**ENMT482 Assignment 1**

**1. Sensor fusion**

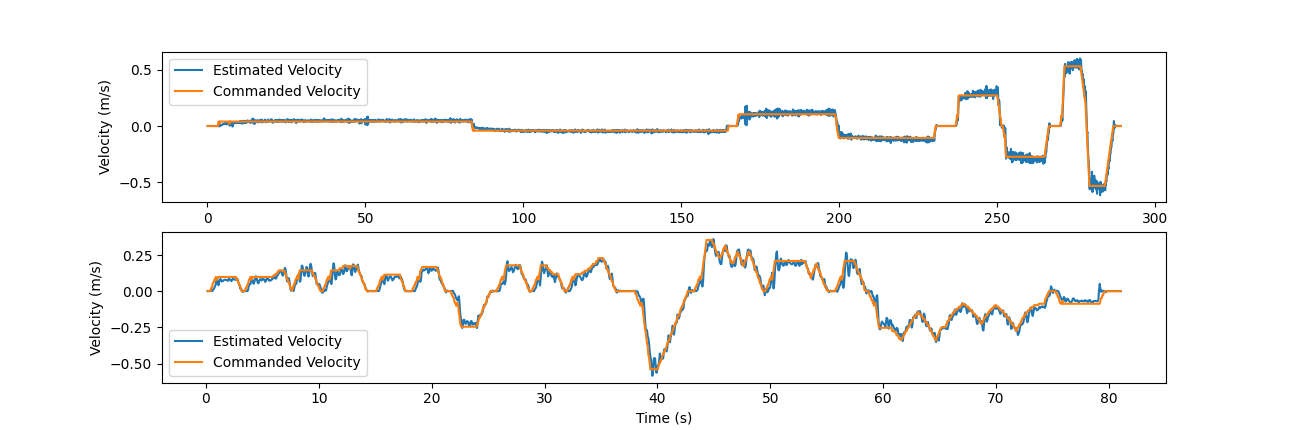
**1.1 Sensor models**

The sensors chosen are the Sonar1, IR3, and IR4 due to their small variances. A hyperbolic fit in the form is used to model IR3. A piecewise function about 0.8m is used for to model IR4 with a quartic in the first quadrant and a hyperbolic fit the similar to IR3 in the second. The sonar model was simply found by iteratively fitting a linear model ( to the data and removing outliers. The variance associated with each sensor is not uniform with varying range as clearly seen in IR3 (Appendix A). To adjust for this, the error in each model was divided into range dependant quadrants and the variance found for each. This enabled a lookup method of dealing with range varying variance. Each of the functions were then inverted to the form with the variances of non-linear models found by linearising the model about the current best estimate at each step such that .

**1.2 Motion model**

Both training1.csv and training2.csv datasets were used to find an appropriate motion model. Velocity estimations were attained by finding the gradient of the range column using the timestep at each interval. From here the estimated velocity was compared with the commanded velocity at each timestep. This showed consistent undershoot in the estimated velocity. To account for this the estimates from each dataset were concatenated, and the estimated velocities were then plotted against the commanded velocities to find the linear fit . This adjustment model removed the undershoot but a lag in response still remained as seen in Figure 1. This lag in response is due to the inertia of the robot. Some form of compensation was applied by reducing the variance of the motion model whenever the difference in previous and current was below a threshold.

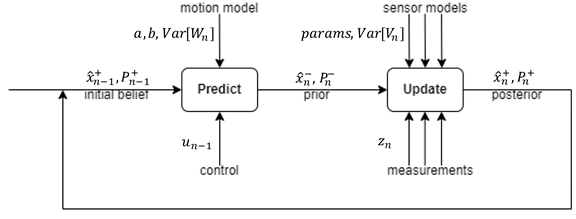
**Figure 1:** Motion model estimated speeds compared to commanded speeds for training1.csv (top) and training2.csv (bottom)



**1.3 Bayes filter**

An Extended Kalman Filter (EKF) was implemented to combine the sensor models and motion model using a Best Linear Unbiased Estimator (BLUE). The overall process is described by Figure 2.

**Figure 2:** Outline of Kalman Filter predict and update process.



**Predict:** The predict stage takes in the initial belief and uses the motion model and commanded velocities to predict the prior estimate. The variance from the motion model is simply added to the initial variance and scaled down depending on the accelerations present as described in Section 1.2.

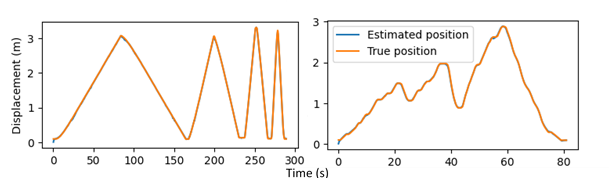
**Update:** The update phase combines all the sensors with the prior in one step using the BLUE Equations 1 and 2 to find the posterior. This effectively gives each of the sensors and the prior estimate a weighting depending on the variance for each. At this stage outliers are detected and omitted in each of the sensor models.

(1)

(2)

**1.4 Results**

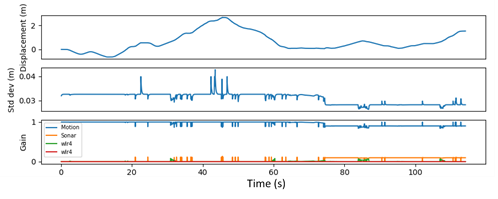
Figure 3 shows the accuracy of the model for the training datasets. Here it can be seen that the EKF and models accurately estimate the position of the robot with a few exceptions. The mean residuals are 0.37cm for training1.csv and 0.01cm for training2.csv. The estimated position, relative weightings of the sensors and motion model for test1.csv can be seen in Figure 4 along with .



**Figure 3**: Estimated position, standard deviation and gains of the model using the test.csv dataset

**1.5 Discussion**

**Figure 4**: Comparison of estimate and true position for training1.csv (left) and training2.csv (right).



Notably the largest difference between estimated position and actual position occurs at 0m for both training datasets however this is quickly corrected by the model. The Gain plot shows that the motion model is being prioritized with a significantly higher weight throughout the whole dataset. The standard deviation for the test.csv dataset shows it oscillating about 0.0325m with a decrease to 0.0282. This decrease corresponds with an increase in the sonar gain (and subsequent decrease in motion model). This indicates that improvements could be made by refining the sonar model further so that it carries more weight. Applying the range varying variance improved the model particularly with the IR3 sensor which has the most range dependent variance. To further improve on this a MAF could be applied to find more continuous model for the variances. Additionally, the motion model could be improved by taking into consideration the actual maximum acceleration of the robot when scaling its variance.

**2 Particle filter**

**2.1 Sensor model**

The sensor model determined the weighting of each particle. Each weight was dependent on the pose of a beacon relative to the robot and the pose of the beacon relative to its particle. The relative beacon poses were each converted into a range and bearing. By converting the relative robot and particle measurements, the joint likelihood function between the beacon pose with respect to the robot, and the particle poses were decoupled. The weighting of each particle was determined by how close its range and bearing values matched the measured values of the robot. Particles with small error values were weighted highly, particles with big errors values were given a low weighting. The particle weights were updated at each time step using Equation 3.

(3)

The PDFs for the range and bearing error were modelled as Gaussian distributions to emulate sensor noise. Standard deviations for each PDF were 0.05 and 0.05 for the range and bearing respectively and chosen through trial and error. This resulted in a particle being given an equal weighting from each PDF if its range and bearing errors were 5cm and 3° respectively.

**2.2 Motion model**

A probabilistic odometry motion model was used for particle position prediction. A probabilistic model was used to cater for the uncertainty generated by gyroscopic drift and wheel slippage. The odometry model was implemented over the velocity motion model due to its higher accuracy and use of an EKF fusing rotary encoder and IMU measurements. The model estimates a change in global pose by parameterizing the local pose change into three independent values, two rotations (), and a translation (). A Gaussian PDF for each parameter was generated to sample a value for and . Each time step, the particle cloud spread out act as hypotheses for the robot’s local pose. The pose of each particle was updated with Equations 4.1-4.3.

(4.1)

(4.2)

(4.3)

The standard deviations for each parameter’s PDF were determined through trial and error. The chosen standard deviations for rotation () and translation () were 0.001 and 0.004 respectively.

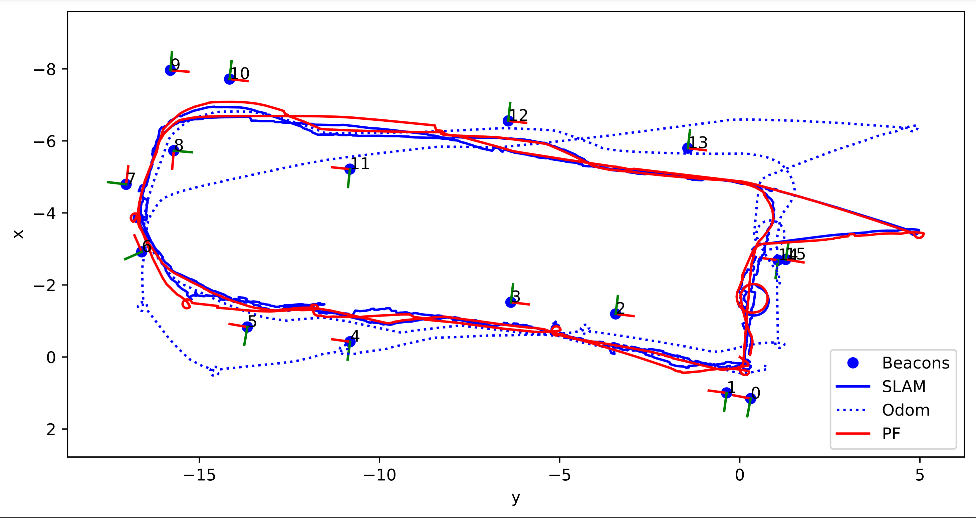
**2.3 Implementation**

The particle filter was implemented knowing the starting position of the robot. A particle cloud of 100 particles was uniformly distributed within a 60cm2 area over the starting position to account for uncertainty. For each time step, the motion model predicted the position of each particle. The sensor model updated the particle weights. Particles were resampled if the weights were degenerate. If the sum of the weights dropped below a threshold, particles were randomly spread around the last odometry position sampling from a PDF with standard deviation 10 times greater than the motion model PDF.

**2.4 Results**

The particle filter was successful in closely following the estimated SLAM model. Figure 5 shows the trajectory of the particle filter. Figure 5 shows two key behaviors of the filter when in operation. Around the corners where a beacon is not visible to the robot the filter relies on the motion model which overestimates the robot’s position. This is the fundamental reason to inaccuracies in the model. Sharp diagonal jumps in the filters estimated position is a result of a beacon becoming visible to the robot. This jump corrects the error accumulated by the motion model when a beacon is not visible.

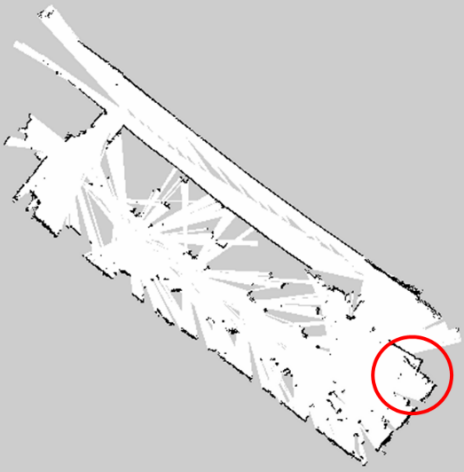
**2.5 Discussion**



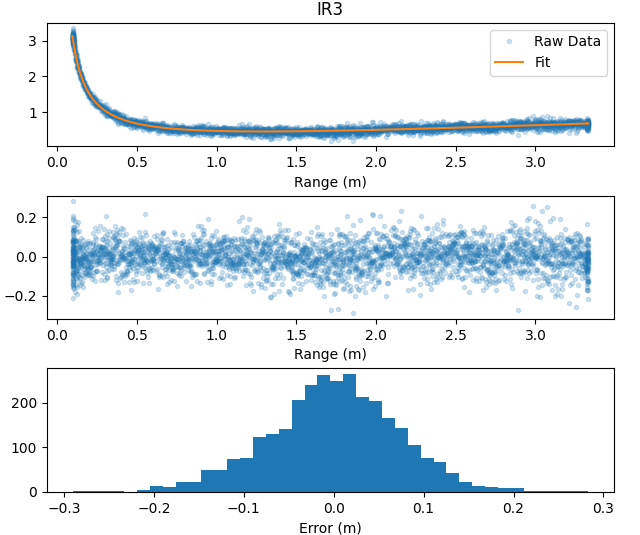
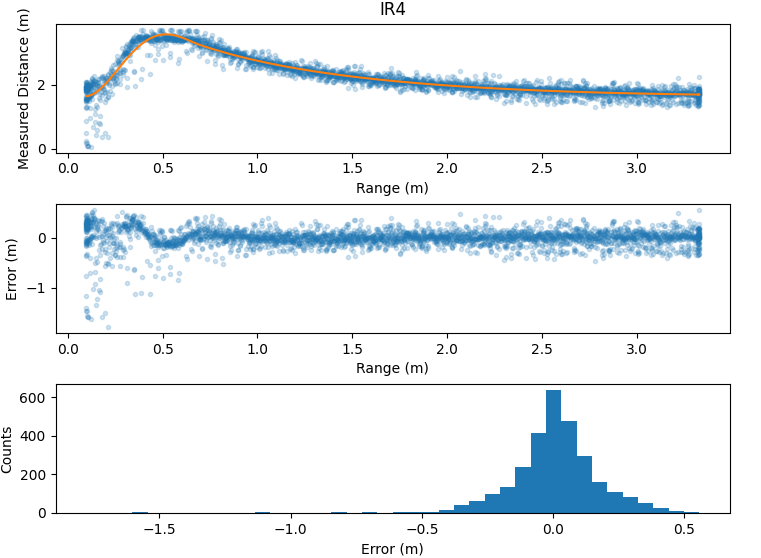
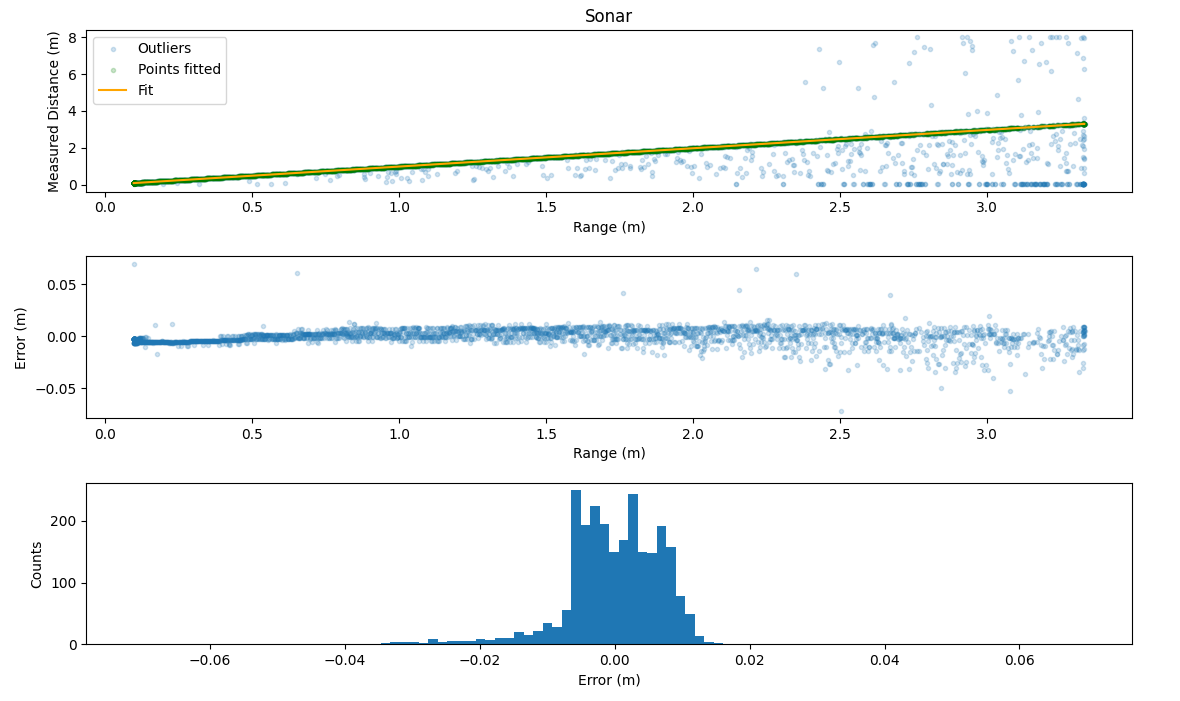
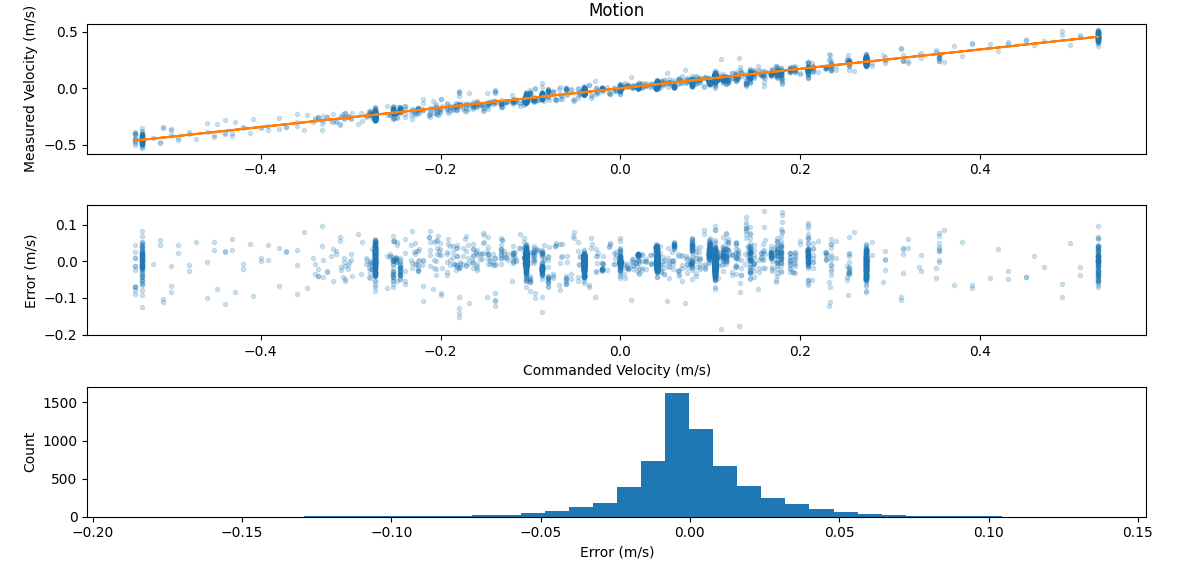
**Figure 5**: Particle filter tracking results.

The particle filter was successful in tracking the robots pose from a known starting position with uncertainty of ±0.3m in the x and y directions. Having small variances on sensor model PDFs ensured the estimated pose of the robot was certain when beacons were visible. This filter could be improved to track the robot from an unknown start position. This could be achieved by placing 100 particles uniformly spread over the map. To resample, particles could be randomly distributed around the posterior belief with the upper and lower bounds of the distribution being proportional to the inverse of the squared sum particle weights. This would create a spread of the particles that dynamically change with the certainty of the filter. Implementing this method would reduce the likelihood of the particles converging to an incorrect initial position. The odometry model was accurate in estimating position when beacons were not visible however could be improved by using a BLUE estimator to fuse the odometry pose with the commanded speed of the robot.

**3 SLAM**

The SLAM program provides a detailed map of the robot’s surroundings in the form of an occupancy grid containing free cells (white), occupied cells (black) and unknown cells (grey). Notable inaccuracies in the model are the curved walls and non-perpendicular corners. These errors are due to the drift in the robots odometry model overtime. This is most obvious in the highlighted bottom right corner where the robot started and finished. The drift may be due to slippage in the wheels causing the encoder readings to not match that actual motion of the robot. The patches in the model are due to shielding from occupied cells which appear closer in the robot’s field of view casting “shadows” behind them.

**Figure 6**: SLAM map with circled error.

**4 Appendix (Model Fits and Errors)**