Particle Filter SLAM

Amitash Nanda

Electrical and Computer Engineering
University of California San Diego
La Jolla, USA
ananda@ucsd.edu

***Abstract*—Autonomous navigation requires precise and robust mapping and localization solutions in real-world scenarios. Simultaneous Localization and Mapping (SLAM) is widely used in solving this problem. SLAM usage ranges from mobile robotics and self-driving cars to unmanned aerial and underwater autonomous vehicles. It uses data from the sensors to perform mapping and localization simultaneously. Particle filter is one of the most adapted estimation algorithms for SLAM apart from Kalman Filter and Extended Kalman Filter. This project discusses an approach to solving the SLAM problem for an autonomous vehicle and attempts to understand the given implementation's shortcomings. We successfully implemented the differential-drive motion and scan-grid correlation observation models for simultaneous localization and occupancy-grid mapping.**

Keywords— SLAM, Particle Filter, Open CV, Dead-Reckoning

I. INTRODUCTION

Autonomous navigation in self-driving cars has been an active research area in the past few decades. Accurate localization and good perception of the environment are the main requirements for autonomous navigation. Global Navigation Satellite System (GNSS) solutions provide absolute positioning on earth with good precision. But, such systems are not always available or accurate depending on the environment (obstacles). Also, such perturbations can lead to errors of some meters, putting effects on safe autonomous navigation. So an autonomous car needs to navigate in a dynamic environment with obstacles without having prior information about the environment. So, to make the autonomous vehicle navigate is to represent the environment in some form. With a 3D map, the car will detect free space, obstacles, and landmarks to navigate precisely and safely. Thus it can self-explore and map an unknown environment.

Such navigation approaches termed as SLAM. It is a process by which an autonomous robotic systems construct the map of an environment using different kind of sensors, while estimating its own position in the environment simultaneously. In this project we implemented SLAM using odometry, 2-D LiDAR scans and stereo camera measurements from an autonomous car. We further used the odometry and LiDAR measurements to localize a robot and built a 2-D occupancy grid map of the environment. We successfully implemented the differentia-drive motion model and scan-grid correlation observation model for simultaneous localization and occupancy-grid mapping.

II. PROBLEM FORMULATION

A. Problem Statement

In this problem statement we are given sensor data from an autonomous vehicle. The data are collected from 3 different sensors: a fiber optic gyroscope (FOG), wheel encoder readings and front scanning 2-D LiDAR. The task of this project is to predict the state of the vehicle over time using its surrounding map that is generated along way.

B. Dataset

We are provided with encoder data from an autonomous car's left and right wheel along with the timestamps. The wheels have diameters of 0.62 m, and the encoder resolution is 4096. The FOG data contains the change in roll, pitch, and yaw of the vehicle in radians with the timestamps. Lidar scan is given to us with a start angle of -5°, end angle of 185°, angular resolution of 0.666, and maximum range of $80m$. Also, stereo images are given for texture mapping.

C. SLAM Definition

The SLAM problem is defined as follows: A mobile vehicle roams an unknown environment starting at an initial location x_0. Its motion is uncertain, making it gradually more difficult to determine its current pose in global coordinates. As it roams, the robot can sense its environment with a noisy sensor. The SLAM problem is the problem of building map of the environment while simultaneously determining the robot's position relative to the map given noisy data.

It's a parameter estimation problem for $x_{0:T}$ and m given a dataset of the robot inputs $u_{0:T-1}$, and observation $z_{0:T}$. All these parameters are related as joint pdf. It is decomposed through conditional probability, Bayes rule and Markov belief network properties. Depending on the observation and motion model, different types of bayes filter can be used for SLAM. The below states prior, observation model, motion model and control policy respectively.

$$\boldsymbol{p(x_{0:T}, m, z_{0:T}, u_{0:T-1})} = \boldsymbol{p_0(x_0, m)} \prod_{t=0}^{T} \boldsymbol{p_h(z_t|x_t, m)} \prod_{t=1}^{T} \boldsymbol{p_f(x_t|x_{t-1}, u_{t-1})} \prod_{t=0}^{T-1} \boldsymbol{p(u_t|x_t)} \quad (1)$$

D. Bayes Filter

We are trying to solve the localization problem, mapping problem and SLAM problem. A SLAM problem can be solved using Bayes Filter. It is a probabilistic inference technique for estimating the state of a dynamical system. The Bayes filter keeps track of $p_{t|t}(x_t)$ and $p_{t+1|t}(x_{t+1})$ using a prediction step to incorporate the control inputs and an update step to incorporate the measurements.

$$\boldsymbol{p_{t+1|t}\,(x) = \int p_f(x|s,u_t)p_{t|t}(s)ds} \quad (2)$$

$$\boldsymbol{p_{t+1|t+1}(x) = \frac{p_h(z_{t+1|}x)p_{t+1|t}(x)}{\int p_h(z_{t+1}|s)p_{t+1|t}(s)ds}} \quad (3)$$

A particle filter is a special kind of bayes filter that does not assume anything about the world.

E. Occupancy Mapping

Occupancy grid mapping represents the environment by dividing it into cells. It keeps track of the state of each map cell as occupied or free.

$$\boldsymbol{p(m|z_{0:t},x_{0:t}) = \prod_{i=1}^{n} p(m_i|z_{0:t},x_{0:t})} \quad (4)$$

F. Vehicle Motion Model

To determine the vehicle movement in the environment, we must use a differential motion model with general form. So, we need to use encoder and fog data to model the motion trajectory.

$$\boldsymbol{x_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = f(x_t,u_t)} \quad (5)$$

G. Vehicle Observation Model

To determine the vehicle movement in the environment, we can use an observation model z_t. The probability of z_t at time t uses the lidar data with the current state x_t and map m_t.

$$\boldsymbol{z_t = h(x_t,m_t,v_t)} \quad (6)$$

III. Technical Approach

A. Data Processing

The first step of our approach is to process the data into more tractable form and allow correspondence between the sensor values.

The wheel encoder data are given in the form of ticks count for both the wheels. We converted these values into distance travelled by the vehicle across the different timestamps. The distance Δx between the consecutive timestamps t is as follows:

$$\boldsymbol{\Delta x = \frac{\Delta ticks * \pi * diameter}{resolution}} \quad (7)$$

The data from the FOG is in rad. The values are also $10x$ faster than both the lidar and encoders. We only needed to change the yaw angle assuming that vehicle is moving in 2D. We computed the net change in the angle in FOG reading. This is done for a particular time interval corresponding to two consecutive encoder readings by summing up all the 10 delta yaw angles in that interval.

The data from the LiDAR didn't require much pre-processing. LiDAR readings gives the scans corresponds to the distance, where the vehicle observes an obstacle. They are sampled at the same rate as the encoder although with around 200 readings less.

B. Dead Reckoning

Dead reckoning is insignificant for the particle filter SLAM. Though it's a good step to build intuition of the data and we are on the right track. It simply means that the trajectory that can obtained considering the single particle using the differential drive-model. This provides an estimation of the look of the trajectory with the use of only encoder and FOG data. The motion model used to describe the rates of the states of the vehicles is as differential drive model. It as follows:

$$\boldsymbol{X = \begin{bmatrix} X_m\, cos(\theta + \Delta\theta) \\ X_m\, sin(\theta + \Delta\theta) \\ \theta + \Delta\theta \end{bmatrix}} \quad (8)$$

Here, X_m is the mean distance travelled by the vehicle using the data from both wheels. θ is cumulative angle and $\Delta\theta$ is change in angle.

Using Euler Integration, the next state is computed as follows:

$$\boldsymbol{X_{t+1} = X_t + \Delta X} \quad (9)$$

We got the coordinates of the vehicle in the world frame.

C. Mapping

Once we obtained deterministic single particle trajectory, the next task was to create a map of the surrounding of the vehicle. We created a $1400 * 1400$ occupancy grid map with 1m resolution. Given a new lidar scan z_{t+1}, we removed the points that are too close ($< 2m$) or too far ($> 75m$) from the vehicle. Then we did the polar to cartesian coordinates conversion of the readings (scan endpoints). Further pose transformation is applied to represent the points from LiDAR frame to the vehicle frame and finally to the world frame.

We converted these coordinates to grid cell units and determined all the cells between the lidar start and end points using bresenham2D ray tracing algorithm. The cells between

the start and end point are free and last cell detected is assumed to be blocked. At the end we updated the log-odds map cells as occupied or free by increasing or decreasing cells by probability mass function by $\log 4$. Also, we clipped the cell values to avoid overconfident estimation.

$$\boldsymbol{\lambda_{i,t+1} = \lambda_{i,t} \pm \log 4} \quad (10)$$

$$\boldsymbol{-10 \log 4 \leq \lambda_{i,t} \geq 10 \log 4} \quad (11)$$

We have taken log4 to assert that our sensors are 80% accurate. We plotted the log-odds occupancy grid map, once we are finished with the mapping. We can recover the map probability mass function from the log-odds map using the logistic sigmoid function to generate a binary map of the world.

$$\boldsymbol{y_{i,t} = \frac{e^{\lambda_{i,t}}}{1 + e^{\lambda_{i,t}}}} \quad (12)$$

D. Particle Filter SLAM

Till now we have implemented the trajectory and mapping for a single particle without any probabilistic approach, the next step is to implement particle SLAM. Here we have taken a probabilistic approach to solving this localization and mapping problem by adding multiple particles with certain noise. Thus, representing the best possible trajectory.

We have taken a set of $\boldsymbol{n = 5}$ particles initially, all having their own 3D states $\boldsymbol{p = [x, y, \theta]}$ just like the single vehicle before. They also have their own assigned weights α which are initialized as $\boldsymbol{1/n}$. The map is updated at any time by taking the particle state with the highest weight.

The $\Delta\theta$ obtained from the FOF readings and the differential-drive model, we predicted the updated pose of each particle as follows:

$$\boldsymbol{u_{t+1} = u_t + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \end{bmatrix}} \quad (13)$$

We added Gaussian noise to each of the particle state in all degrees using zero mean and a variance of $\boldsymbol{\max(\Delta x/10)}$, $\boldsymbol{\max(\Delta y/10)}$, and $\boldsymbol{\max(\Delta\theta/10)}$, like the form of a noisy envelope around the particle trajectory.

E. Particle Filter SLAM Update

Once the prediction step is completed, we updated the pose and weight of each particle using the map correlation function. We achieved this be converting the LiDAR scan points to the world frame coordinates and created a $9 * 9$ grid around each particle's pose. Hence finding the map cells corresponding to this pose and computed map correlation for each particle.

After that we found the best correlation particle with projected scan using the prior map. Particle weights are further calculated using the SoftMax function and the **Dead-Reckon** particle state with the highest wight becomes the new vehicle position considered for our optimum trajectory. Once we got the best particle state, we updated the log-odds map. Again, we started with this particle position and projected the new lidar scan from the location. To deal with possible depletion of the particles, particle resampling can be implemented, when our effective number of particles become less than the assigned threshold.

IV. RESULTS

In this project we have implemented the particle filter SLAM for $n = 5, 10, 20$ particles. We have plotted the log-odds occupancy map and binary map at various intervals for particularly $n = 5$ particle case. Also, we have updated the map only at every 5 LiDAR readings for every case. Gaussian noise is used for each particles state using zero mean and variance of max $(\Delta x/10)$, max $(\Delta y/10)$ and max $(\Delta\theta/10)$. The dead-reckoning trajectory path is also plotted prior to using particle filter. We noticed that particle filter SLAM gives almost same trajectory as the dead-reckoning trajectory with some variations. One particular thing to note is the realization of the lane on which the vehicle traverses which becomes very conspicuous from the SLAM implementation. We observed that the change in the number of particles doesn't have much of an effect on the map or the vehicle trajectory. The trajectory aligns well through out the journey. All the plots are plotted below.

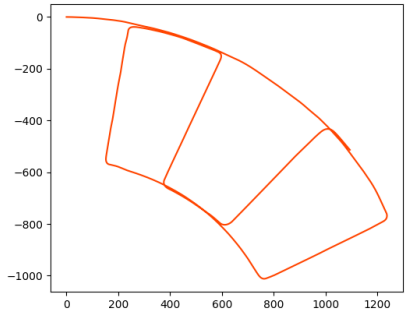


Fig 1. Dead-Reckoning

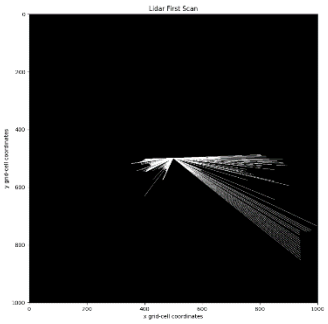


Fig 2. First Lidar Scan

Fig. 1.

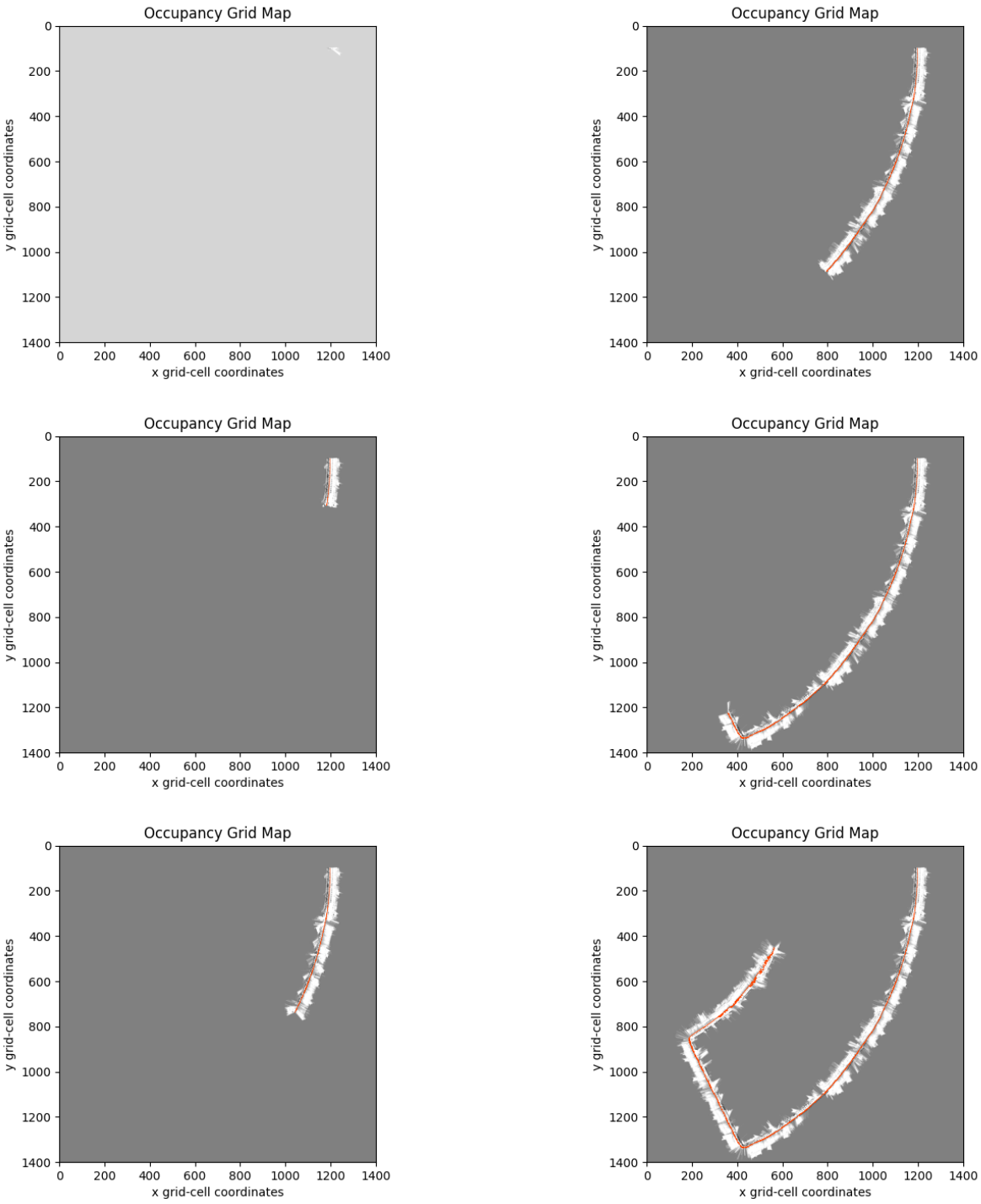
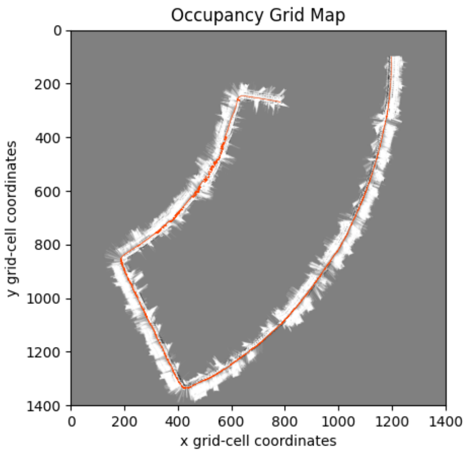
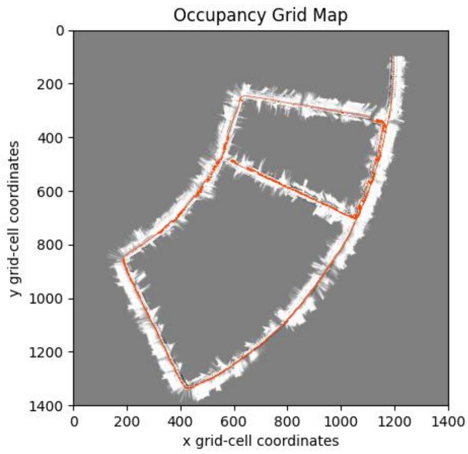
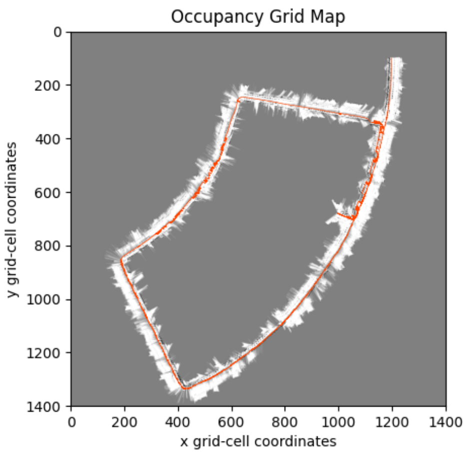
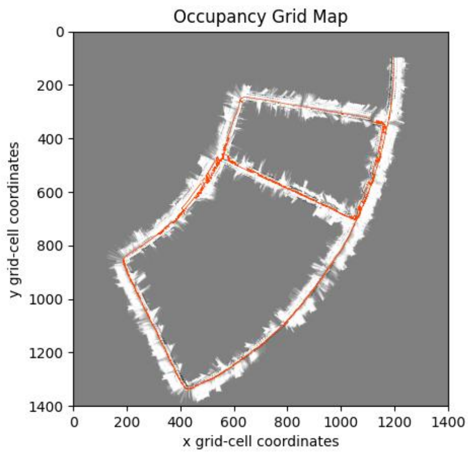


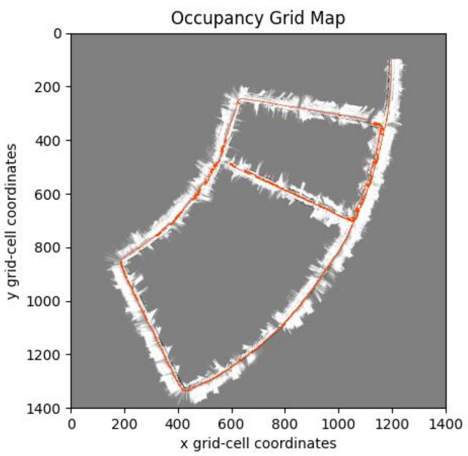
Fig 3-8. Occupancy grid map with 5 particles

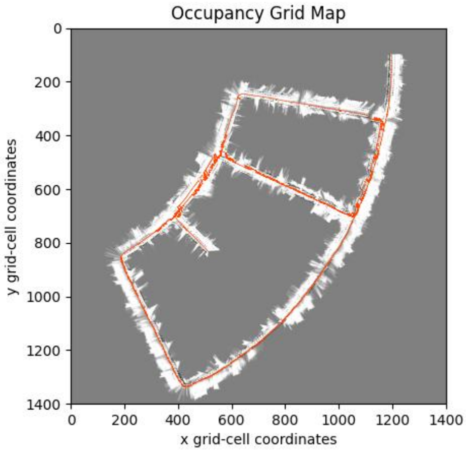

Occupancy Grid Map
y grid-cell coordinates
x grid-cell coordinates
0 200 400 600 800 1000 1200 1400

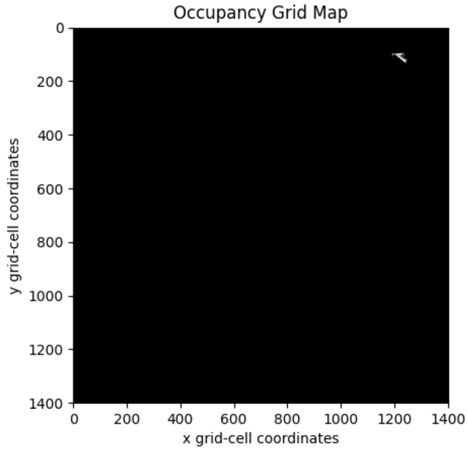

Occupancy Grid Map
y grid-cell coordinates
x grid-cell coordinates
0 200 400 600 800 1000 1200 1400

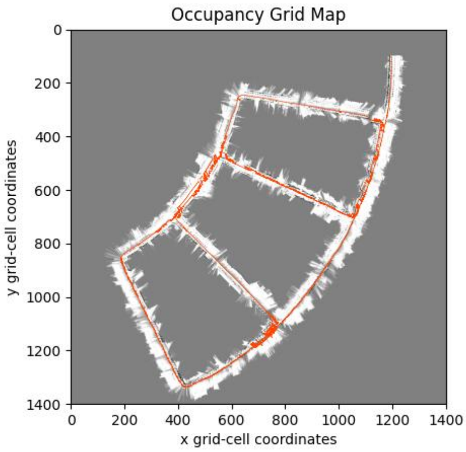

Occupancy Grid Map
y grid-cell coordinates
x grid-cell coordinates
0 200 400 600 800 1000 1200 1400


Occupancy Grid Map
y grid-cell coordinates
x grid-cell coordinates
0 200 400 600 800 1000 1200 1400


Occupancy Grid Map
y grid-cell coordinates
x grid-cell coordinates
0 200 400 600 800 1000 1200 1400







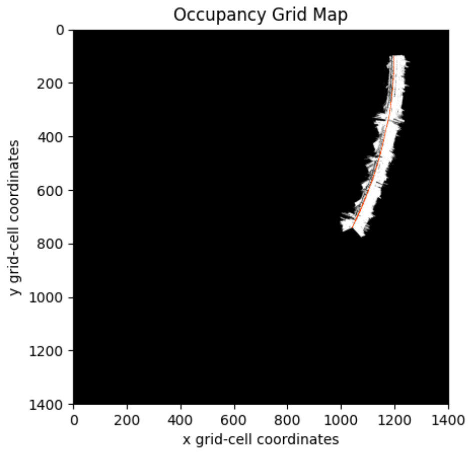
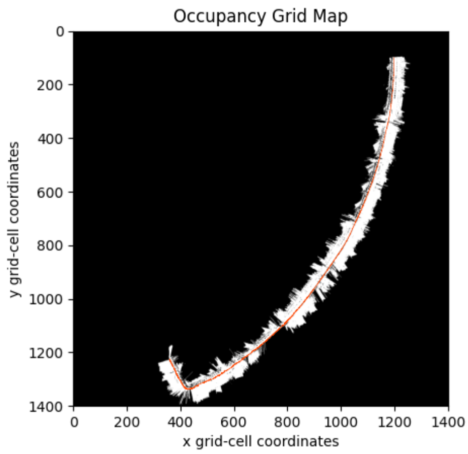


Fig 9-15. Occupancy grid map with 5 particles



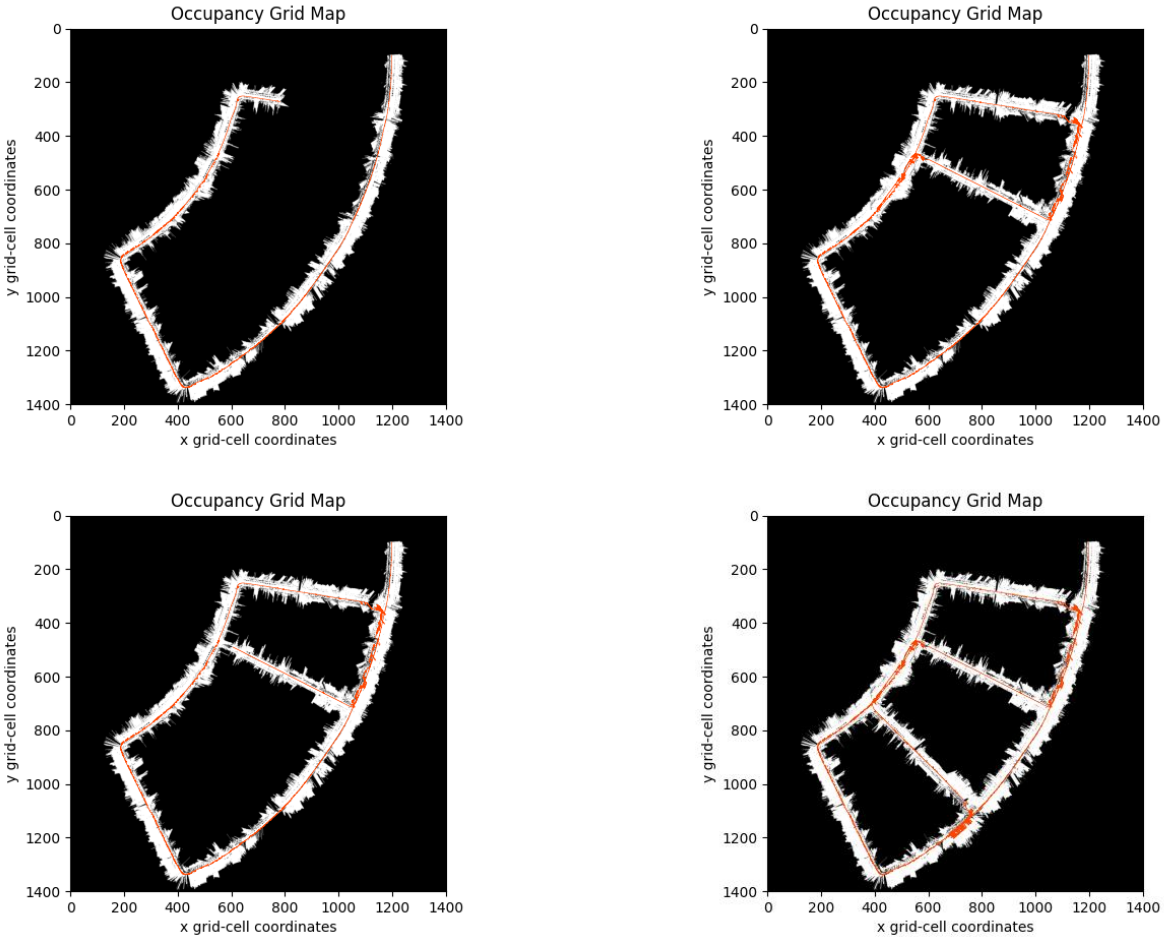


Fig 16-22. Binary Map with 5 particles

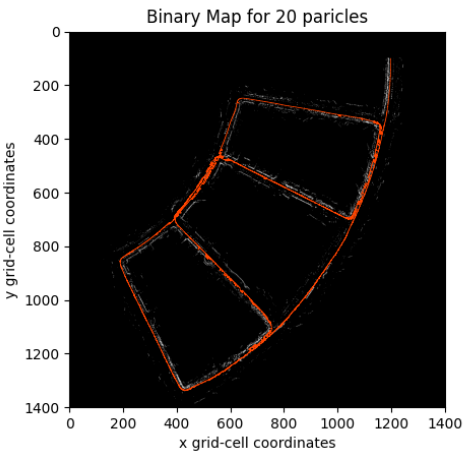
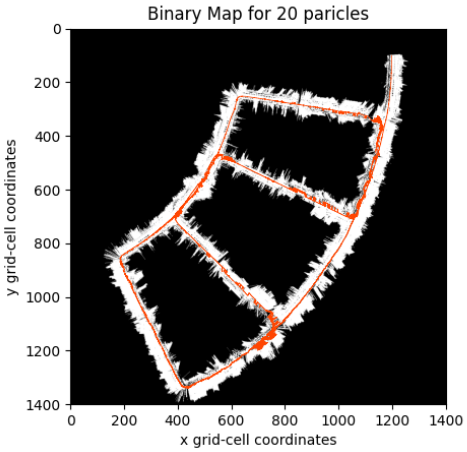


Fig 23. Binary Map Trajectory



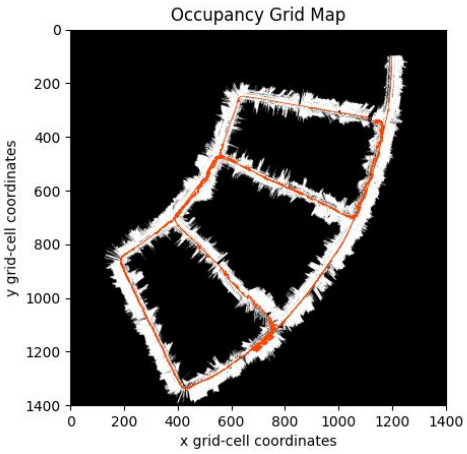


Fig 24. Binary Map with 20 particles

Fig 25. Binary Map with 10 particles