GROUP 2

ENMT482 ROBOTICS ASSIGNMENT 1

Liam Hare: 27686710
William Johanson: 12015205
10/09/2020

SENSOR FUSION AND PARTICLE FILTER

Part A

Sensor Models

Sonar1, sonar2 and IR3 we selected as the three sensors to be modelled as they collectively covered the range required of 0-3m. The calibration data provided was used to characterise the sensors as follows.

Sonar1 and sonar2 have an observation model of the form;

$$Z = h(x) + V \quad (1)$$

Where Z is the random variable describing the sensor measurement, h(x) is a model describing the relationship between the state of the robot and the sensor measurement, and V is the random variable describing the additive noise. Sonar1 and sonar2 were determined to have a linear relationship between the state of the robot and the measurement from the sensors. The sensors were both modelled using the Equation 2;

$$h(x) = z = cx + d \quad (2)$$

Where x is the state and z is the measurement, both in meters. The unknown parameters, $c\ and\ d$, were determined using the calibration data and method of least squares for two iterations. Outliers were removed using Iglewiz and Hoaglin's modified Z-score after each iteration to help produce a more accurate model. The sensor models matched the true range well as shown in Appendix A, Figure 1. The outlier's rejection was required as the sonars had a bias of outliers below the expected measurement.

IR3 has a non-linear relationship between the state and measurement with a model equation,

$$h(x) = z = k_0 + \frac{k_1}{k_2 + x} \quad (3)$$

Where $k_0, k_1, and\ k_2$ are the unknown parameters determined by the method of non-linear least squares within the usable range of 10-80cm. The sensor model for IR3 was then linearized, as required by the Kalman filter, using the first derivative Taylor series approximation of Equation 4. The successful linear approximation of IR3 is depicted in Appendix A, Figure 2. Where,

$$h(x) \approx h'(x_0)(x - x_0) + h(x_0)$$

$$h(x) \approx \left(-\frac{k_1}{(x + k_2)^2}\right)(x - x_0) + h(x_0) \quad (4)$$

Changing this into the same form as the sonar sensors gives;

$$Z \approx cx + d + V$$

$$c = h'(x_0)$$

$$d = h(x_0) - x_0 h'(x_0)$$

The Gaussian additive noise was modelled by first creating an error vector by taking the difference between the measured values, z, and initial sensor model, $h(x)$. This was then used to determine the mean and variance of the noise. The PDF of the noise for each sensor was plotted using Equation 5 and is depicted in Appendix A, Figure 3. The variance and mean values before and after filtering are depicted in Table 1.

$$f_V(v) = \frac{1}{\sqrt{2\pi\sigma_V^2}} \exp\left(-\frac{1}{2}\frac{(v - \mu_V)^2}{\sigma_V^2}\right) \quad (5)$$

Table 1: Comparison table of sensor noise variance and mean pre and post filtering

Sensor	μ_V(unfiltered)	σ_V^2(unfiltered)	μ_V (filtered)	σ_V^2 (filtered)
Sonar1	0.843	-0.193	0.000113	-6.61*10^-16,
Sonar2	0.237	1.03	9.76*10^-6	5.94 x 10^-16
IR3	7.788*10^-15	0.00468		

Motion Model

The motion model was created by comparing the command distance, velocity, u(t), multiplied by the change in time, to the true distance travelled, b,

$$u(t)\Delta t = mb + c \qquad (6)$$

Using method of least squares, the constants m and c were approximated as $m = 1.07$ and $C = 0.0699$. For each time step equation # is rearranged to solve for b which is then used to solve for the prior estimation, $\hat{x}_1^-$, where,

$$X_1^- = aX_0 + b + W_1$$
$$\hat{x}_1^- = a\hat{x}_0^- + b \qquad (7)$$

The model was tested using training data1 and plotted against the true distance, Appendix A, Figure 4. The variance and were calculated to be 0.033 and -0.078 respectively.

Estimation approach

The position of the robot was estimated using a Kalman filter which combines the motion modal and sensor model to determine a better approximation of the position than if the motion model or sensors were used individually. For each sensor reading the maximum likelihood estimate (MLE) is determined and then fed into a best linear unbiased (BLU) estimator which weights the sensor estimate relative to their variances. Where the MLE and corresponding variance is calculated as follows.

$$\hat{x} = \frac{z-d}{c}\ (MLE)$$
$$\sigma_{\widehat{X_n}}^2 = \frac{\sigma_V^2}{(c)^2}\ (Variance)$$

The BLU estimator is then implemented as shown below where ω_n is the respective weighting.

$$\hat{x} = \omega_1\hat{x}_1 + \omega_2\hat{x}_2 + \omega_3\hat{x}_3 \qquad (8)$$
$$\omega_n = \frac{\frac{1}{\sigma_{X_n}^2}}{\frac{1}{\sigma_{X_1}^2} + \frac{1}{\sigma_{X_2}^2} + \frac{1}{\sigma_{X_3}^2}}$$

The posterior estimate, $\hat{x}_1^+$, was determined using a Kalman filter, Equation 9, which combines the results from the prior estimate and the sensor fusion via a weighting called the Kalman gain, K.

$$\hat{x}_1^+ = K_1\hat{x}_1(MLE) + (1-K_1)\hat{x}_1^-(prior) \qquad (9)$$

The estimate was then iteratively solved where the starting position of the motion model, X_0, is determined by the first measurement from the sensor fusion as the motion model has no idea of the robot's position. The result of applying the Kalman filter on the test data is displayed in Appendix A, Figure 5. The discontinuities of the resulting estimate and invalid Kalman gain values are discussed in the following section.

Discussion

The final model is a relatively poor implementation of the Kalman filter. This was due to a combination of poor filtering and unexpected bugs that formed in the final code. Significant improvements could be made by improving the variance calculations of both the sensor model and the motion model.

The variance of the sensor models are assumed to be constant, for simplicity, whereas in reality the variance changes with respect to range for each sensor. A varying variance function for each sensor would allow the BLU estimation to give low weightings to the sensor values when the robot is not within their accurate bounds. The is emphasised as the noise in the final Kalman filter increases as the robot's range increases as the IR3 becomes inaccurate outside its specified range. An improved variance model therefore results in the removal of outliers unwanted outliers contributing to the position estimation in the Kalman filter.

The motion model assumes a linear relationship between commanded distance and true distance which provides a model that overshoots the true distance. This results in a large variance of 0.033 compared to the BLU estimation and therefor the Kalman filter is heavily weighted toward the sensor. Therefore, an improvement in the motion model would help to improve the overall variance of the estimation model.

Part B

Motion Model

The odometry motion model for the robot was utilised due to its improved accuracy over the velocity motion model by utilising actual measurements. The odometry measurements given were fused readings from the odometry wheel sensors and IMU which estimated a 2-dimensional local pose for the robot as in Equation 10:

$$\boldsymbol{x}'_n = (x'_n, y'_n, \theta'_n)^T \qquad (10)$$

Given the current (n) and previous $(n-1)$ local poses of the robot, the probability density of the transition between poses could be modelled. The error in the local transition was assumed to be small. Therefore, the global pose was parameterised identically to the local pose change as in Equation 11:

$$\begin{bmatrix} x_n \\ y_n \\ \theta_n \end{bmatrix} = \begin{bmatrix} x_{n-1} \\ y_{n-1} \\ \theta_{n-1} \end{bmatrix} + \begin{bmatrix} d\cos(\theta_{n-1} + \phi_1) \\ d\sin(\theta_{n-1} + \phi_1) \\ \phi_1 + \phi_2 \end{bmatrix} \qquad (11)$$

Where d is calculated as in Equation 12:

$$d = \sqrt{(x_n - x_{n-1})^2 + (y_n - y_{n-1})^2} \qquad (12)$$

To determine ϕ_1 and ϕ_2, the angle φ from the global x-axis to the direction of travel between the two poses (where the magnitude of the line is d) was calculated first as in Equation 13:

$$\varphi = \tan^{-1}\left(\frac{y_n - y_{n-1}}{x_n - x_{n-1}}\right) \qquad (13)$$

Giving ϕ_1 in Equation 14:

$$\phi_1 = \theta_{n-1} - \varphi \qquad (14)$$

And giving ϕ_2 in Equation 15:

$$\phi_2 = \theta_n - \varphi \qquad (15)$$

The angle subtractions were calculated by an angle difference function which ensured the result was between $[-\pi, \pi]$.

The first particle pose for the 100 defined particles was the initial slam pose, giving an accurate starting point. The next estimate of the particle poses was determined by adding the transition between poses to the current particle estimate. Noise was added to the estimates due to the residual error in the motion model. The errors were Gaussian modelled with error means $\mu_x = \mu_y = \mu_\theta = 0$. The variance of each direction were altered to best model the true behaviour of the estimate error in each respective direction giving; $\sigma_x = 0.025, \sigma_y = 0.01, \sigma_x = 0.05$.

Sensor Model

From the motion model, the particles move a certain distance plus additive noise. For each of the particle measurements from the motion model, the weightings of each particle needed to be updated. Firstly, the relative pose measurements were broken into range and bearing as in Equations 16 and 17:

$$r = \sqrt{({}^r x_b)^2 + ({}^r y_b)^2} \quad (16)$$

$$\varphi = arctan2\,({}^r y_b, {}^r x_b) \quad (17)$$

Where ${}^r x_b$ and ${}^r y_b$ are the measured pose of the beacon in the robot's camera coordinate frame. The range and bearing measured by the m^{th} particle are described in Equations 18 and 19:

$$r^m = \sqrt{(x_b - x^m)^2 + (yb - y^m)^2} \quad (18)$$

$$\varphi\,(m) = angdiff\left(arctan2\left(y_b - y^{(m)}, x_b - x^{(m)}\right), \theta^m\right) \quad (19)$$

Where x_b and y_b are the true locations of the currently visible beacon in the map coordinate system. Whilst, $x^{(m)}$, $y^{(m)}$ and θ^m are the mth particles 2D location and orientation in the map coordinates. Each particle determines its range and bearing to the measured beacon. The particles weight depends on how close its range and bearing are to the measured values. The particles with smaller discrepancies get higher weights according to Equation 20:

$$a_n^{(m)} = a_{n-1}^{(m)} f_R\left(r_n - r_n^{(m)}\right) f_\Phi\left(angdiff\left(\phi_n + \phi_n^{(m)}\right)\right) \quad (21)$$

Where $f_R(r)$ is the range error PDF and $f_\Phi(\phi)$ is the bearing error PDF. The particle weightings therefore get proportionally smaller as the range increased. The mean values for both the PDF functions were set to 0. Additionally, the variance was set to the square of the radius between the robot's position and the markers position. Due to the nature of the PDF's, as r increased, the poorer the results got.

Estimation Results

The first stage of the estimation approach was to determine a motion model which accurately tracked the odometry. Thus, confirming the equations implemented for the pose transitions were correct. Within the estimation this was very accurate. Therefore, the particles could be updated with the motion model estimation alongside the defined noise values. The resulting estimation plot is shown in Appendix B, Figure 6. Again, this closely follows the odometry's estimated path. Whilst this is expected, the true path of the robot as estimated by the SLAM algorithm is much different, especially on the second lap.

By implementing the sensor model to update the weightings of the particles to ensure the positioning was accurate with reference to the beacons located around the path. From the result of the sensor models implementation in Appendix B, Figure 7, it is observed that the robots position estimate is often corrected towards the SLAM estimation of the robot's position when the robot gets close to a beacon. This confirms that the sensor model is working as expected, but without the perfect weightings to implement the correct adjustment every time. Thus, during the estimation of the second lap, the position is very similar to the odometry estimation of the position which is less than desirable. To improve this requires adjustment of both the particle estimation errors in the motion model and the adjustment of the PDF mean and variances for the sensor model. The confirmation that the first lap is much more accurate than the second lap, the errors between the estimated position and the SLAM estimate position is plotted in Appendix B, Figure 8. It can be observed that around halfway through the iterations the error becomes more erratic in both directions.

Part C

The map obtained from the gmapping is shown in Appendix C, Figure 9. The map was an occupancy map which represented the environment as a collection of cells. Each cell was a binary random variable with two states: occupied or free. White represented a free cell, black was an occupied cell, grey was unsure since the robot sensors didn't reach those cells.

The benefits of occupancy grids were that they were simple to implement and view. The ease of viewing is evident in the outputted map of Figure 9 as there are clearly obstacles (the chairs and desks of the environment) which are clearly shown in the map. Another benefit was that occupancy grids provide a dense representation of the environment. By having a dense representation, the accuracy of the determined obstacles and free areas was improved.

From the outputted map the occupancy grid provides an explicit representation of the environment in terms of both occupied and free space. Thus, the created map can be used for path planning. The rectangular grid of the map was suitable for the environment since the walls are of a rectangular shape as well. The errors crept in with regards to the chairs and desks which are non-rectangular elements of the environment thus giving discrepancies and potential errors.

The grid resolution selected for the gmapping algorithm was a trade-off with computational complexity. The higher (more refined) the resolution the more computationally complex the algorithm is. Thus, from the given output the resolution can be deemed as reasonable based on the suitability of the map.

Loop-closure is difficult for grid-based maps. This phenomenon was experienced by the difficulty in returning the robot to its starting spot as the robot would overshoot its initial starting point when trying to close the loop.

Appendix A: Sensor Fusion Outputs

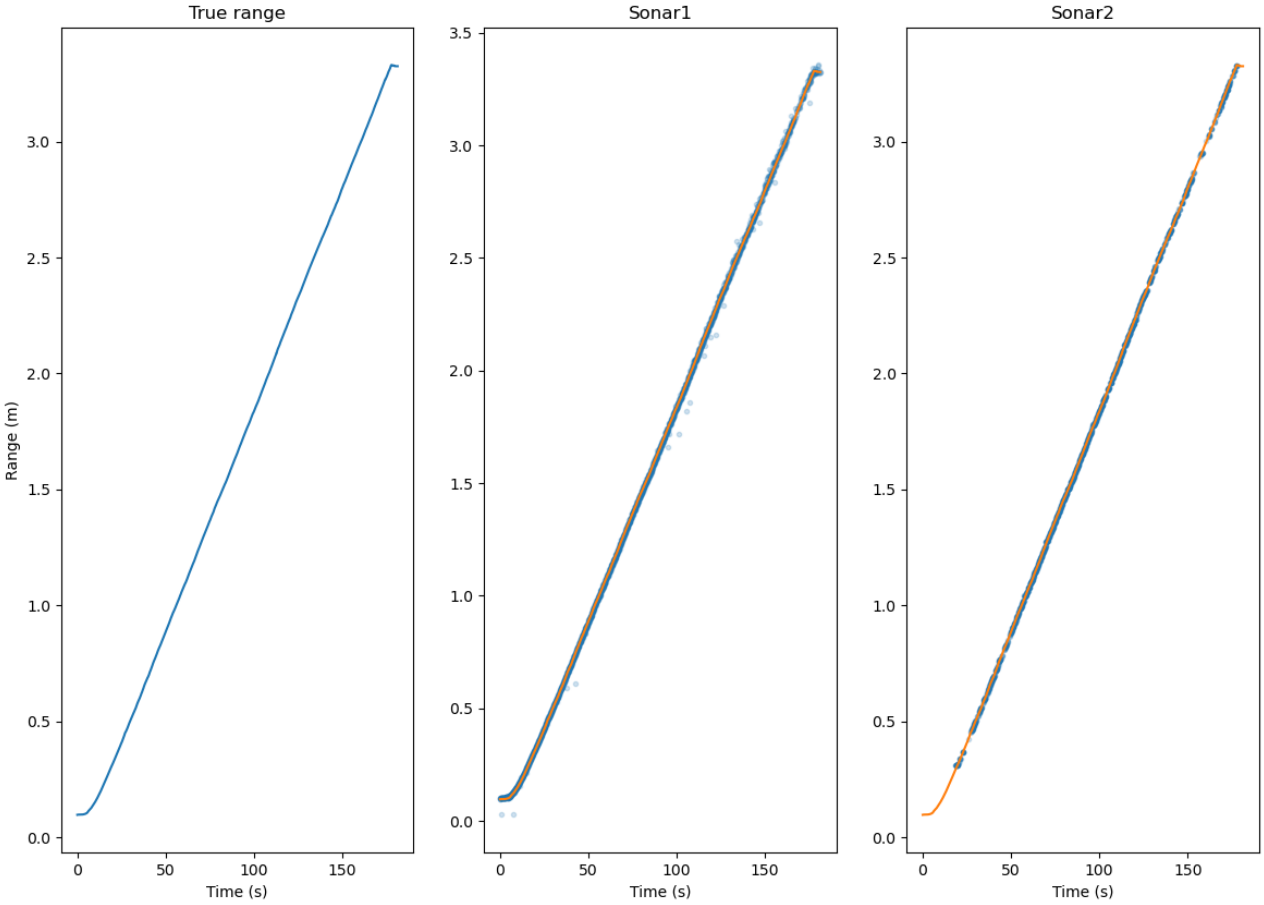

Figure 1: Sonar1 and sonar2 sensor models compared against the true range.

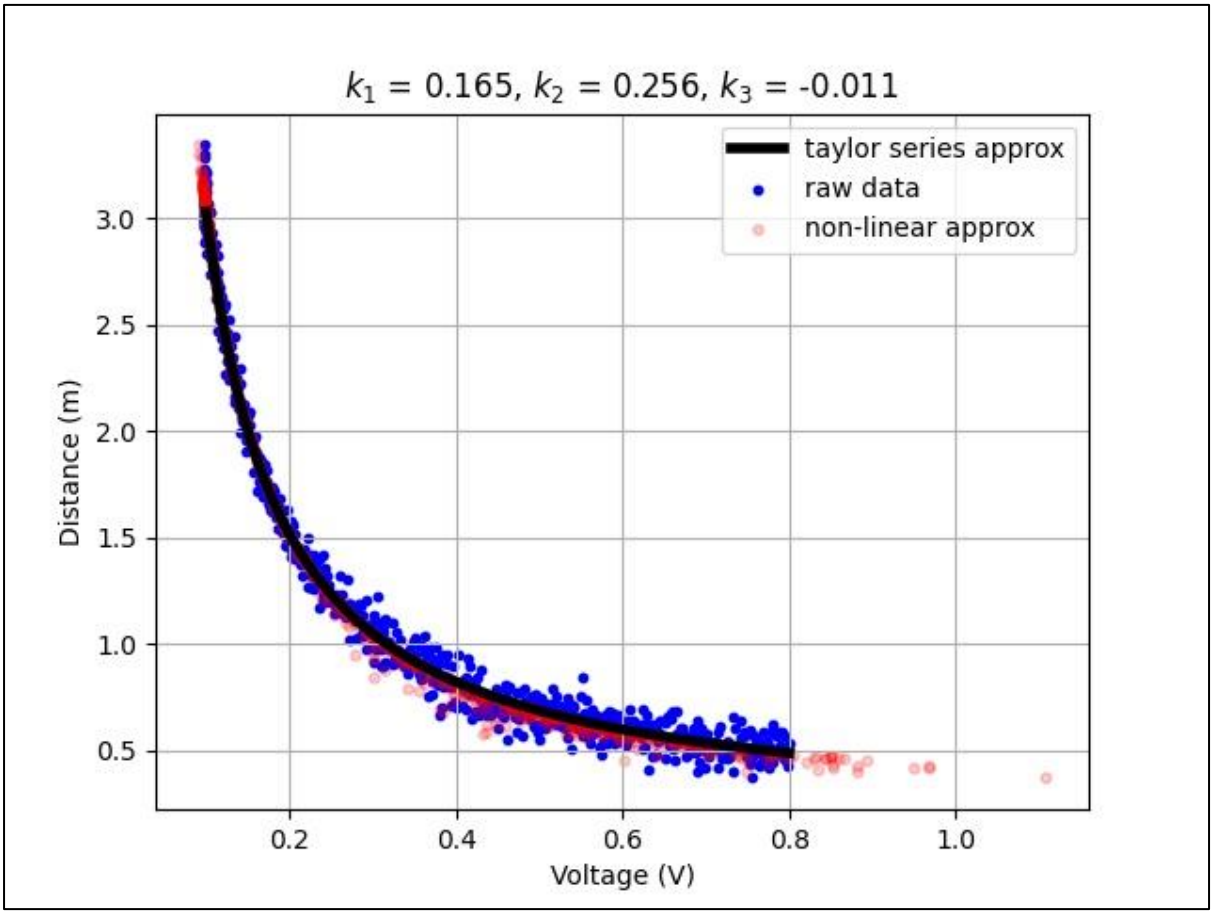

Figure 2: The IR sensor model compared against the raw data.

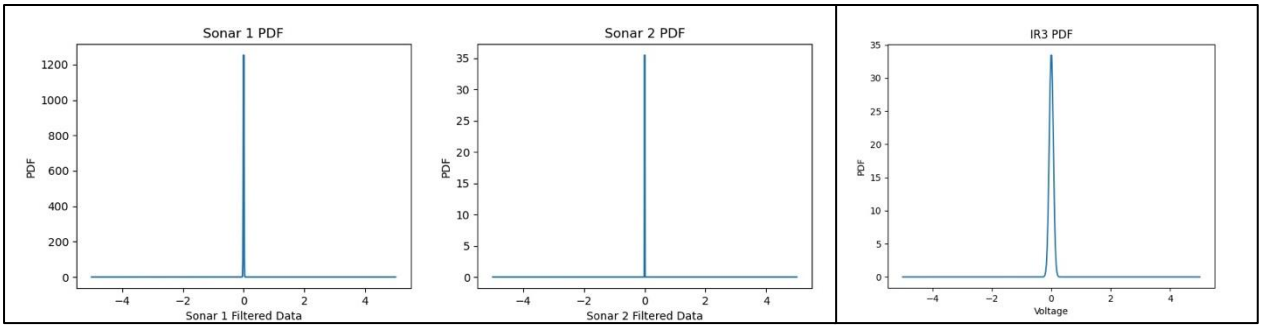

Figure 3: The probability density function for filtered sonar1, sonar2 and unfiltered IR3

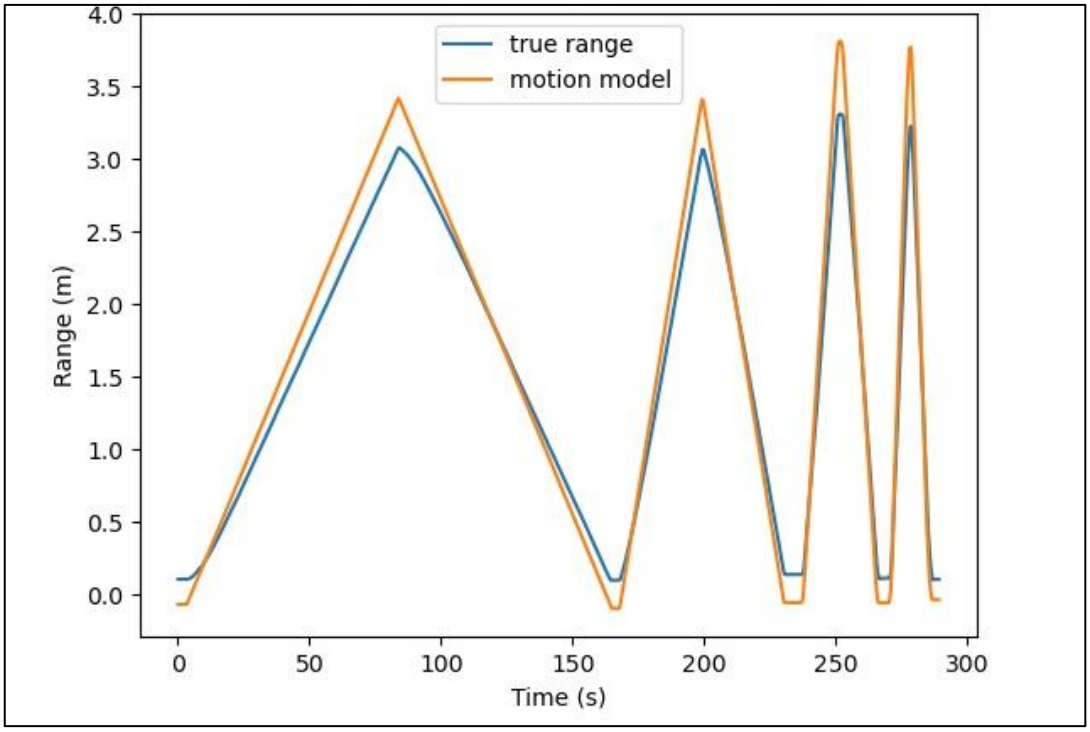

Figure 4: The motion model of the range compared to the true range.

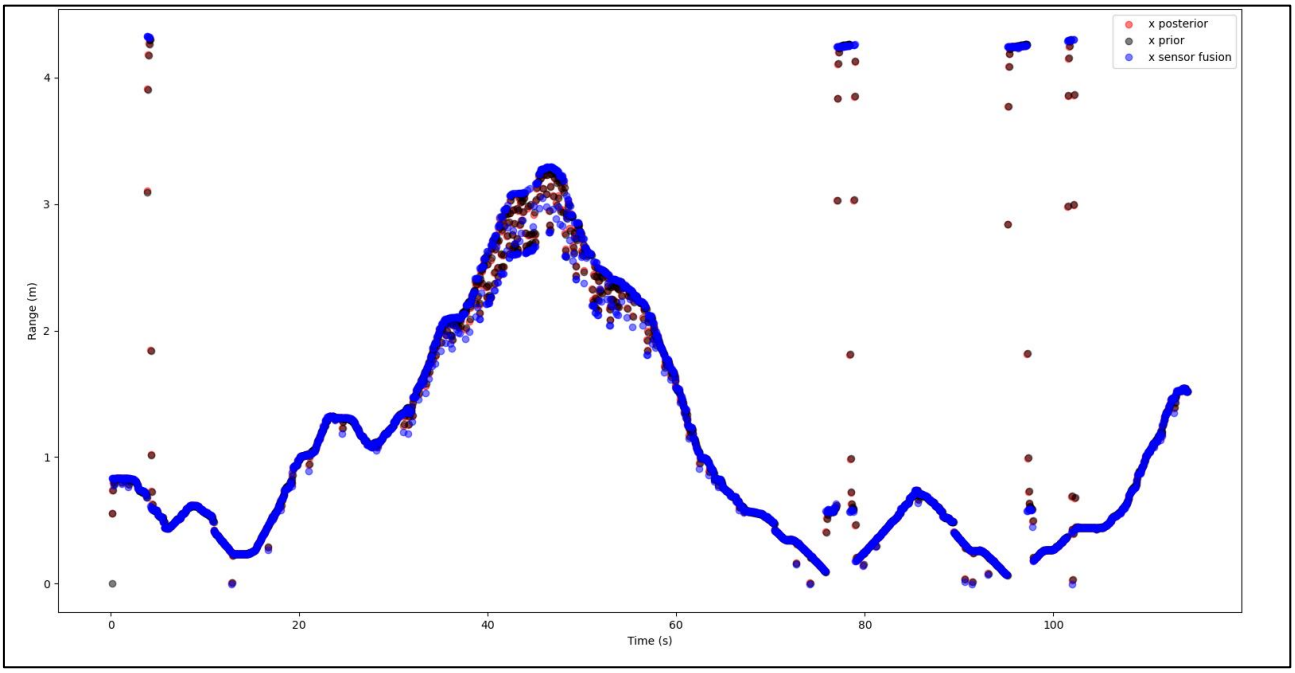

Figure 2: Kalman estimation of the test case.

Appendix B: Particle Filter Outputs

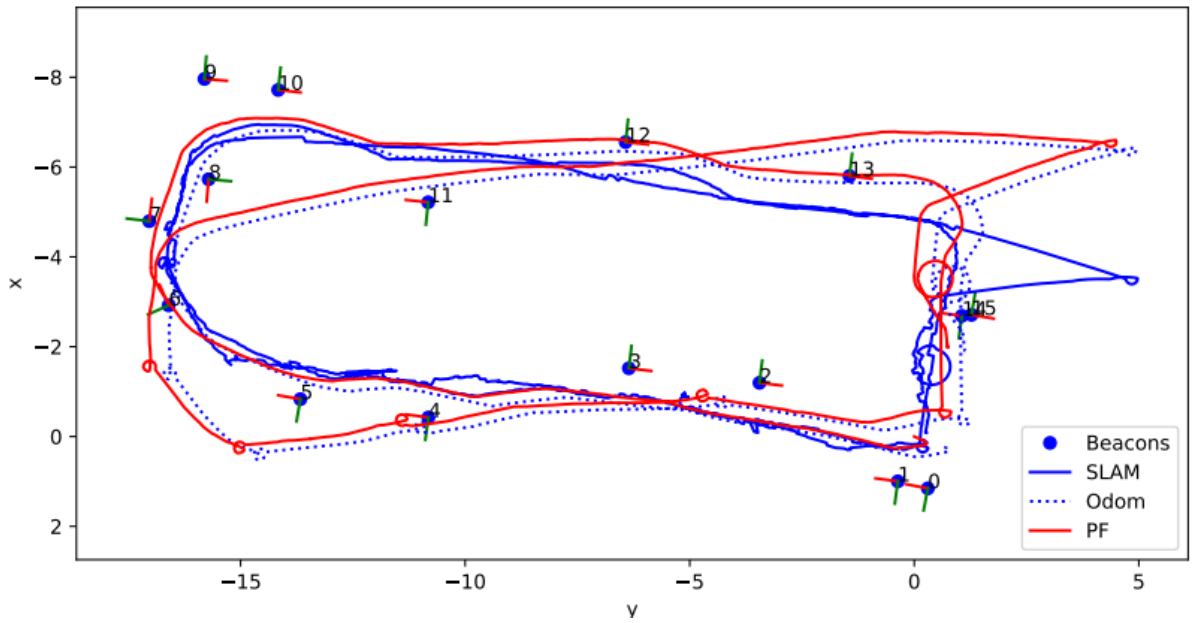

Figure 3: The estimation from only the motion model.

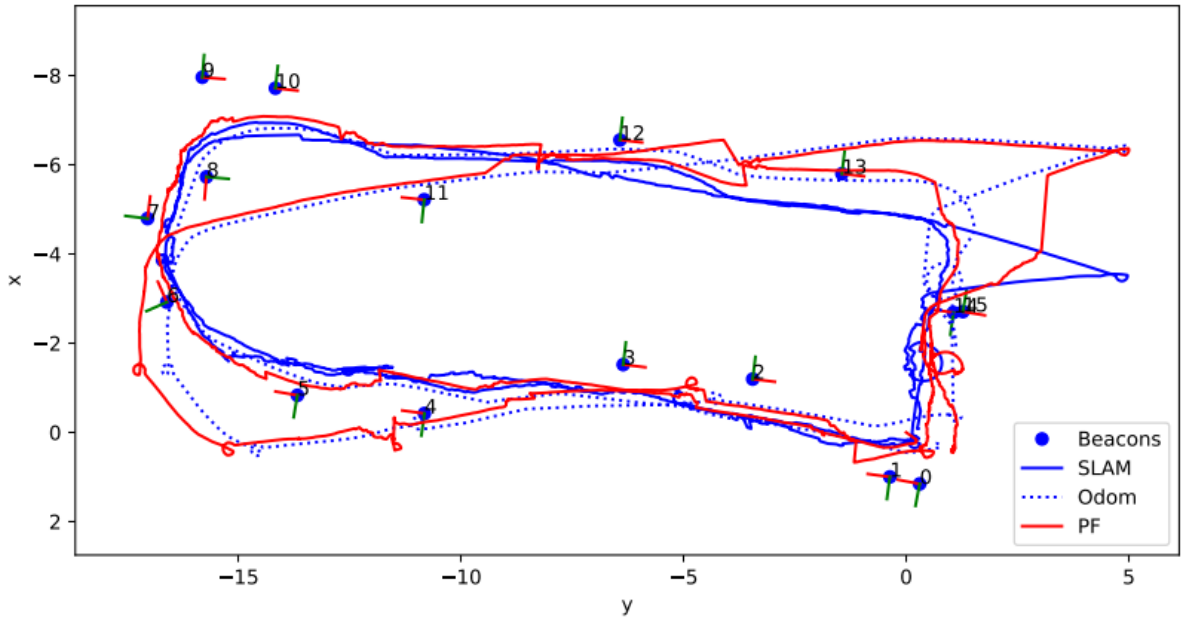

Figure 4: Resulting plot from the particle filter model implemented where the red line is the estimated route of the Turtlebot.

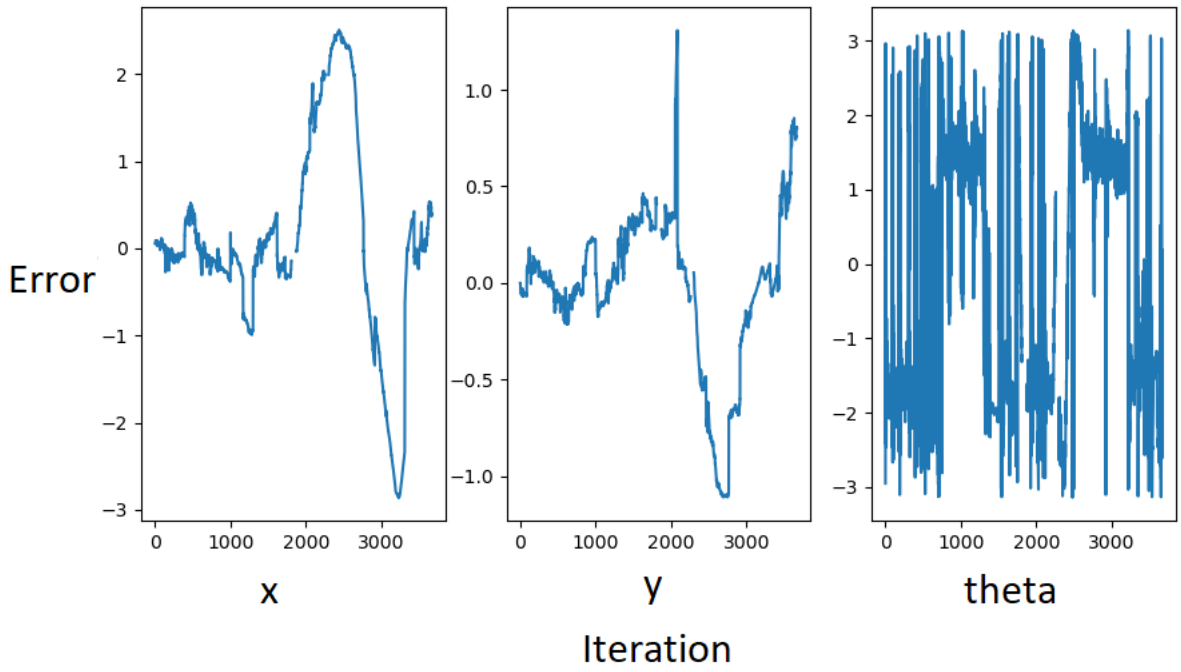

Figure 5: The corresponding error plots between the estimated route and the route estimated by the SLAM algorithm.

Appendix C: Gmapping Map Output

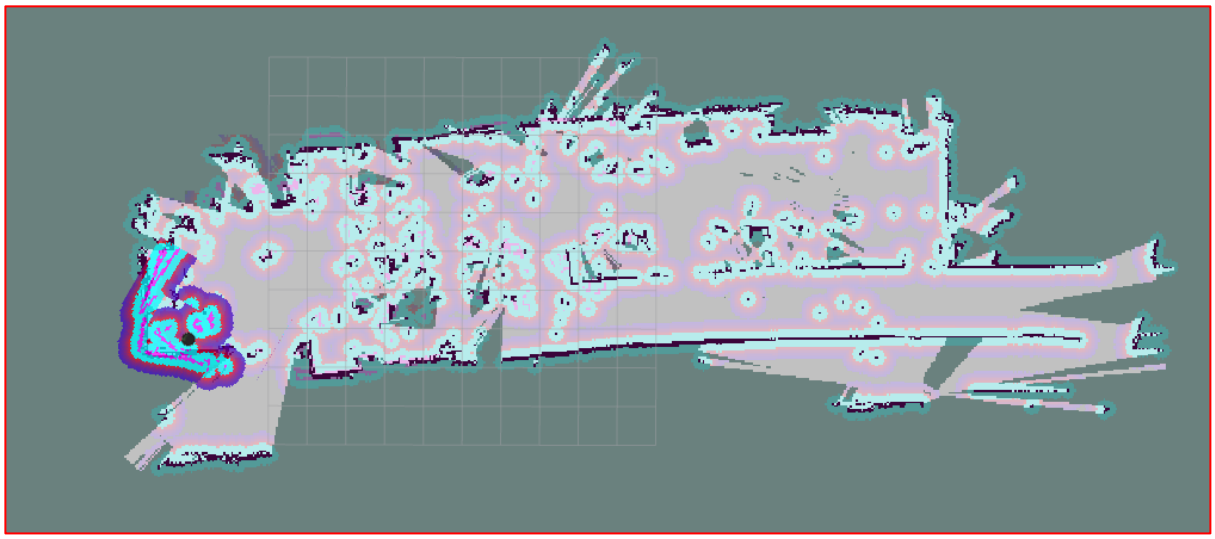

Figure 6: Map obtained from the gmapping output.