Humanoid Robot-SLAM with Particle Filter and Occupancy Grid Mapping

Aditya Kulkarni
A53322526
ECE 276A
UCSD
adkulkar@eng.ucsd.edu

Abstract

The problem of estimating the robot's location and the map of the surrounding for a robot is considered here. This problem is popularly known as SLAM and is a very important problem solved and analysed in robotics. Here we perform SLAM to a humanoid ground robot with three degrees of freedom. With the sensory data collected from LIDAR, RGBD and Odometric readings. We propose to use a particle filter to perform localization of the robot and use occupancy grid mapping to map the environment. A total of 5 data set was used to perform experiments to analyse the effect of noise and hyper parameters on the performance of the SLAM. Result??

1 Introduction

In any robotics application it is very crucial to know the state of the robot and also the environment. The state of the environment is well known in some situations either by using pre-known map of the surroundings or using systems like GPS. But robots become more important and relevant to map in mapping completely unknown terrain because of the dangers that the environment might present. Examples include interplanetary motions and underground and underwater missions, where there is risk for human life it is best to use a robot that can know its current location and also the environment. We try to solve a similar problem for a humanoid robot equipped with LIDAR (Light Detection and Ranging) and Odometry to measure the relative poses of the robot. SLAM solves this problem by assuming each sensor readings and odometry readings have some unreliability and some certainty. To get best of both it combines the two measurements to decrease the overall uncertainty. This is done by a step by step approach of sensing, predicting, updating, and mapping. This particular combination helps to improve the performance of the system significantly. In this paper we will discuss the idea behind each of these steps, their significance to SLAM. Also we will discuss about parameter tuning for SLAM system.

2 Problem Formulation

The problem of SLAM can be modeled as that of parameter estimation. Where the parameters are the state x of the robot and the map m. And then we try to maximize the joint probability distribution of the state, observation and controls. It is common to assume markov assumptions for robotic systems that makes the equations pretty simple for analysis. The assumption is that the current state of the robot depends only on its previous state and controls it sees. This sim-plies the joint probability distribution into product of terms over time (1). Where p_f is the motion model, p_h is the observation model for the robot and u is the control input to the robot.

$$p(x_{0:T}, m, z_{0:T}, u_{0:T-1}) = p_{0|0}(x_0, m) \prod_{t=0}^{T} p_h(z_t|x_t, m) \prod_{t=1}^{T} p_f(x_t|x_{t-1}, u_{t-1}) \tag{1}$$

Here we estimate the parameters using an approach of MLE because we have not sufficient prior information of the robot and also because it is simpler computational complexity. Then the problem becomes

$$max \sum_{t=0}^{T} log(p_h(z_t|x_t, m)) \sum_{t=1}^{T} log(p_f(x_t|x_{t-1}, u_{t-1})) \tag{2}$$

In the next section we will reduce the above equations to that of humanoid robot and use the approach of particle filter to get the state estimates. Additionally we are also interested in the map, for this we intend to build an occupancy grid map. Here we divide the map into smaller cells or grids each indicating a particular position in the real world. Since each cell can be occupied or unoccupied we will consider each cell as a random variable m_i and make an assumption that each cell is independent on each other. This is done to simplify the computations. We then try to find the probability of a cell being occupied and then estimate the over all map (2).

$$m_i = \gamma_{i,t} \tag{3}$$

We use particle filter approach to find the state and update the map based on the best particle's state. Robot state information is got by using observatory model based on LIDAR and motion model based on Odometry more on this in the following section. And since our robot is a rigid body we need to account for the pose difference in robot's centre of mass and LIDAR position with respect to the body. This is done by using transformation matrices using homogeneous equations which accounts for both rotation and translations (4)

$${}_wT_l =_w T_b *_b T_h *_h T_l \tag{4}$$

Where ${}_aT_b$ is a transformation matrix from frame b to a and w, l, b, h are world,lidar, body, and head frame of reference respectively. And we assume that the robot starts at origin. And the estimate of the map happens with respect to origin we translate all measurement to the initial robot's position using (4) i.e. world frame.

3 Implementation Details

The overall implementation can be broken into three sections transformation of measurements to world frame and apply particle filter to get an estimate for pose for the robot and update the occupancy grid map based on the estimate best particle. By pose or state of the robot we mean it's $[x, y, \theta]$ in the world frame. Let look at each steps in detail.

3.1 Pose Equation

The LIDAR sensor is mounted on the head of the robot and all the measurements need to be translated to the world frame for localization and mapping purpose. We achieve this by calculating the pose matrices as we know the dimensions of the robot and the position of LIDAR on the robot. The transformation can be described by three equations as stated in previous section. First we translate the LIDAR readings to head which is just a translation transform. And then translate it to the body of the robot which has head angles in terms of yaw and pitch along with translation. From body we again transform the readings to world frame which includes the three degrees of freedom of the robot x, y, θ. The individual transforms are considered to be a motion in a more general SE3 settings even though some motions are just SE2 for the sake of generalization. The equations for transforms are listed below. And we also convert the LIDAR scan to Cartesian coordinates from polar coordinates. All these values are obtained from the robot specification sheet.

$$_hT_l = \begin{pmatrix} I & _hp_l \\ 0^T & 1 \end{pmatrix}$$

Where $p_l$is the translation from the head to lidar. And the total rotation from head to body has matrices for yaw(ψ) and pitch(ϕ) multiplied together as mentioned below

$$_bR_h(\psi) = \begin{pmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$_bR_h(\phi) = \begin{pmatrix} cos(\phi) & 0 & sin(\phi) \\ 0 & 1 & 0 \\ -sin(\phi) & 0 & cos(\phi) \end{pmatrix}$$

$$_bT_h = \begin{pmatrix} R_\phi R_\psi & _bp_h \\ 0^T & 1 \end{pmatrix}$$

$$_wR_b(\theta) = \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$_wT_b = \begin{pmatrix} _wR_b(\theta) & _wp_b \\ 0^T & 1 \end{pmatrix}$$

$$_wT_l =_w T_b *_b T_h *_h T_l \tag{5}$$

The above mentioned transforms are done for every LIDAR scan. Now we have all the measurements with respect to world frame we can start with the process of particle pose estimation and mapping step.

3.2 Occupancy Grid Mapping

We represent the environment in which the robot is present using occupancy grid mapping. Here the total environment is broken down into grids. And we try to assign a probability to each of the cells representing the chance of it being occupied. As mentioned in the previous we try to find the probability a particular cell being occupied and then by estimating this for all the cells in the grid we get an estimate for the over all map. Detecting an obstacle at a point on the map by LIDAR will result in increasing the probability of finding an obstacle there. And similarly an empty space predicted by LIDAR will result in reduction of probability at that cell. However to simplify the calculation and to avoid computing extra normalizing term for the probability we compute the ratio of the probabilities and interestingly this ratio turns out to be dependent on the type of sensor and if the sensor predicts obstacle or free space. Therefore the equation of update for each cell is given by

$$\lambda_{i,t+1} = \lambda_{i,t} + \sum_{t=0}^{t+1} \log g_h(z_t|m_i, \mu_t) \tag{6}$$

$$\lambda_{i,t+1} = \lambda_{i,t} + \Delta\lambda_{i,t} \tag{7}$$

The second term is called the measurement trust. That is how much do we trust the sensor readings. And for the LIDAR scan the value is typically around $\log 9$. This basically means that the ratio of probability of finding an obstacle given LIDAR detects to probability of not finding an obstacle given LIDAR detects an obstacle. In terms of probability this can be written as

$$\log g_h(z_t|m_i, \mu_t) = \frac{p(m_i = 1|\hat{m_i} = 1)}{p(m_i = -1|\hat{m_i} = 1)} \tag{8}$$

So for every detection of obstacle we add a term of $\log 9$ to the corresponding cell in occupancy grid. Similarly we subtract $\log 9$ for every free space detected by the sensor. And then to get a discrete map for the environment we threshold these values and classify into free or occupied. Since we use particle filter to perform state estimation of the robot we choose the particle that has best correlation with the map to perform update.

3.3 Particle Filtering

We implement particle filter to do the state estimation for the robot. Particle filter tries to find the best possible estimate for the pose given the observations and the motion models by combining them together and reducing the over all uncertainty. We use a delta mixture to represent the pdf of robot state at time t. And thus the equation of the state will be given by

$$p_{t|t}(x_t) = p(x_t|z_{0:t}, u_{0:t-1}) = \sum_{k=1}^{n} \alpha_{t|t}^{(k)} \delta(x_t : \mu_{t|t}^{(k)}) \tag{9}$$

Where $\mu^{(k)}$ is the hypothesise for each of the particle. And α is the confidence in that particle and z and u are the observation and control respectively. Motion model describes how the pose of the robot changes based on the current state and control inputs. Here the motion model here is based on the odometry data and the model can be represented as

$$\mu_{t+1|t}^{(k)} = f(\mu_{t|t}^{(k)}, \epsilon_t) \tag{10}$$

$$\mu_{t+1|t}^{(k)} = \sum_{t=0}^{t} \delta_{t|t} + \delta_{t+1} + \epsilon_t \tag{11}$$

where ϵ_t is the 3-Gaussian noise at time t, $\mu_{t+1|t}$ is the prediction of state, $\mu_{t|t}$ is the current state and δ is the relative change in pose observed. So here the motion model is a simple accumulator of previous changes in pose. And we assume that the initial change in pose in $[0, 0, 0]$. The observation model is derived from the LIDAR scan. Our observation should indicate how well is our understanding of the map with respect to the observed sensor readings. These can be done considering correlation of the observed scan with the previous estimate of map. And since we assume that the robot cannot vanish and appear somewhere else this assumption holds true. So the observation model is described as

$$p_h(z_{t+1}|\mu_{t+1|t}^{(k)}, m_t) \propto exp(corr(y_{t+1}, m_t)) \tag{12}$$

$$p_h(z_{t+1}|\mu_{t+1|t}, m_t) = softmax(corr(y_{t+1}, m_t)) \tag{13}$$

Where p_h is the observation model, m is the map based on prior estimates. And $corr$ is the correlation function which determines the similarity between previous map and the estimate of map y. The best way to remove the proportionality is to normalize and we recognize this function is now softmax. Estimation of the map is done by thresholding the log-odds of the map obtained as mentioned in the Occupancy Grid section. But it was found that it is insufficient to rely just on one particle's (even though best particle) to make the prediction so we implement a mean shifted threshold to do the map classification. So we assign a cell to free or occupied only if 8 or more best particles classify it as a free or occupied. This was a hyper parameter and using a value of 8 does a good job for 20 particles filter. This in some way makes the model forget insignificant results of the past hence makes the map finer as in figure 2 and 1.

$$corr = \sum_{i=0}^{N} 1(y = m) \tag{14}$$

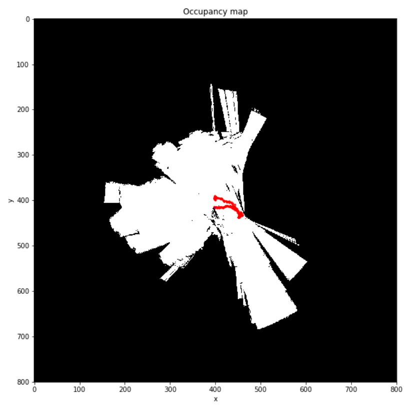

Figure 1: Train Data 0 without Log-Odds thresholding

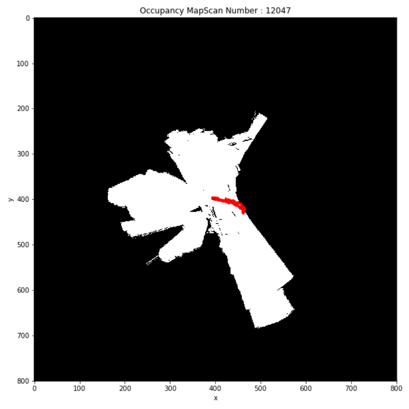

Figure 2: Train Data 0 with Log-Odds thresholding

And since we perform softmax operation and update the weights there is high chance that we may end up with single particle and hence our model becomes less exploratory in nature. To avoid this we use the technique of re-sampling. In this approach we check for the number of effective particles in the model by computing

$$N_{eff} = \frac{1}{\sum_{i=1}^{n} w_i^2} \tag{15}$$

Then we decide what is minimum number of particles we want the system to always want to have call it N_{th} and we perform re-sampling whenever we observe that the number of particles falls below this. Re-sampling here is done by choosing n number of particles from the distribution whose distribution is the weights of the particles. Thus we end up replacing a particle with high probability with many similar particles thus increasing our chance our exploring better.

3.4 Functional Flow and Practical Considerations

Here we describe the steps that we perform the SLAM. We first determine the size of the map using the technique of dead reckoning. Here we set the number of particles to 1 and add no noise to the system and make it move around and the limits of the path and thus choose the grid map size needed. The images for dead reckoning for all the data sets are shown in figures 3 4 5 9 6. Then by experimenting it was found that 20 particles do a fair job of mapping and estimation. Then we first perform an initial scan and update the log-odds map. Then we perform the predict step for each of particles. This gives rise to multiple poses for the robots position. And then from each of these we compute the correlation of the LIDAR scans with our prior estimate of the map. This is done as it the robot was to be present at these locations and received the scan from LIDAR. Then we see which particle had the most correlation with the map to decide the best particle and use its pose to update the map log odds. Choosing of the best particle can also be looked upon as that of having a softmax function. These steps are repeated until all the laser are completed.

There were some practical problems encountered during the SLAM. Firstly, The head of the robot can pitch causing the LIDAR to detect ground as wall. But this is wrong classification and this can distort our map. To avoid this we add a filter that will allow points that are above the ground level. This is done by tracking the z component of the LIDAR scan when projected to body frame and only values that are positive are considered. Negative values correspond to robot looking down at the floor. Secondly, the odometry sensor readings and the LIDAR readings are out of sync hence we try to get a matching value of odometry value for a LIDAR reading using the timestamps provided.

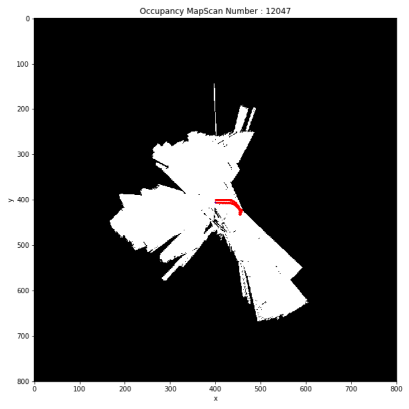

Figure 3: Data 0 Dead Reckoning, Red is Particle path and Blue is Start Point

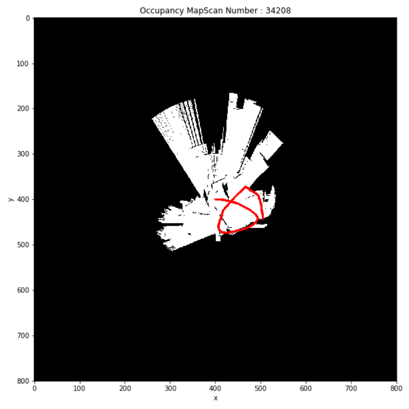

Figure 4: Data 1 Dead Reckoning, Red is Particle path and Blue is Start Point

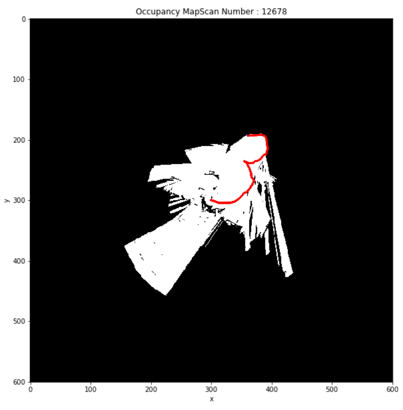

Figure 5: Data 2 Dead Reckoning, Red is Particle path and Blue is Start Point

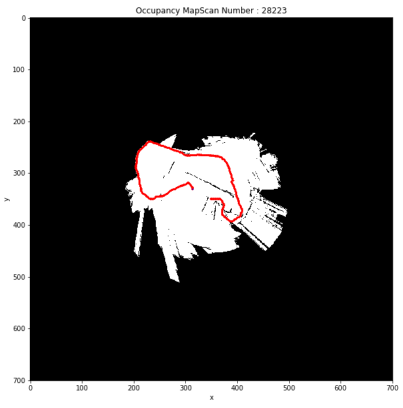

Figure 6: Data 4 Dead Reckoning, Red is Particle path and Blue is Start Point

Next we will discuss the hyper parameter tuning for the model. Main parameters to be considered were re-sampling threshold, number of particles, LIDAR sensor confidence score, occupancy grid map thresholding, x,y,θ noise. Let's consider the effect of these things. First for all the data sets we observed no significant changes between number of particles as long as number of particles was greater than 10. If the number of particles is very small then we do not get the best possible trajectory. And since LIDAR is supposed to be very accurate sensor we choose a confidence score ratio of 9 for it. Re-sampling threshold number was chosen to be 50% of total number of particles. This number was chosen because 50% of 20 particles is 10 and this means we will have at least ten unique particles hence making it possible to scan significant number of possible options in the x-y plane of motion for the robot. This equals at least 2 particles per quadrant. Occupancy grid thresholding was set to 8, meaning only if a particular cell was classified as occupied or free more than 8 times. And additionally we capped the log-odds sum to 100 that is any increment beyond that will not increase to log-odds further. Similar lower bound was set. Noise addition to the pose of the robot was found to be very sensitive and seemed to have a significant effect on the performance. Hence trying out random values seemed to be the only apparent solution to the problem so we try to propose an alternate approach to find the estimate for the noise. So we try to estimate the deviation of the δ poses. This indicates the deviation of movements the robot can have. So we add a noise that is equivalent to twice the number to take into consideration the robot can move in either direction for each pose values ensuring we have a good spread of estimate for the particles.

4 Results

Below are the tables containing all the hyper parameters chosen to run each of the data sets. Black region represents unmapped terrain and white region represents empty space. Some interesting findings are listed here. For data set 0 no significant change was observed after increasing the number of particles to 50. This can be seen from the figure 7 8. And we can find the parameters used for data set 0 in table 1. Next we consider data set 3, we can compare the final result obtained from SLAM with dead reckoning and we observe that proper correction in particle pose is seen. This can be visually verified using the RBG images. The difference can be spotted in the final MAP generated figures 13 and 9. Similarly the final map for data set 1,2 and 4 listed below. The model did struggle when there were large variation in the yaw of the robot which causes the particles to drift off. Trying a higher number of particles could help solve these problems as we can explore the terrain better.

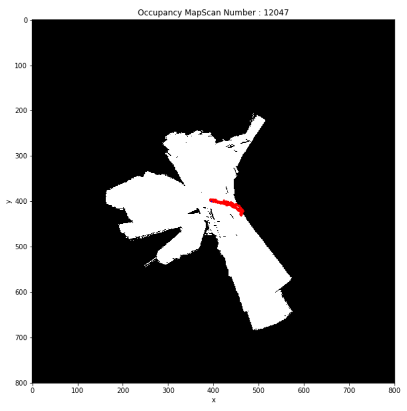

Figure 7: Data Set 0 Number of Particles 20, Red is Particle path and Blue is Start Point

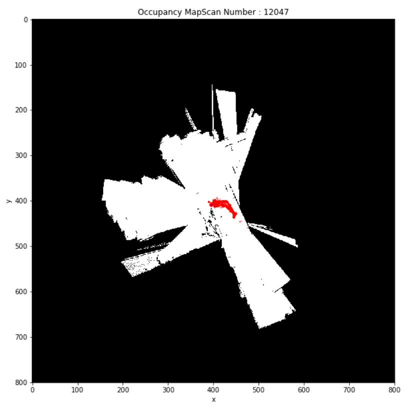

Figure 8: Data Set 0 Number of Particles 50, Red is Particle path and Blue is Start Point

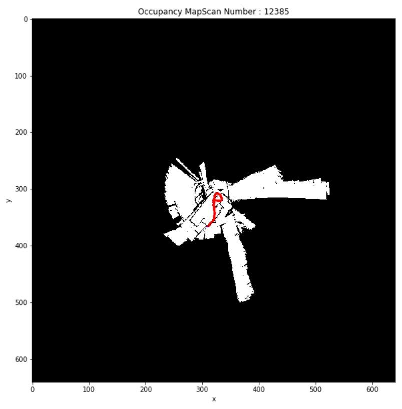

Figure 9: Data 3 Dead Reckoning, Red is Particle path and Blue is Start Point

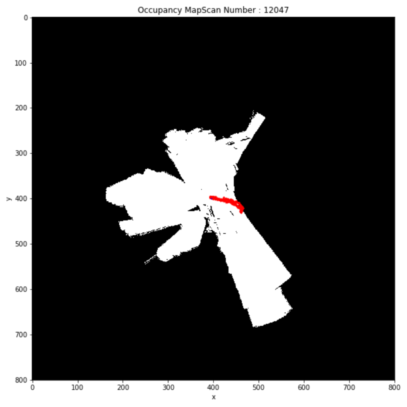

Figure 10: Data Set 0 Complete Map, Red is Particle path and Blue is Start Point

The yaw of the robot is represented by the plots shown below. Where x-axis is the linear time and y axis is the best particles yaw based on various scans. We can observe how the robot turns as it moves around in the terrain as shown in the figure 15 16 17 18.

Parameter	**Value**
Map Size	40x40 m^2
Map Size Resolution	0.05m
Number of Particles	20
Lidar Confidence	9
Resampling Threshold	50%
Noise Standard Deviation	0.003514,0.00398,0.00218 (m)
Noise Mean	0,0,0
Code Run Time	1471.66s

Table 1: Hyper Parameters for Data Set 0

Figure 11: Data Set 1 Complete Map, Red is Particle path and Blue is Start Point

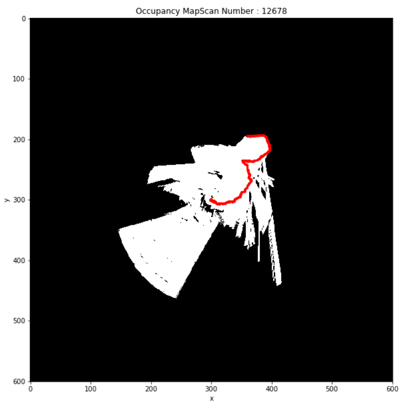

Figure 12: Data Set 2 Complete Map, Red is Particle path and Blue is Start Point

5 Conclusion

In this paper we demonstrated implementation of SLAM for a humanoid robot equipped with LIDAR sensor and Odometry. We modeled a particle filter to estimate the pose of the robot and used occupancy grid to represent the environment in which the robot was present. Implementation of SLAM involves tuning on many parameters which were discussed and how each can impact the performance of the system was also studied. In general it was observed that the model is very sensitive to yaw estimates received from the odometry. And particle filter does a pretty job in estimating the trajectory of the particle and similarly occupancy grid does a decent job for some data sets however we found that there is a need to play around with more hyper-parameters and run more trials to get better results. And to run more cases, machine time is a main concern so an approach to make the estimation of hyper parameters a deterministic of the observations made was tried here. And some promising results were observed. There is still room for trying out different statistic of the observation to make more accurate estimation of the parameters.

PS: The hyper-parameter files and all the intermediate step images are attached in the file. There is also a video attached for data set 0 and 3.

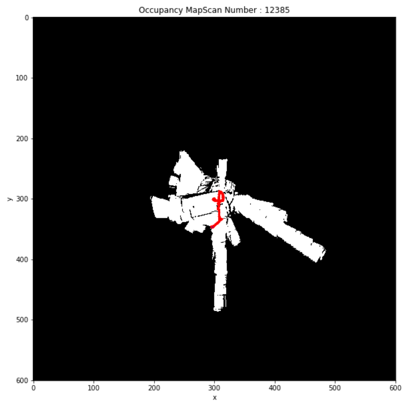

Figure 13: Data Set 3 Complete Map, Red is Particle path and Blue is Start Point

Figure 14: Data Set 4 Complete Map, Red is Particle path and Blue is Start Point

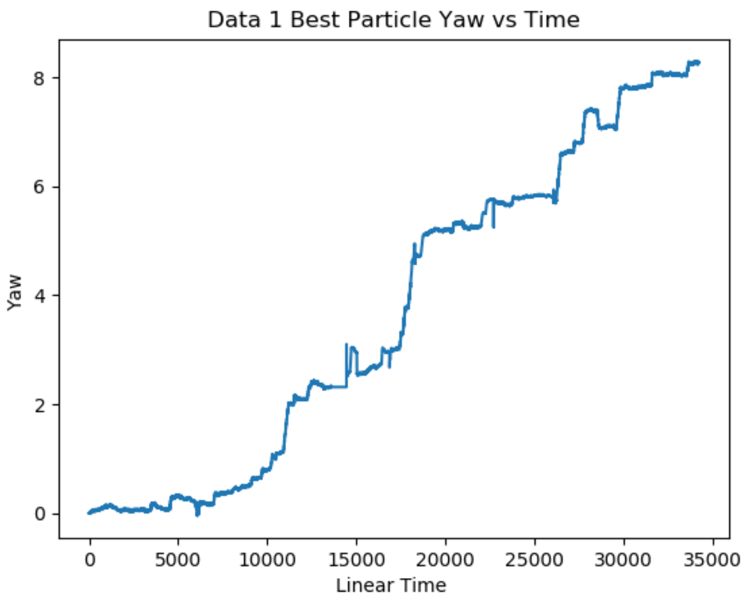

Figure 15: Data Set 1 Yaw vs Time

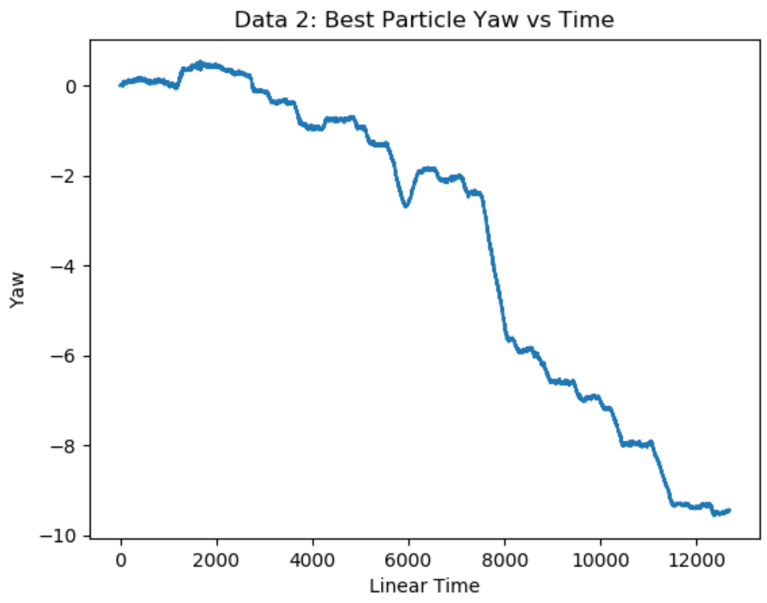

Figure 16: Data Set 2 Yaw vs Time

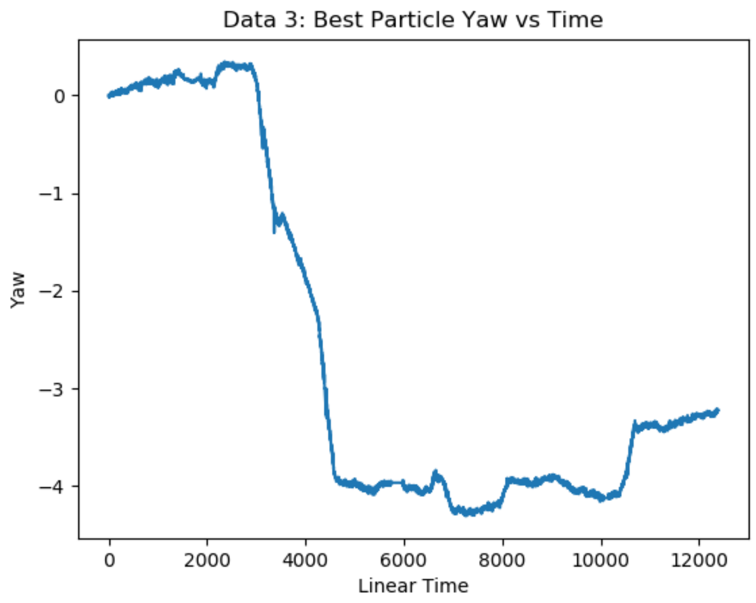

Figure 17: Data Set 3 Yaw vs Time

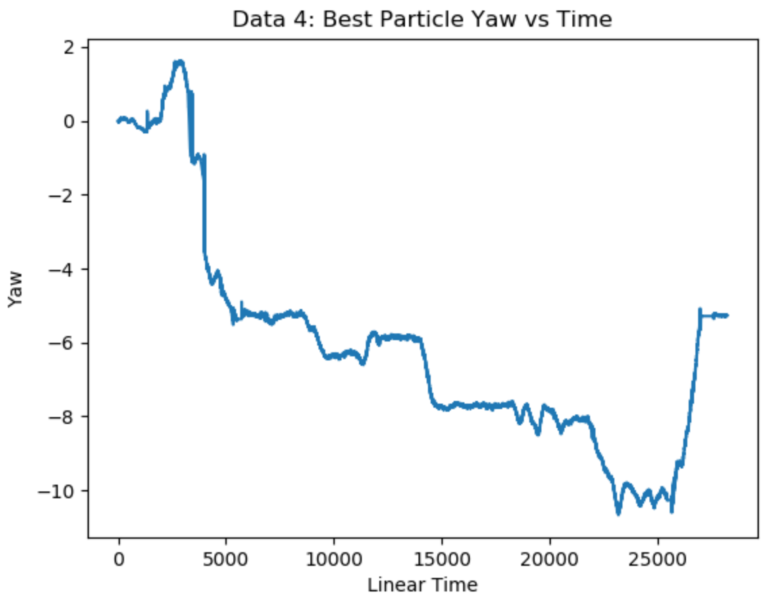

Figure 18: Data Set 4 Yaw vs Time

Parameter	**Value**
Map Size	30x30 m^2
Map Size Resolution	0.1m
Number of Particles	20
Lidar Confidence	9
Resampling Threshold	50%
Noise Standard Deviation	0.0043972,0.0031219,0.00254868 (m)
Noise Mean	0,0,0
Code Run Time	1378.952s

Table 2: Hyper Parameters for Data Set 3