouse plan was modelled. No doorways have been modelled, to avoid the robot wandering out of the map. In reality, IR gates or similar could be used.

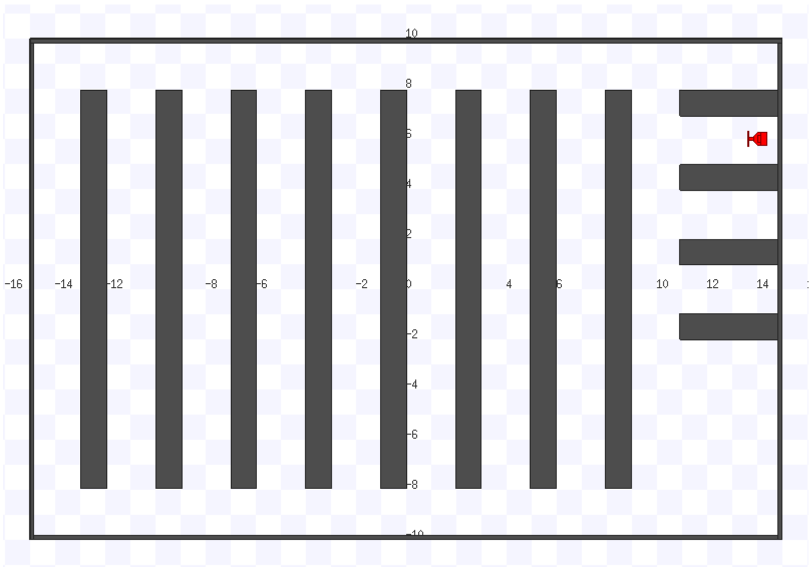

Figure 2: Simulated warehouse floorplan in Player-Stage

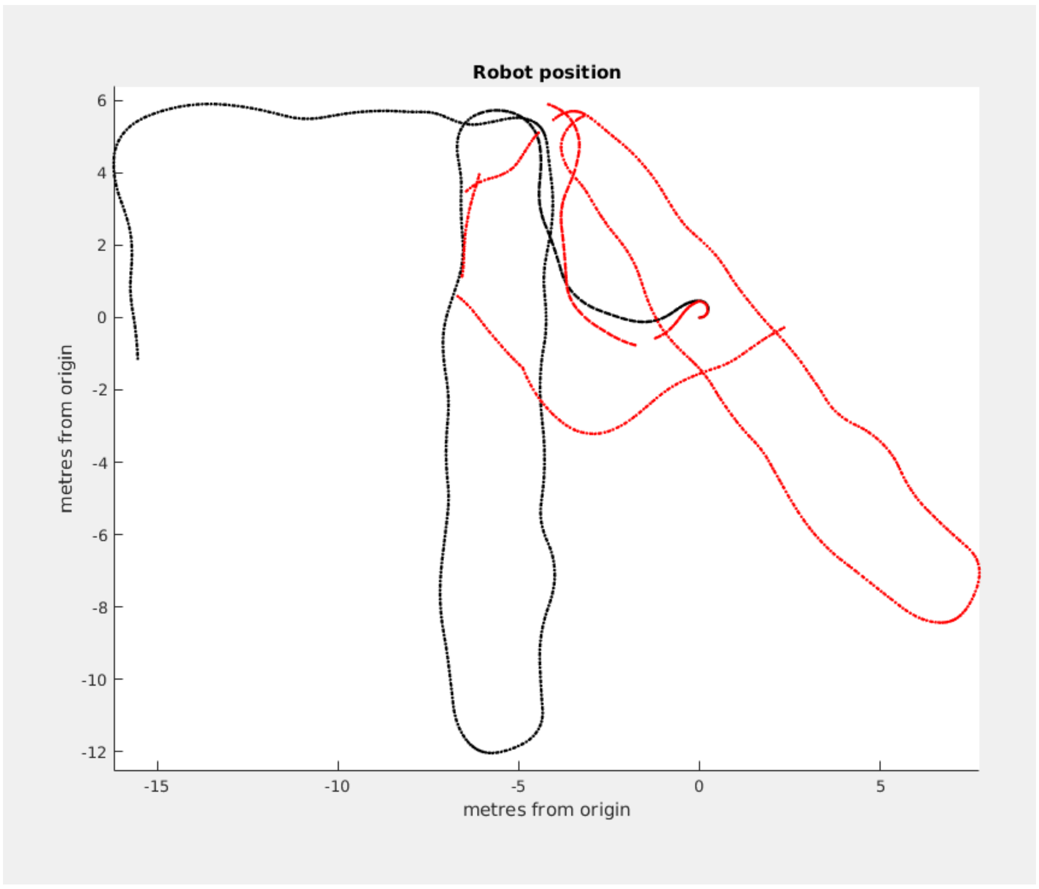

Figure 3: True robot path (black) vs. Odometry (red)

The robot was controlled manually. The black trace in figure 3 shows the true path of the robot reported by the simulator. Semi-realistic noise is added by sampling from a normal distribution based on the previous pose and current odometry, using a motion model algorithm (table 5.6, Thrun et al, 2006). Thus, the red trace represents real odometry from a robot's wheel encoder sensors.

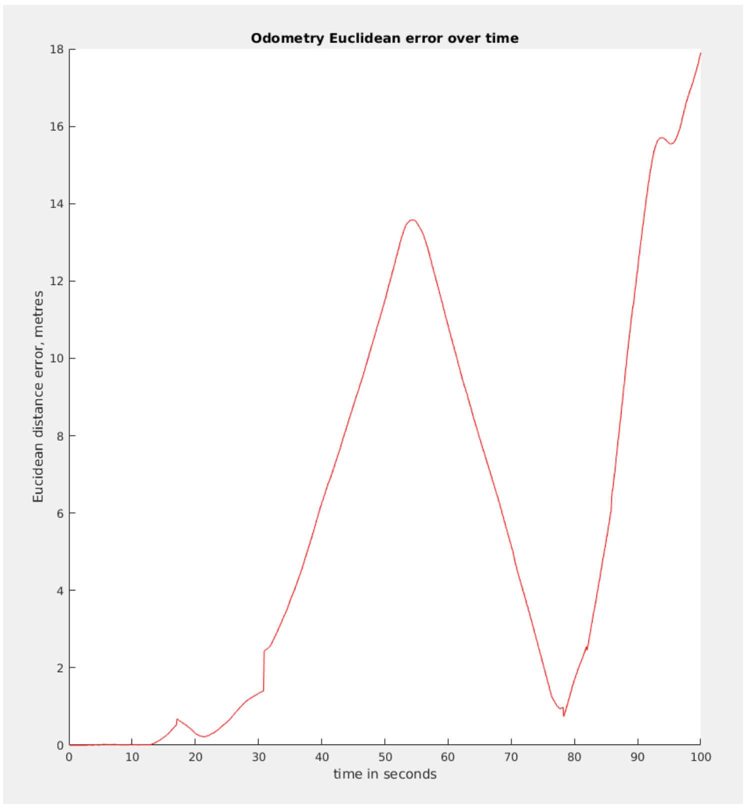

Figure 4: Plot of position error in the Odometry

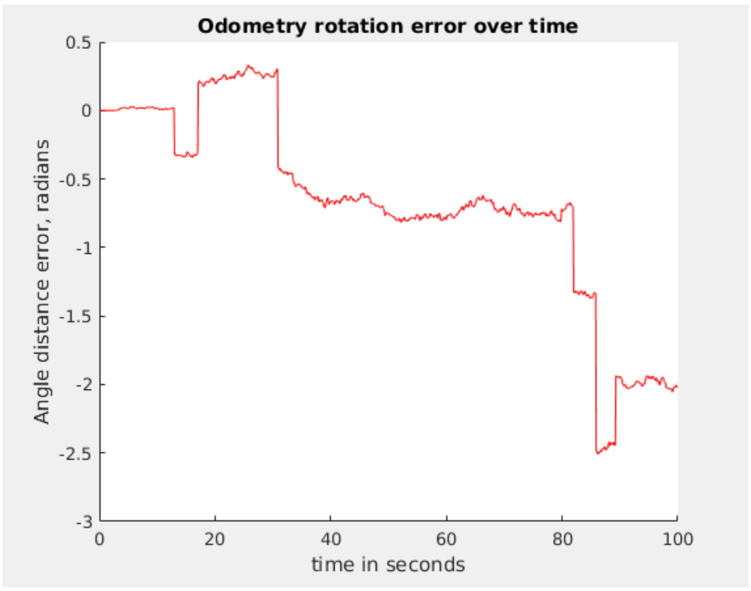

Figure 5: Plot of orientation error in the odometry

An average error of 1% was used. In practice the error figures would be obtained by characterising the motion of the real robot, by repeatedly giving commands and measuring how far off target it is.

Clearly, a robot depending on odometry alone will get badly lost. This illustrates the need for probabilistic techniques that can accommodate uncertainty and correct for it using observations.

2.2 Demonstration of Improvements

2.2.1 Autonomous Roaming

I have used an array of 5 ultrasound rangefinders. This provides enough coverage to avoid collisions.

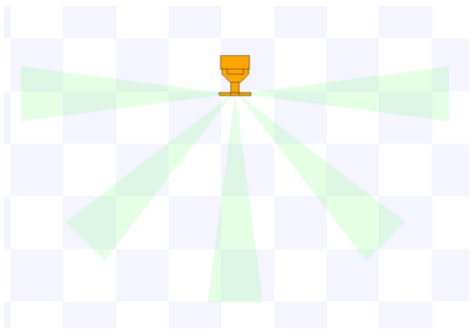

Figure 6: Coverage of rangefinder array

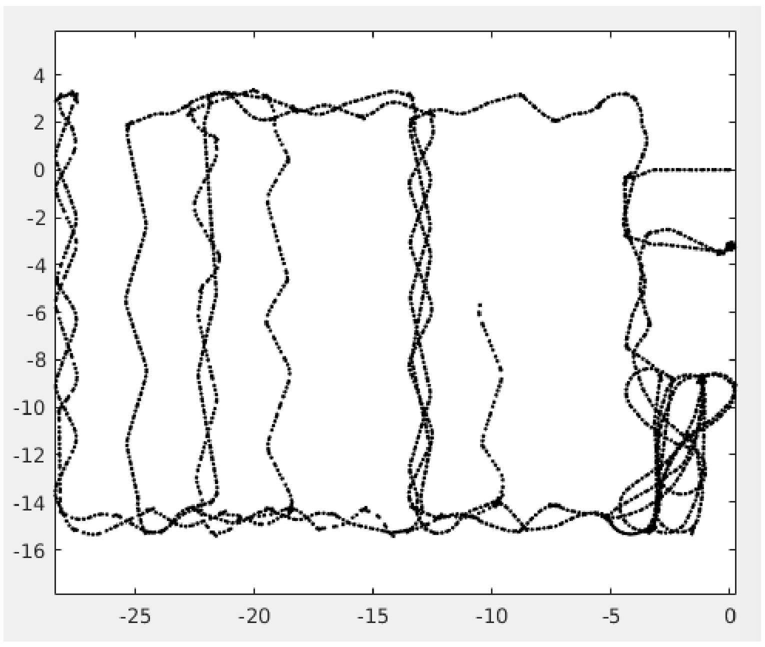

Figure 7: Autonomous exploration using simple rangefinders.

Figure 7 shows the robot exploring the warehouse environment from figure 2. With no knowledge of anything outside its immediate location, it wanders around semi-randomly.

The logic is a state machine with a set of simple behaviours such as turning away from a wall on one side when the sonar readings suggest it is too close (one side shown):

```
if right_front_sonar < 0.5m OR right_sonar < 0.5m)
    set turn_rate high
    if front_sonar < 1.5m
        set speed to slow_reverse
    fi
fi
```

The behaviour to deal with dead ends works in this manner:

```
if all sonars < midrange
    set speed to 0
    set turn_rate high
fi
```

There is a set of behaviours to help it explore. To encourage entering side paths it checks for long-range readings on one side:

```
if sonar_front > midrange AND sonar_left is max
    turn left;
fi
```

The behaviour to deal with T-junctions:

```
if sonar_front < midrange AND sonar_left is max AND sonar_right is max
    set turn_rate to high * random direction;
fi
```

An ‘agoraphobic’ behaviour which discourages spending too long open spaces:

```
if sonar_front > midrange AND sonar_right_front > midrange AND sonar_right > max
    set turn_rate to slow left;
fi
```

An ‘evading’ flag with a timer is included, which suspends exploration behaviours when obstacle-avoidance has been triggered. Another flag & timer periodically changes the reversing direction to help avoid getting stuck in corners.

2.2.2 Localisation

The Extended Kalman Filter is used with a feature map of fiducial markers, and a wide-angle camera which is represented by a simulated fiducial detector which includes measurement noise in its readings. The robot moves around the environment using the roaming behaviour from 2.2.1 influenced only by the rangefinders.

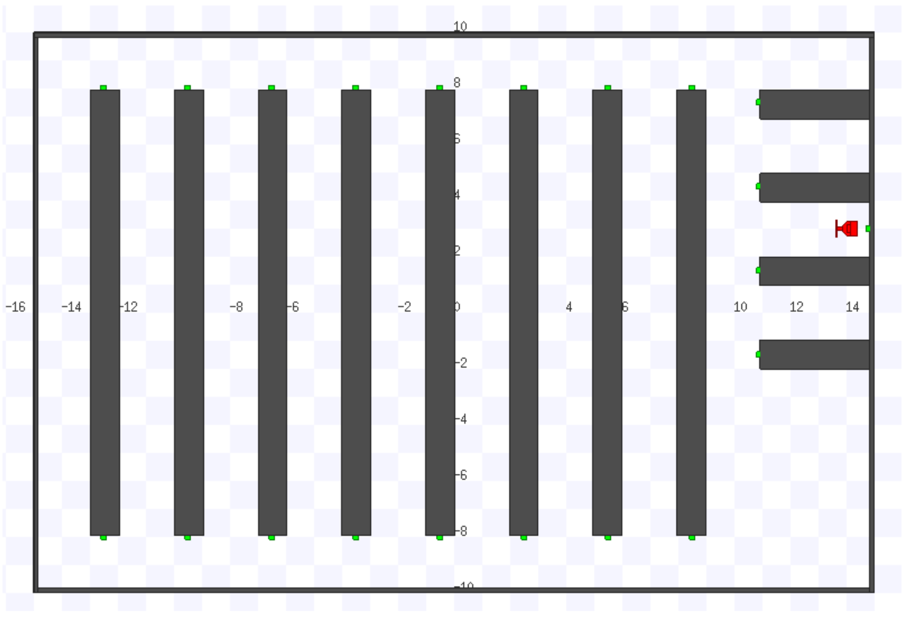

Figure 8: Plan view of warehouse model with added fiducial markers in green.

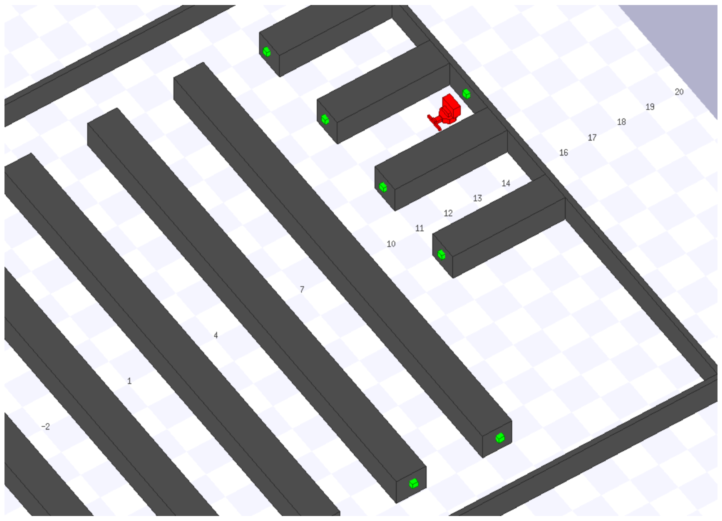

Figure 9: 3D view of warehouse model with added fiducial markers in green.

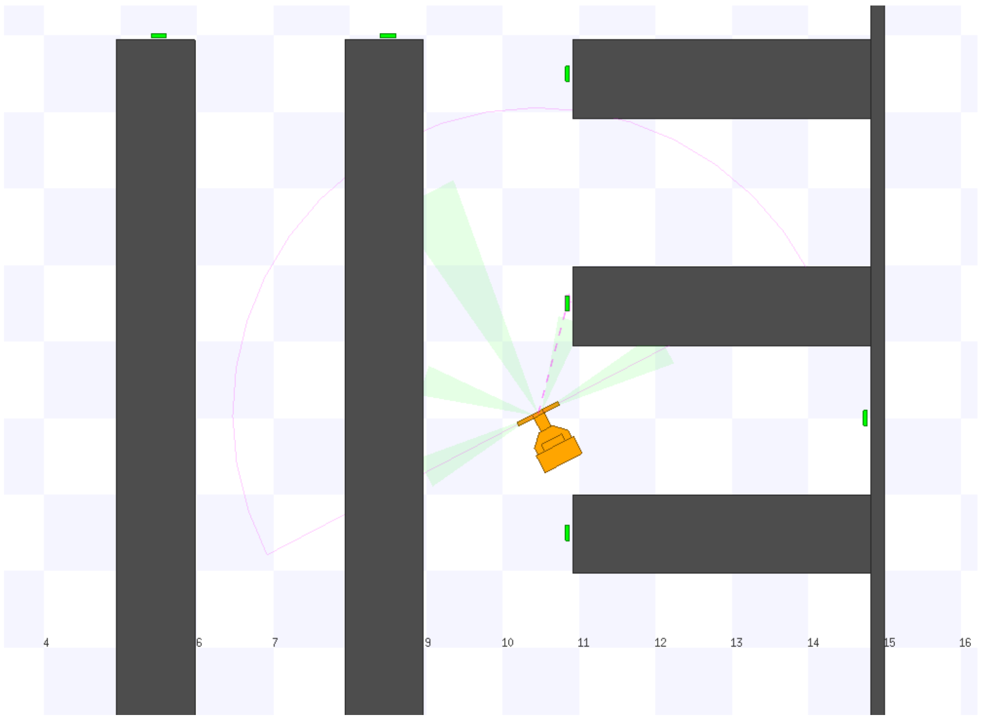

Figure 10: Detection of a marker, indicated by red dashed line.
The semicircle indicates camera field of view.

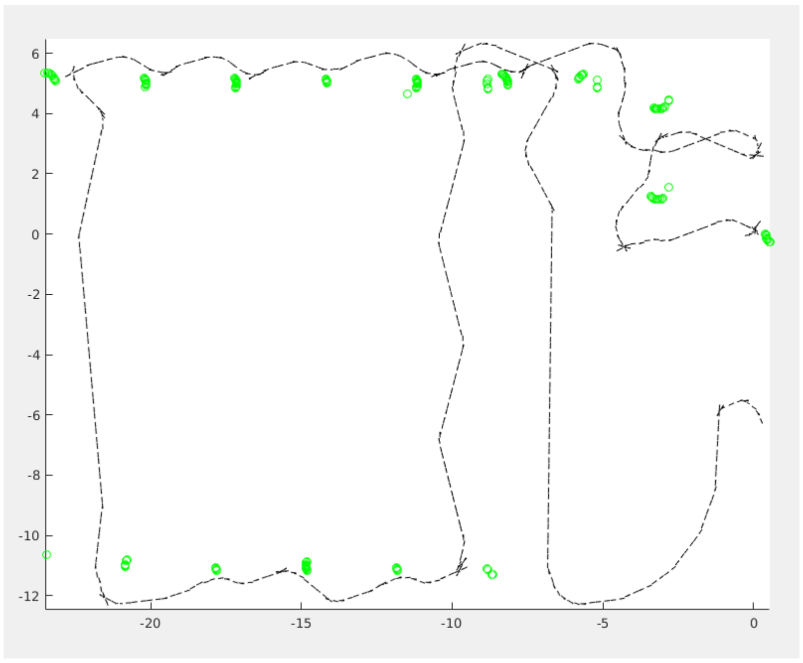

Figure 11: Matlab plot of marker locations according to sensor, plotted relative to ground truth. The jitter in adjacent readings demonstrates the inherent uncertainty in the sensor.

The EKF algorithm with known correspondences (Section 7.4.2, Thrun et al, 2006) was implemented largely successfully. Figure 12 makes it clear that more fiducial markers are needed.

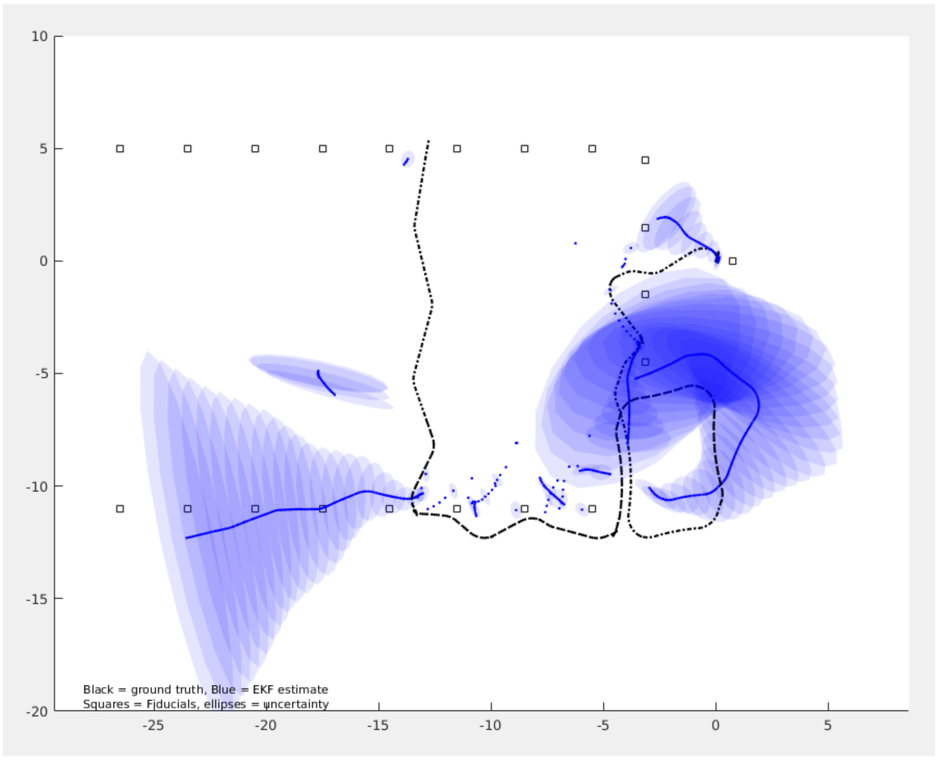

Figure 12: EKF localisation progress over. Starting from origin 0,0

After adding markers, figs. 14-16 show much better tracking, with the error ellipses only really growing in a long aisle with no markers, and we can see it quickly recovers at the end of the aisle.

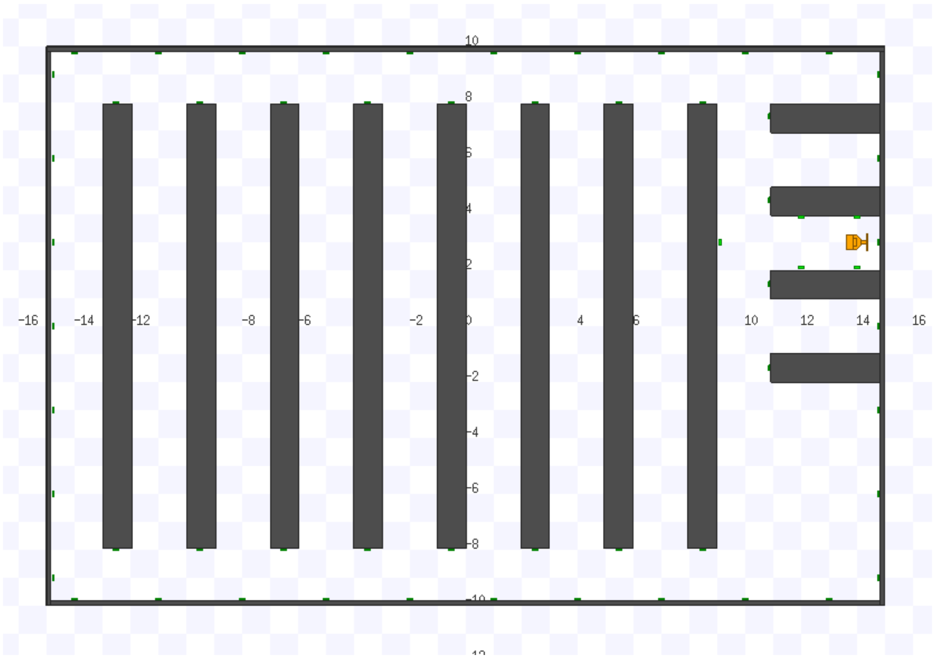

Figure 13: Extra fiducial markers added.

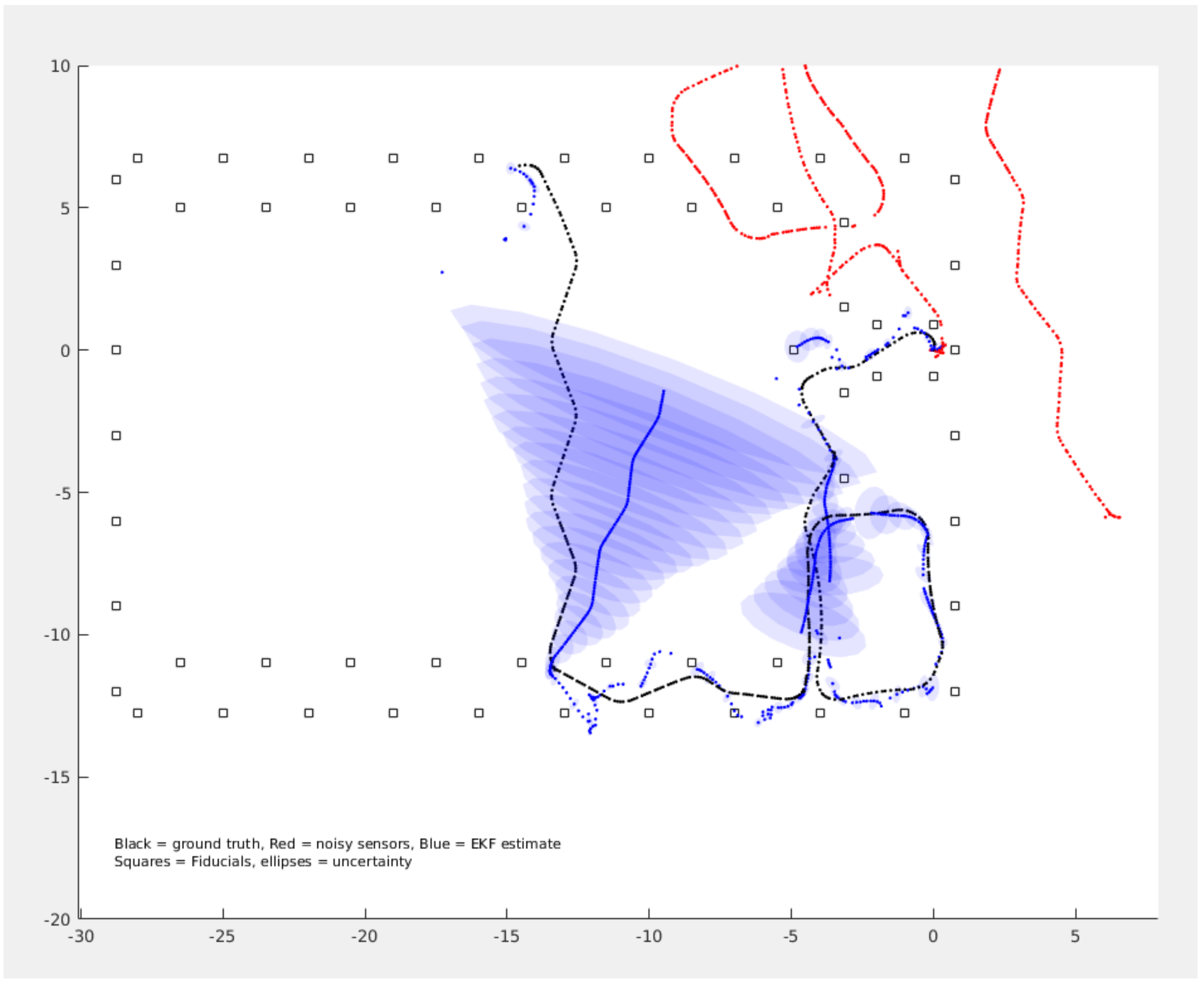

Figure 14: EKF localisation with extra markers.

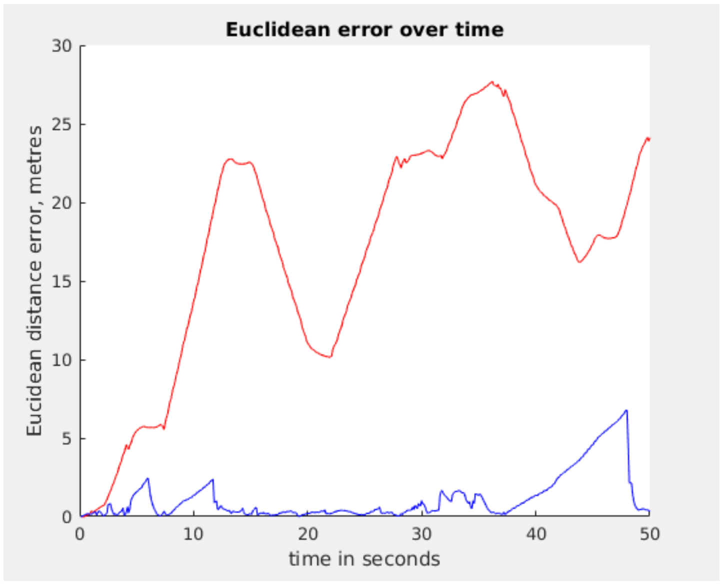

Figure 15: Cartesian error for plot in fig. 14.
Odometry error in Red, EKF error in blue.

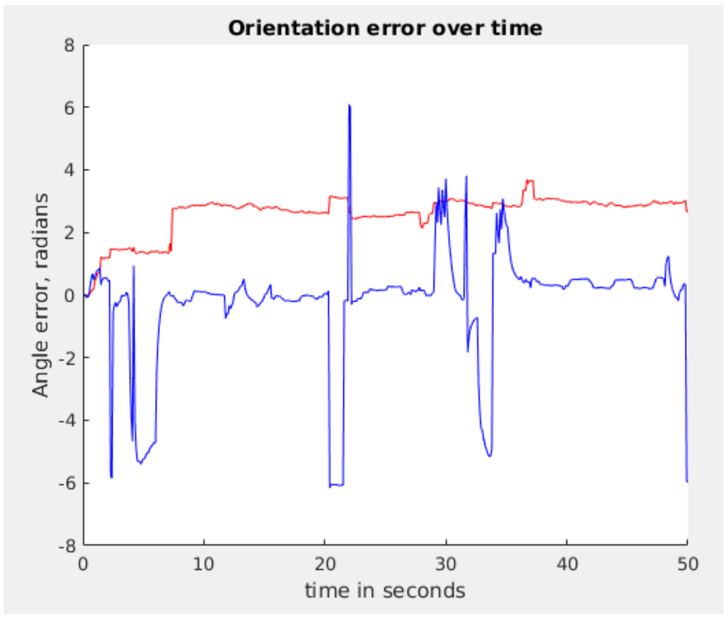

Figure 16: Rotation error for plot in fig. 14.
Odometry error in red, EKF error in blue

My control code first takes the linear and angular velocity commands from the roaming function and adds random noise using Matlab's randn function, which draws from a normal distribution:

```
velNoiseScale = 0.1

velocity_noisy = velocity + velNoiseScale * randn

angular_velocity_noisy = angular_velocity + velNoiseScale * randn
```

Then the already noisy readings from the simulated fiducial detector, supplied as relative coordinates are converted to relative range and bearing:

```
fidRange = sqrt( fidPose_x ^2 + fidPose_y ^2);

fidBearingRad = atan2(fidPose_y, fidPose_x);
```

The EKF works by first moving its estimate of the robot position according to the noisy velocity commands. EKF assumes the robot will not exactly follow them and produces a Gaussian of uncertainty around the expected point, which grows over time as the uncertainty compounds. It then takes any fiducial range & bearing observations available, compares them against what it should see for the known marker location if its motion prediction were correct. It then shifts the pose estimate towards what is suggested by the marker observation and uncertainty decreases.

2.2.3 Mapping

Here I demonstrate occupancy grid mapping using ultrasound rangefinders.

This demonstration ignores the localisation problem, using perfect odometry supplied by the simulator.

The grid starts with every cell having a neutral occupancy value. The mapping algorithm takes the readings from the rangefinders in relation to the known pose and works out which grid squares fall within the sectors formed by the known sensor orientations, field of view and range readings. It then pushes the value of any cells within towards the unoccupied state, and the cells at the leading edge towards occupied.

It is naturally adaptable to temporary obstacles like people, boxes etc. as later scans will find the area empty and the map cells will clear.

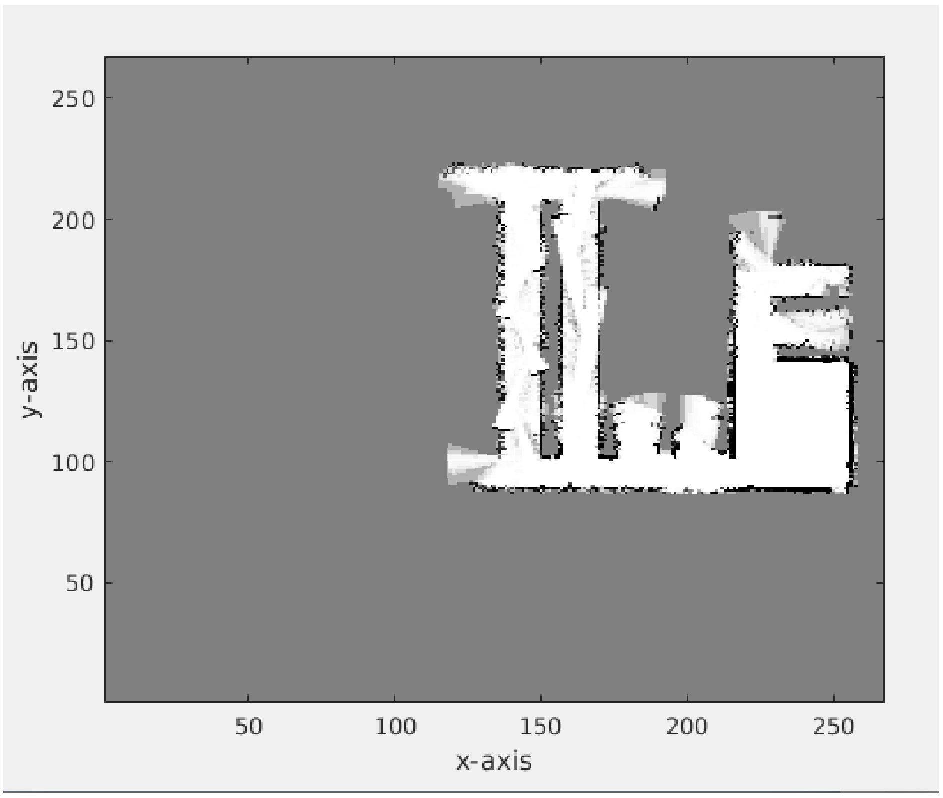

Figure 17: Mapping progress after a few minutes.

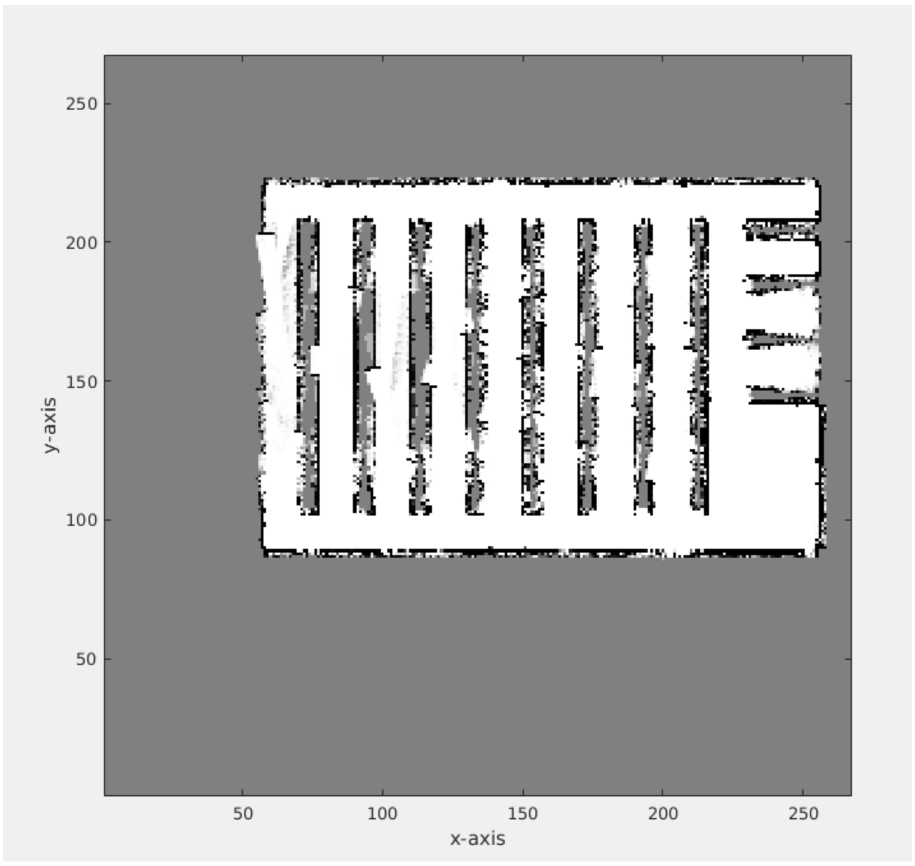

Figure 18: Mapping completed, after 27 mins.

2.2.4 Localisation Recovery

The recoverability of a landmark-map based localisation system depends on the strength of the correspondences. Fiducials have unique IDs so there is no ambiguity about which detected feature corresponds with which map feature. No matter how lost the robot gets, as soon as it sees one fiducial its pose estimate improves rapidly as seen in 2.2.2. To be accurate, 3 fiducials need to be seen at once.

If the robot is occupancy-grid-mapping at the time then the map will get partially corrupted, but this will self-correct gradually once localisation is restored

3 Conclusions and Discussion

The roaming solution works well, eventually exploring the entire warehouse. It is useful for low-level obstacle avoidance behaviour but for a competent system it should be subsumed by more sophisticated path planning behaviour driven by SLAM.

The EKF + fiducials solution works very well as long as there are plenty of landmarks.

The mapping solution works well. Its quality is rough due to using ultrasound rangefinders rather than LIDAR, but should be adequate for navigation purposes.

While the solutions presented meet each of the requirements individually, they need to be expanded into a full SLAM + navigation system for a capable, marketable system.

The exploration behaviour could be overridden with something based on Bayesian exploration. This could have an ‘age’ layer where each area of the map is tagged with the last time it was visited, and another layer that is cost based on distance from current location. These are blended to decide on the next waypoint.

For warehouses that want to run the robot with the lights off, ultrasound works in the dark but fiducial detection doesn’t. Mounting markers on lightboxes would fix this.

The computation required for fiducial detection precludes using a microcontroller, and implementing SLAM would require more. I suggest an ARM based computer for efficiency, such as Raspberry Pi, or Odroid.

Running ROS (Robot Operating System) middleware gives access to many algorithms for easier prototyping, and allows simple distributed computing, offloading demanding tasks to a PC.

4 References & Bibliography

Thrun, S., Burgard, W. and Fox, D. (2006) *Probabilistic Robotics*. Cambridge, MA, USA: The MIT Press.

Pearson, M. (2018). Probabilistic Robotics (lecture slides). *BEng Robotics* [online]. Available from: https://my.uwe.ac.uk [Accessed 19 Apr 2018].

Sol`a, J. (2014) *Simultaneous localization and mapping with the extended Kalman filter* Available from: http://www.iri.upc.edu/people/jsola/JoanSola/objectes/curs_SLAM/SLAM2D/SLAM%20course.pdf [Accessed 19 April 2018]

Ho, N. (2017) *EKF Localisation with Known Correspondences* Available from: http://nghiaho.com/?p=2355 [Accessed 19 April 2018]