Implementation of Particle Filter SLAM using Odometry, 2-D LiDAR Scans and Stereo Camera Measurements

Sanchit Gupta
Department of Electrical and
Computer Engineering
University of California San Diego
sag006@ucsd.edu

***Abstract*—This report discusses about the methodology to implement Particle Filter SLAM using the odometry data from encoders and Fiber Optic Gyroscope (FOG), 2-D LiDAR scans and stereo camera measurements. The report further discusses about the localization of the robot and development of a 2-D occupancy grid map of the environment using the data from the sensors. The results of the path followed by the robot in dead reckoning state and after implementation of the particle filter are then presented in this report.**

Keywords—Particle Filter SLAM, Dead reckoning, Robot Localization, encoder, FOG, LiDAR

I. Introduction

Simultaneous Localization and Mapping (SLAM) is a popular and widely researched topic in the field of robotics, autonomous driving and augmented reality. SLAM is a technique through which the robot estimates its state in the environment it is traversing and simultaneously, constructs the map of the environment according to the feedback received from the sensors. This technique aids the robot in identifying free spaces for locomotion in the unknown environment. The map further helps the robot in avoiding the occupied spaces in the environment during its motion.

The feedback from the sensors plays an important role in the implementation of SLAM algorithms. In this project, the odometry data comprising of distance travelled and orientation, obtained from encoders and FOG respectively, help in estimating the trajectory of the vehicle. The differential-drive motion model takes input from these sensors to calculate the predicted step. Further, the 2-D scan data from the LiDAR over a field of view (FOV) of 190 degrees help in building the map of the environment in which the vehicle is travelling as well as aids in correcting the trajectory of the robot. This provides the observation model through which the updated step is calculated.

The particle filter method is implemented in this project, which uses a set of particles to represent the vehicle's state. The particle representing the most accurate state of the vehicle according to the map of the environment is considered to localize the vehicle. This cycle continues for all the data points of the sensors and the path covered by the vehicle. The mapping is performed using log-odds occupancy grid map. It is updated for every available scan data from the LiDAR. The steps followed to perform localization and mapping are discussed in detail in this report. The results for the dead reckoning state of the vehicle, trajectory of the vehicle obtained from particle filter method along-with the occupancy grid map and binary map constructed for the environment using different number of particles are also presented in this report.

II. Problem Formulation

This section will discuss about the problem of localization of the car and construction of occupancy grid map of the environment in terms of mathematical formulas. The problem formulation of SLAM includes estimation of robot states $x_{0:t}$ and map states $m_{0:t}$ from the sequence of control inputs $u_{0:t}$ and observations $z_{0:t}$. The states of the robot and map, control inputs and observations are related through the joint distribution given in equation 1.

$$p(\mathbf{x}_{0:T}, \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}) = \underbrace{p(\mathbf{x}_0)}_{\text{prior}} \prod_{t=0}^{T} \underbrace{p_h(\mathbf{z}_t \mid \mathbf{x}_t)}_{\text{observation model}} \prod_{t=1}^{T} \underbrace{p_f(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})}_{\text{motion model}} \prod_{t=0}^{T-1} \underbrace{p(\mathbf{u}_t \mid \mathbf{x}_t)}_{\text{control policy}} \qquad (1)$$

The joint Probability Distribution Function (PDF) given in equation 1 is factorized using Markov assumptions, conditional probability and Bayes rule of probability.

Sensor Dataset: The outputs from encoders, FOG and 2-D LiDAR along-with timestamps for the path covered by the vehicle are provided in this project. The encoder data consists of left and right wheels counts of the vehicle with a resolution of 4096. FOG data provides with the changes in roll, pitch and yaw angles of the vehicle. The LiDAR has a FOV of 190 degrees with the rays spreading from -5 degrees to 185 degrees over an angular resolution of 0.666 degrees. The range data for all 286 rays between -5 degrees and 185 degrees is provided with the maximum range being 80 meters. The odometry data from encoder and FOG will assist in developing the motion model and the LiDAR scan data will aid in the observation model. As the update rate of FOG is nearly 10 times that of encoder, it becomes important to synchronize the outputs from both the sensors.

Motion Model: The differential-drive kinematic model is to be used to predict the trajectory of the vehicle. For the given control inputs of linear velocity v_t and angular velocity ω_t, the vehicle's location can be predicted using equation 2.

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \tau * \begin{bmatrix} v_t * \cos(\theta_t) \\ v_t * \sin(\theta_t) \\ \omega_t \end{bmatrix} \quad (2)$$

Data from encoder and FOG will help in calculating the linear and angular velocities at every instant. A Gaussian noise $\in_t$ should be added to both the velocities to increase the accuracy of prediction.

Dead Reckoning: The motion model given in equation 2 will help in constructing and visualizing the dead reckoning trajectory of the vehicle. Dead reckoning is a prediction only filter with number of particles as 1 and no noise added to the velocities.

Observation Model: The observation model will relate the robot state x_t and map state m_t with the sensor observation z_t subject to the noise $\in_t$. The relation is given in equation 3. The observation z_t has a PDF given by equation 4.

$$z_t = h(x_t, m_t, \in_t) \quad (3)$$

$$z_t \; has \; pdf \; p_h(.\,|x_t, m_t) \quad (4)$$

The scan data from LiDAR will aid in building the observation model.

Localization: Given a grid map m, sequence of control inputs $u_{0:t}$ and observations $z_{0:t}$, the vehicle's state and trajectory $x_{0:t}$ can be inferred. The particle filter prediction step is to be performed using the motion model p_f to obtain prediction PMF $P_{t+1|t}(x_{t+1})$. The update step is to be performed using the observation model p_h to obtain the updated PMF $P_{t+1|t+1}(x_{t+1})$.

Occupancy Grid Mapping: Given the vehicle's state and trajectory $x_{0:t}$ and a sequence of measurements $z_{0:t}$, the map m of the environment can be built. The occupancy-based grid map is one of the simplest representations which assigns (+1) to occupied cells and (-1) to free cells. The map model is given in equation 5.

$$m_i = \begin{cases} +1 \text{ (Occupied)} & \text{with prob. } \gamma_{i,t} := p(m_i = 1 \mid \mathbf{z}_{0:t} \\ -1 \text{ (Free)} & \text{with prob. } 1 - \gamma_{i,t} \end{cases} \quad (5)$$

The log-odds ratio for the map is defined using the equation 6.

$$\Delta\lambda_{i,t} = \log \frac{p(m_i = 1 \mid \mathbf{z}_t, \mathbf{x}_t)}{p(m_i = -1 \mid \mathbf{z}_t, \mathbf{x}_t)} = \begin{cases} +\log 4 & \text{if } \mathbf{z}_t \text{ indicates } m_i \text{ is occupied} \\ -\log 4 & \text{if } \mathbf{z}_t \text{ indicates } m_i \text{ is free} \end{cases} \quad (6)$$

Binary Map Creation: To create the binary map of the occupancy grid map, the map PMF $\gamma_{i,t}$ can be recovered from the log-odds $\lambda_{i,t}$ via the logistic sigmoid function given in equation 7.

$$\gamma_{i,t} = p(m_i = 1 \,|z_{0:t}, x_{0:t}) = \sigma(\lambda_{i,t}) = \frac{\exp(\lambda_{i,t})}{1 + \exp(\lambda_{i,t})} \quad (7)$$

Texture Mapping: Given the images from the stereo cameras, the RGB values can be projected on to the occupancy grid map as 3D vector.

III. Technical Approach

The technical approach to solve the problems of localization of the vehicle and construction of map of the environment are discussed in this section and are elaborated in different sub-sections below.

A. Transformation of LiDAR Data

The data given in lidar.csv file consists of range values of each of the 286 rays spread out by the LiDAR in each scan over varying timestamps. The range values are given in the sensor frame. The initial step is to filter the data to remove the scan points which are too close or too far from the vehicle. Hence, the data is filtered to consider only the scan points with range value falling between 2 m and 75 m. These are then converted values into x and y co-ordinates. As these co-ordinates are in sensor frame, they are converted to the body frame using pose transformation given in equation 8, with rotation and translation matrices provided in the project.

$$\begin{bmatrix} x_b \\ y_b \\ z_b \\ 1 \end{bmatrix} \quad (8)$$

$$= \begin{bmatrix} 0.001302 & 0.796097 & 0.605167 & 0.8349 \\ 0.999999 & -0.0004190 & -0.001600 & -0.01268 \\ -0.001020 & 0.605169 & -0.796097 & 1.76416 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$$

These co-ordinates in the body frame are then converted to world frame using pose transformation given in equation 9. x_t and y_t in equation 9 are obtained from encoder data and θ is obtained from FOG data. The world frame co-ordinates are finally converted to grid cells. The Bresenham2D function given in the pr2.utils file is used to obtain the grid cells between the starting and end point of each ray of LiDAR scan. An image of the first LiDAR scan is shown in Figure 1.

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & x_t \\ \sin(\theta) & \cos(\theta) & 0 & y_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_b \\ y_b \\ z_b \\ 1 \end{bmatrix} \quad (9)$$

It is further possible to filter the LiDAR data further on the basis of z_w values, however, when such filter was implemented, most of the scan points were filtered. Hence, this filter is currently not implemented in the project.

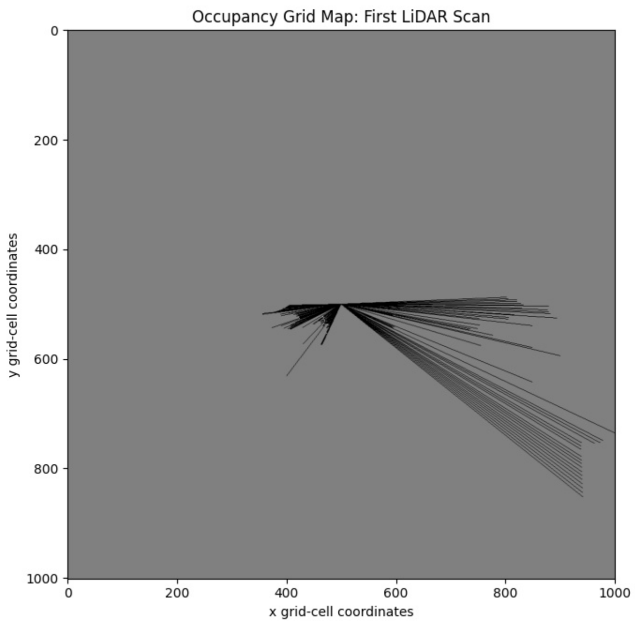


Figure 1: Occupancy Grid Map of first LiDAR Scan

B. Synchronization of Encoder & FOG Data and Dead Reckoning

As the update rate of FOG is nearly 10 times the update rate of encoder, synchronization of the data of both the sensors become necessary. It was found that the timestamp difference between two consecutive encoder data points is equal to the timestamp difference between every 11th data point of FOG. Hence, to synchronize both the data, every 11 angle values given by the FOG are summed together. This brought the FOG data to be of the same length as encoder.

Next step is to convert the encoder data in to the distance travelled by the vehicle. This is performed using equation 10.

$$Distance\ between\ two\ ticks = \frac{\pi\ x\ d\ x\ \Delta ticks}{encoder\_resolution} \quad (10)$$

In equation 10,

- d is the wheel diameter
- $\Delta ticks$ is the difference between two consecutive encoder counts
- encoder_resolution = 4096 as given

The encoders are mounted on both left and right wheels. The distance covered by each wheel between two consecutive encoder counts is calculated using equation 10. An average of both the values is taken as the distance travelled by the vehicle at the same instant. This distance value helps in calculating the linear velocity of the vehicle at every timestamp. Change in yaw angle values from the FOG help in calculating the angular velocity. These linear and angular velocities are fed in motion model given in equation 2 to predict the next step of the vehicle. This prediction-only particle filter gives the dead reckoning trajectory of the vehicle, as plotted in Figure 2.

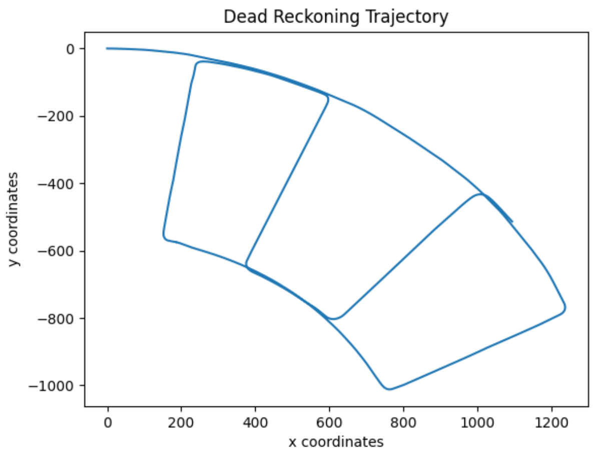


Figure 2: Dead Reckoning Trajectory of the vehicle obtained from Prediction-only Particle Filter

C. Particle Filter SLAM

The particle filter is a special case of Bayes Filter which helps to perform localization of the vehicle and construction of grid map simultaneously. It follows the joint distribution given in equation 1. As the prediction and update steps are to be computed recursively, these are given by equation 11 and 12 respectively for the particle filter.

$$p_{t+1|t}(\mathbf{x}) \approx \sum_{k=1}^{N} \alpha_{t+1|t}^{(k)} \delta\left(\mathbf{x} - \boldsymbol{\mu}_{t+1|t}^{(k)}\right) \quad (11)$$

$$p_{t+1|t+1}(\mathbf{x}) = \sum_{k=1}^{N} \left[\frac{\alpha_{t+1|t}^{(k)} p_h\left(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}_{t+1|t}^{(k)}\right)}{\sum_{j=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(j)} p_h\left(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}_{t+1|t}^{(j)}\right)} \right] \delta\left(\mathbf{x} - \boldsymbol{\mu}_{t+1|t}^{(k)}\right) \quad (12)$$

The algorithm was initially tested with a particle count of 4. The weights for each particle were initialized as 1/N, where N is the particle count. The trajectory for each particle is predicted through the prediction step and motion model given in equation 2. Linear and angular velocities are calculated from encoder and FOG data. Subsequently, update step is performed using the LiDAR data. Each particle maintains it 2D position (x, y) and orientation θ. This process works recursively to compute the trajectory of the vehicle and for construction of map.

The algorithm was tested for varying number of particles. It was observed that small number of particles, in the order of less than 10, do not improve the accuracy of the trajectory and map much. However, higher number of particles, in the order of 40 or more, show significant improvement. These results are discussed further in the Results section.

D. Addition of Noise and Update Step

After analysing the dead reckoning trajectory and initializing particles, Gaussian noise is added to the linear and angular velocities obtained from the prediction step, to improve the accuracy of the trajectory. As a starting point, a Gaussian noise of mean 0 and variance 0.5 was added to linear velocity and noise of mean 0 and variance 0.05 was added to angular velocity. Since the prediction step is performed for each particle, the noise is also added to the state of each particle.

Following the prediction step, the mapCorrelation function given in pr2_utils.py file is used to calculate the correlation of each particle on the map. A grid of 9x9 is created around the particle and all map cells which correspond to the LiDAR scan are found. The particle with the maximum correlation is found and accordingly, the weights of all the particles are updated using softmax function through equations 13 and 14.

$$\beta^k = \exp\left(corr(y_{t+1}^k, m) - \max\left(corr(y_{t+1}^k, m)\right)\right) * \alpha_{t+1|t}^k \tag{11}$$

$$\alpha_{t+1|t+1}^k = \frac{\beta^k}{\sum_j \beta_j} \tag{12}$$

α corresponds to the particle weights in the equation. The particle with the maximum weight is the one which best defines the trajectory of the vehicle on the map. The state of this particle is used to compute the update step for vehicle and map using the LiDAR scan data. This process is followed recursively till all the data points from the sensors are accounted. Another important point here is that only one out of 5 LiDAR scan points are used to update the trajectory and map. This means that predicted step is performed for all the encoder and FOG data points whereas the updated step is performed once out of every 5 LiDAR data points. This was implemented in order to bring down the overall time taken to construct the whole map and trajectory.

Many iterations were carried out to determine the adequate noise levels. The details of the noise levels and their corresponding figure numbers are given in Table 1.

Table 1: Details of Noise Levels

Iteration No.	**Linear Velocity Noise**	**Angular Velocity Noise**	**Particle Count**	**Fig. No.**
1.	Mean = 0, Variance = 0.5	Mean = 0, Variance = 0.05	5	3
2.	Mean = 0, Variance = 1	Mean = 0, Variance = 0.05	5	4
3.	Mean = 0, Variance = 2	Mean = 0, Variance = 0.1	5	5

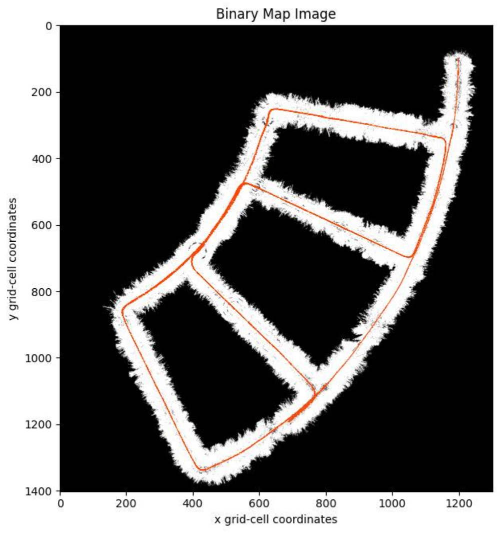


Figure 3: An Image of Map and Vehicle Trajectory for Iteration number 1 from Table 1

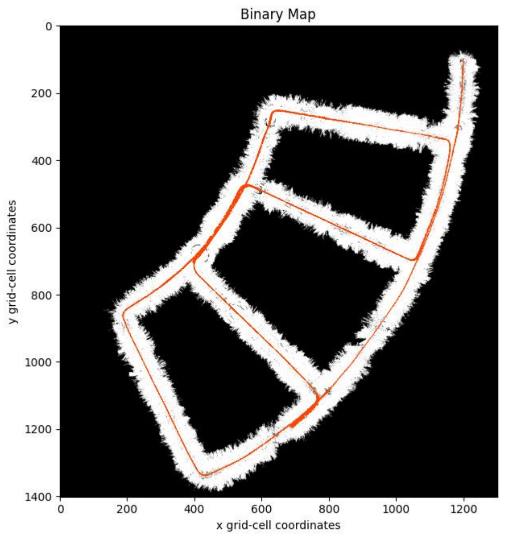


Figure 4: An Image of Map and Vehicle Trajectory for Iteration number 2 from Table 1

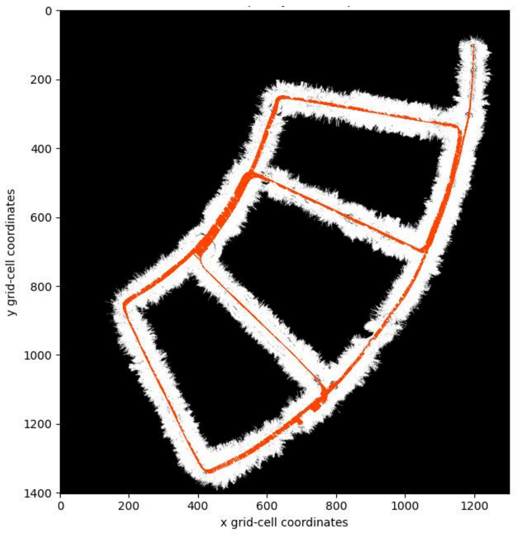


Figure 5: An image of Map and Vehicle Trajectory for Iteration number 3 from Table 1

It is clearly evident from Figures 3, 4 and 5 that as the noise increases, the vehicle is not able to maintain the right trajectory. The trajectory shown in red color in Figure 5 indicates that the particle keeps deviating from its trajectory. Hence, the further iterations were performed with the low noise of mean 0 and variance 0.5 for linear velocity and of variance 0.05 for angular velocity.

Occupancy Grid Mapping: The map is initialized as a grid of 1400 x 1400 with resolution 1. It is updated every time the update step is performed using the LiDAR scan point. The occupancy grid map is created as given in equation 5. Considering the true positives observations as 80% and false positives observations as 20%, their ratio would be 4 and hence, the log-odds ratio of log4 is considered in the project.

The log-odds for the free and occupied cells in the map are updated for every update step as per equation 6. The map PMF $\gamma_{i,t}$ is recovered from the log-odds $\lambda_{i,t}$ via the logistic sigmoid function given in equation 7. This helps in creating the binary map from the occupancy grid map. In order to prevent the values in map from becoming too high or too low, the values are bounded between -10*log-odds ratio and 10*log-odds ratio.
For every 10000 iterations, the occupancy grid map and the binary map are stored.

Resampling: Resampling becomes important in order to prevent the particle depletion. Thus, every time N_{eff} goes below the threshold, the particles are resampled according to equation 13.

$$N_{eff} = \frac{1}{\sum_{k=1}^{N}\left(\alpha_{t|t}^{k}\right)^2} \leq N_{threshold} \qquad (13)$$

IV. Results

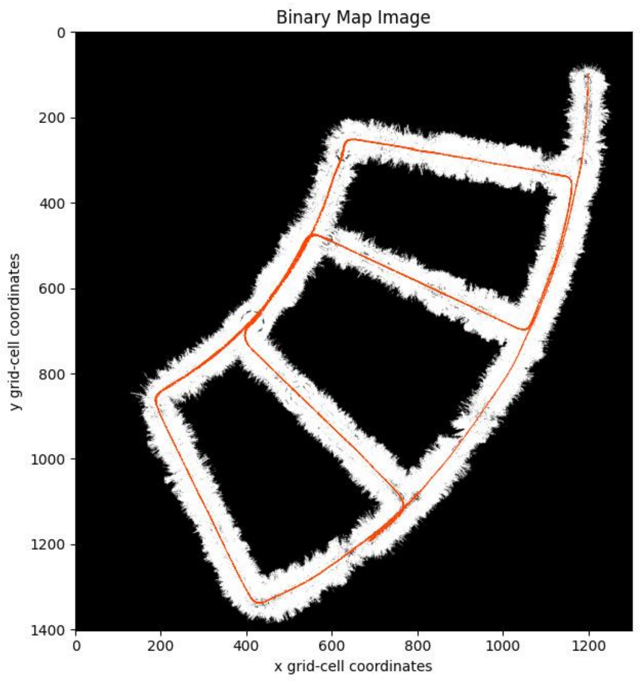
This section discusses about the results as obtained different iterations.

The important inferences learnt from all the iterations are listed below:

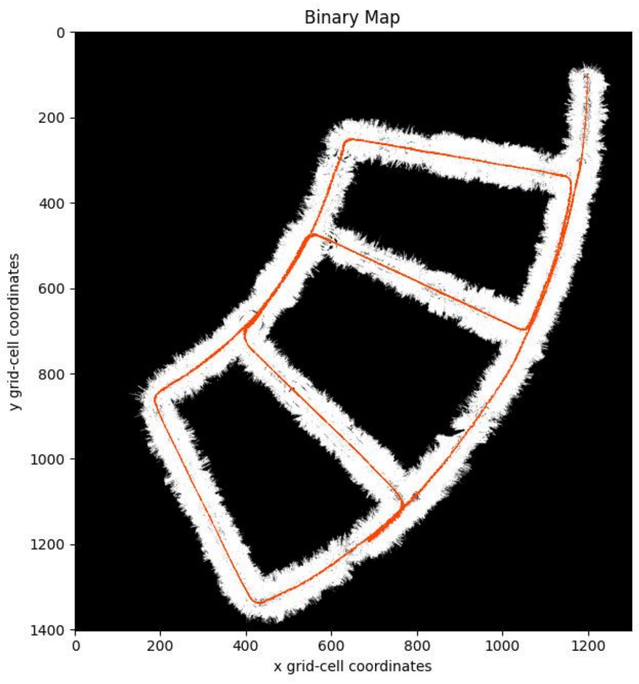
1. Number of particles is clearly a variable in this project. The trajectory of the vehicle improves as the number of particles increase. Particle count of less than 10 does not show as significant improvement as particle count of 40 or higher. The trajectory and map images of particle count 5 and particle count 40 are presented in this report. Figures 6 to 12 show the comparison between the vehicle trajectory and map for particle count 5 and particle count 40. These figures also show the occupancy grid map for particle count 40.
2. Gaussian noise to be added in linear and angular velocities is another variable in the project. As shown in Figures 3 to 5, higher noise leads to deviation of particle from its path. Lower noise level does not improve the accuracy of the trajectory of the vehicle. Even though a noise of mean 0 and variance 0.5 is added in linear velocity and of variance 0.05 is added in angular velocity for iterations in this project, it is further possible to refine the noise levels to improve the accuracy. One idea which can be implemented is the dynamic scaling of noise basis the values of linear and angular velocities.
3. It also becomes important to optimize the code in order for the iterations to take as minimum time as possible. Step size of update steps, particle count and map resolution are significant contributors to the time taken for creation of entire map. On the basis of iterations performed in this project, an estimation of time for different cases is given in table 2.

Table 2: Analysis of Time Taken for Full Map Construction

Iteration No.	**Step Size – Update Step**	**Particle Count**	**Map Resolution**	**Time Taken**
1.	All LiDAR points	1	1	15 mins
2.	1 out of every 5 LiDAR points	1	1	12 mins
3.	1 out of every 5 LiDAR points	5	1	20 mins
4.	1 out of every 5 LiDAR points	40	1	50 mins
5.	1 out of every 5 LiDAR points	1	0.1	20 hours (estm.)

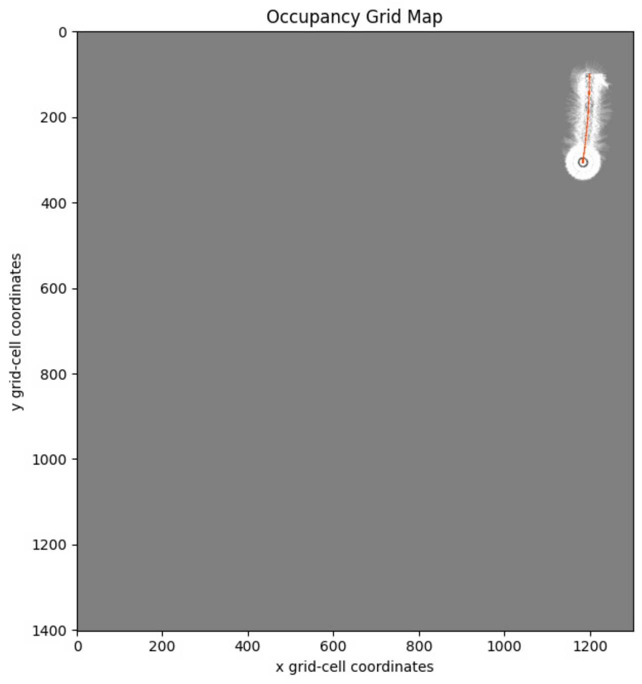


(a)

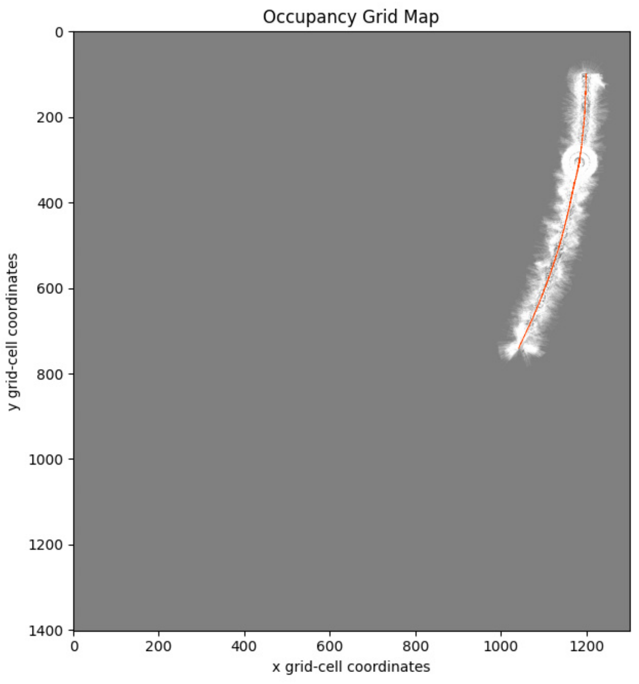


(b)

Figure 6: (a) Binary Map for Particle Count: 5, (b) Binary Map for Particle Count: 40

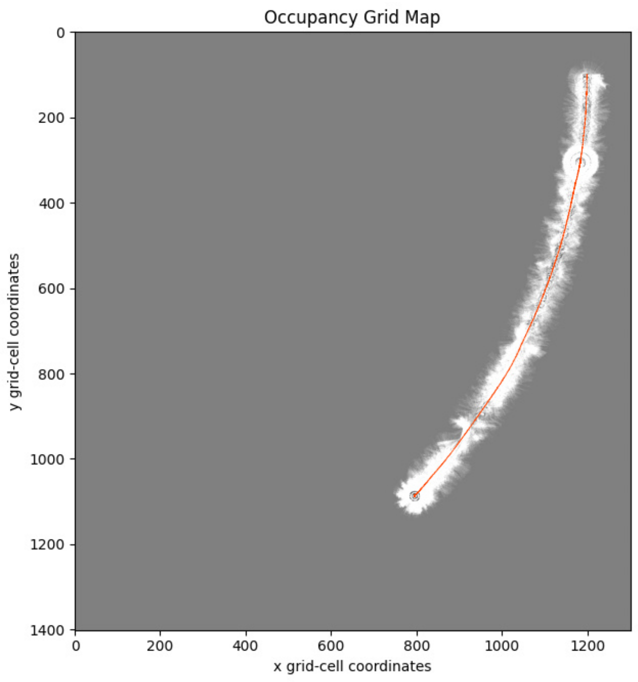


(a)

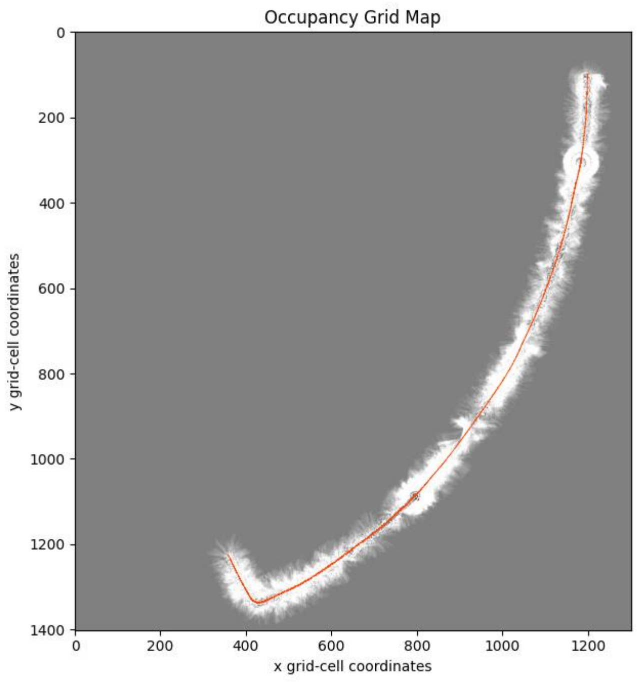


(b)

Figure 7: (a) Occupancy Grid Map for Particle Count: 40, Iteration: 10000 (b) Occupancy Grid Map for Particle Count: 40, Iteration: 20000

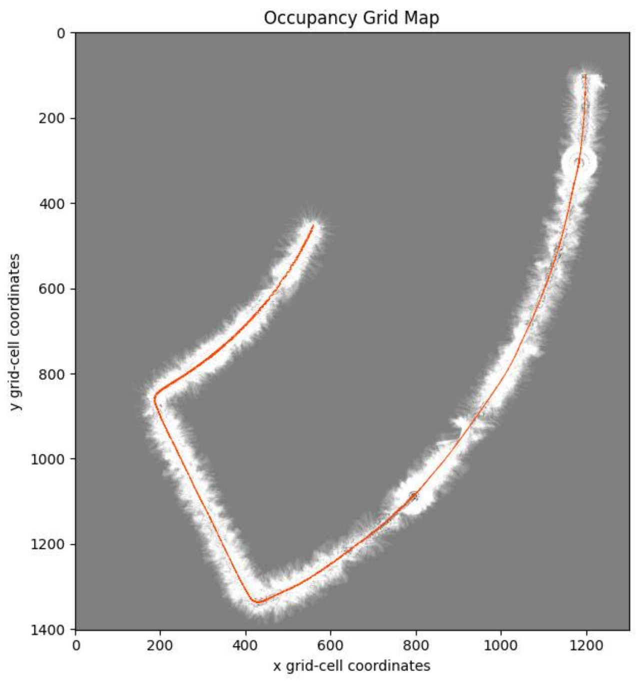


(a)

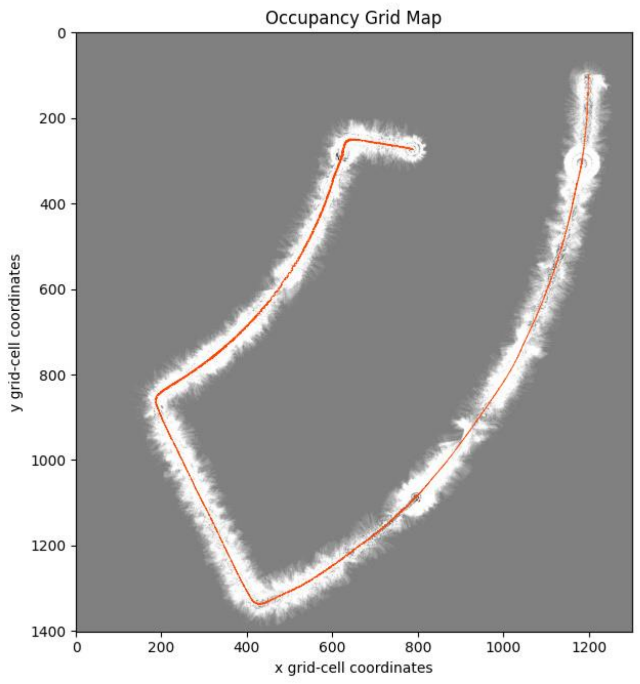


(b)

Figure 8: (a) Occupancy Grid Map for Particle Count: 40, Iteration: 30000 (b) Occupancy Grid Map for Particle Count: 40, Iteration: 40000

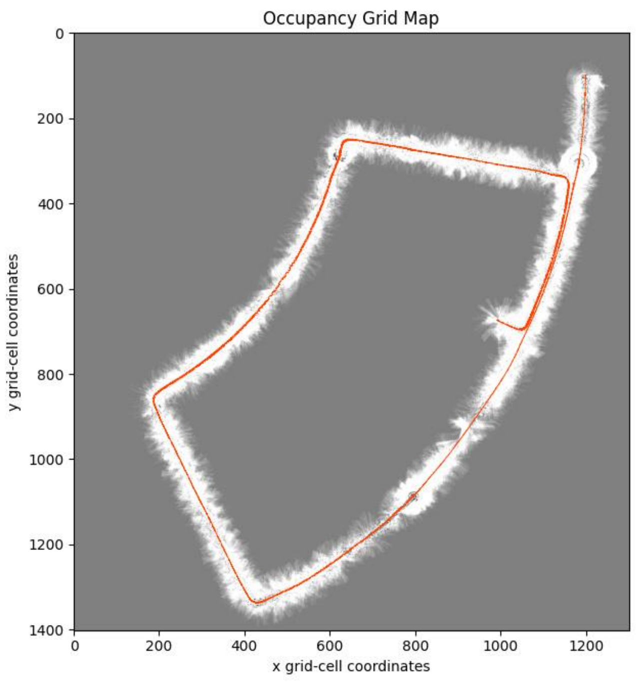


(a)

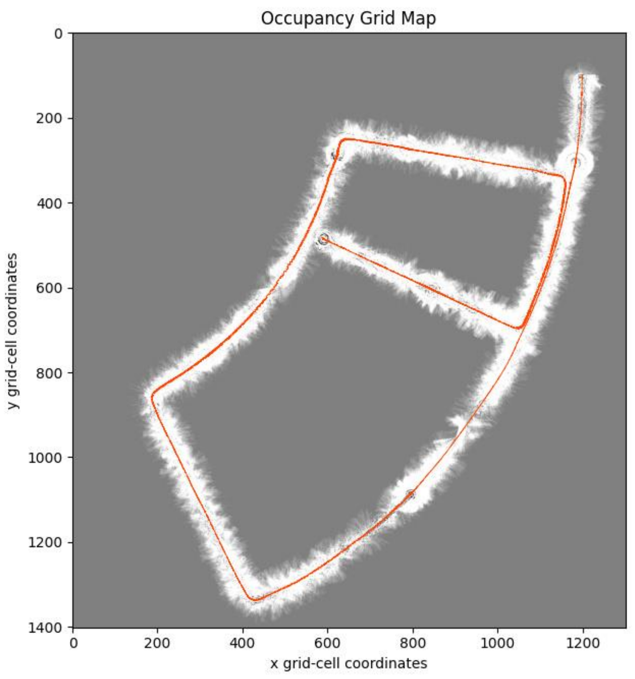


(b)

Figure 9: (a) Occupancy Grid Map for Particle Count: 40, Iteration: 50000 (b) Occupancy Grid Map for Particle Count: 40, Iteration: 60000

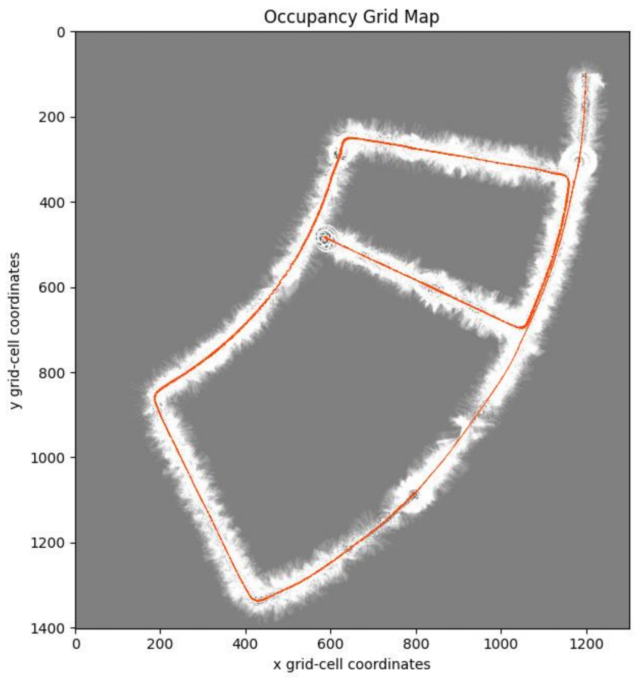


(a)

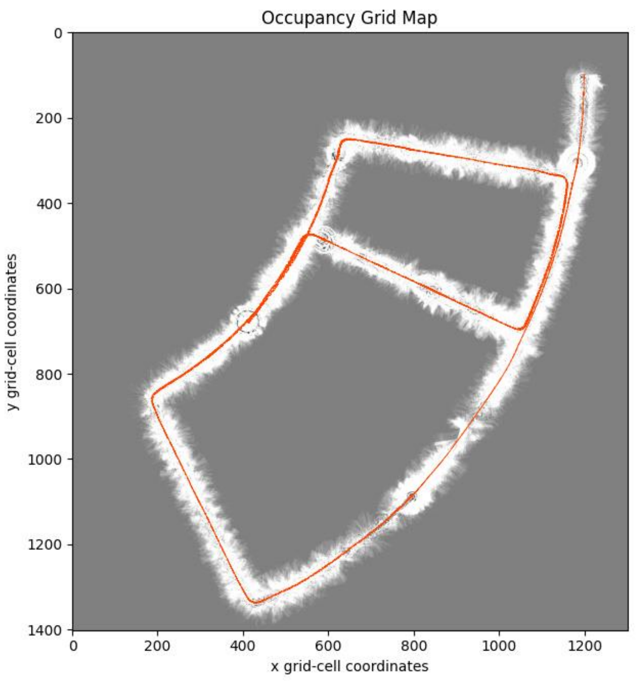


(b)

Figure 10: (a) Occupancy Grid Map for Particle Count: 40, Iteration: 70000 (b) Occupancy Grid Map for Particle Count: 40, Iteration: 80000

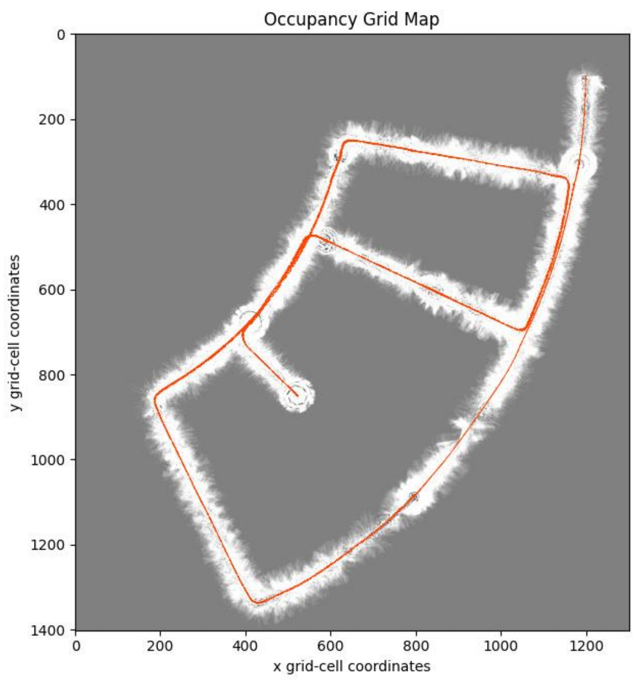


(a)

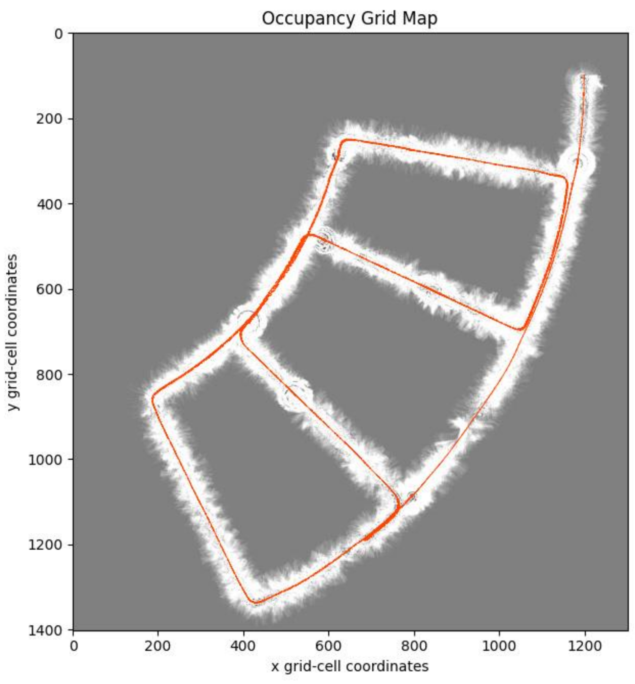


(b)

Figure 11: (a) Occupancy Grid Map for Particle Count: 40, Iteration: 90000 (b) Occupancy Grid Map for Particle Count: 40, Iteration: 100000



(a)



(b)

Figure 12: (a) Occupancy Grid Map for Particle Count: 40, Iteration: 110000 (b) Occupancy Grid Map for Particle Count: 40, Final Iteration