Particle Filter SLAM

Lulua Rakla
PID A59012895
lrakla@ucsd.edu

***Abstract*—This report presents approaches to Simultaneous Localization and Mapping of an automobile trajectory using odometry and lidar data (a differential drive motion model). The output is given in forms of occupancy grid maps and texture maps.**

***Index Terms*—SLAM, computer vision, autonomous driving**

I. Introduction

Simultaneous Localization and Mapping (SLAM) is an approach to help a robot or autonomous agent navigate in an unknown environment. Using readings from various sensors, the agent is able to simultaneously generate the map of the environment and keep a track of its position in it. It seems like a chicken-egg problem as agent needs a map of the environment to know where it is, and it needs to know where it is to figure out the map. There are many approaches which vary depending on the motion and observation model distributions selected (Rao-Blackwellized Particle Filter, Extended Kalman Filter, Unscented Kalman Filter, Factor Graphs etc) This paper presents an approach to localization and mapping using a variation of FASTSLAM using Particle Filter. A Particle Filter [1] is used to maintain the PDF of the robot trajectory and log-odds mapping to maintain a probability map for every particle. Lastly, texture mapping is done to project the RGB color values from the stereo cameras to the occupancy grid map.

II. Problem Formulation

SLAM is a parameter estimation problem for state $x_{0:T}$ and map m conditioned on the robot inputs $u_{0:t-1}$ and observations $z_{0:t}$ In this particular project, the problem is to estimate the following

$$p(x_{0:T}, m|u_{0:t-1}, z_{0:t}) \tag{1}$$

x : state of the vehicle
m : map of the environment
u : control input
z : observations
t: time

This can be estimated using the data provided

A. Dataset

The readings of the following sensors and their parameters are used
- Enocoders : Encoder wheel counts and diameters of the left and right wheel are provided. The encoder has resolution of 4096. Encoder readings are used to calculate the predicted position of the vehicle by using the instantaneous linear and angular velocity.
- Fiber Optic Gyroscope (FOG) - FOG readings are provided as delta yaw, delta roll and delta pitch. Out of these, delta yaw can be used to estimate vehicle orientation. FOG sample rate is 10x the encoder sample rate.
- Lidar (LMS511) - Lidar scans are provided using a 2D lidar with FOV 190 degrees, max range of 80m, and angular resolution 0.666. Lidar scans act as observations to update the confidence of the vehicle position in a particular place.
- Stereo Images - 1161 left and right stereo images are provided in Bayer format. The camera parameters such as their baseline values, projection and camera matrices are provided.
- Parameters - Various transformation between frames (lidar to vehicle, stereo to vehicle, etc) and parameters unique to the sensor are provided. These aid in solving equations of velocity, depth etc

B. Mapping

The problem here is to build a map m of the environment using the robot state trajectory $x_{0:T}$ and sensor observations z. Occupancy grid mapping is a 2D mapping technique which divides the map into cells with values independently conditioned on the robot trajectory.

$$p(m|z_{0:t}, x_{0:t}) = \prod_{i=0}^{N} p(m_i|z_{0:t}, x_{0:t}) \tag{2}$$

Here $m \in R^2$. Each m_i has a value 1 if it is occupied or -1 if it is free.

C. Localisation

In this step, given a map m, a sequence of control inputs $u_{0:t-1}$ and a sequence of measurements $z_{0:t}$, infer the particle state at x_t. In this project, each particle $\mu_{t|t}^{(k)}$ is a hypothesis on the state x_t with confidence $\alpha_{t|t}^{(k)}$. The particles indicate the pdf of vehicle state at time t as shown in Fig. 1

1) Predict: In this step of localisation, the motion model p_f is used to get the predicted pdf $p_{t+1|t}(x_{t+1})$. For every particle $\mu_{t|t}^{(k)}$, $k = 1, ..., N$ compute equation in Fig 2. f is the differential drive model, u_t is the linear and angular velocity and ϵ_t is Gaussian Noise. The problem here is to get the change in particle state using motion model and noise.

$$p_{t|t}(\mathbf{x}_t) := p(\mathbf{x}_t \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1}, \mathbf{m}) \approx \sum_{k=1}^{N} \alpha_{t|t}^{(k)} \delta\left(\mathbf{x}_t; \boldsymbol{\mu}_{t|t}^{(k)}\right)$$

Fig. 1. Vehicle state pdf (from slides)

$$\boldsymbol{\mu}_{t+1|t}^{(k)} = f\left(\boldsymbol{\mu}_{t|t}^{(k)}, \mathbf{u}_t + \boldsymbol{\epsilon}_t\right) \qquad \alpha_{t+1|t}^{(k)} = \alpha_{t|t}^{(k)}$$

Fig. 2. Predict Step (from slides)

2) Update: In this step of localisation, the observation model p_h is used to get the updated pdf $p_{t+1|t+1}(x_{t+1})$. Th particle pose is unchanged, but weights are updated using the observation model (Fig. 3). The problem here is to define the observation model using lidar readings, and update weights using a laser correlation model.

D. Texture Mapping

A texture map is $m \in R^3$ where each cell in this map has a RGB value associated with it according to the projected pixel coordinates in the occupancy grid plane. The RGB color is obtained from 2×1161 left and right stereo images. Each image has $h \times w$ of 560×1280.

III. Technical Approach

This section outlines the technical details of the project

A. Preprocessing

The sensors are in this project are wheel encoders, FOG and 2D lidar. The sensors are not synchronized (each having a different sample rate). FOG and encoder sync was done by the fact that there are 10 FOG values for every encoder reading. The 10 values were summed up (as each represents delta yaw). The lidar was downsampled by a factor of 10 (this was done to not increase SLAM time in update step). The synced FOG values were saved.

B. Motion Model

The equations for the Euler discretized differential drive motion model over time interval τ are shown in in Fig 4. The predicted state of a particle is its current state added to its distance moved in time τ.For e.g Distance in x direction is a function of linear velocity at time t ν_t, time interval τ and particle orientation θ. The θ_t values at obtained from FOG sensor.

C. Observation Model

In the obsevation model, given observation z_t, the objective is to calculate p_h. The observations are present in form of lidar scans. The model converts the observations from cartesian to polar coordinates using the matrix given below. These scans are then used to get map log odds correlations which are described further in Update section.

$$p_h(z_t|x_t, m_t) \tag{3}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} rcos\theta \\ rsin\theta \\ 0 \end{bmatrix}$$

D. Prediction

To start the prediction step using a particle filter, it is important to understand what a particle $\mu_{t|t}^{(}k)$ represents. Each particle is initialized as μ = $[x, y, \theta]$ where x,y are position and θ is orientation. All particles are initialized to zeros with a uniform weight $\alpha_{t|t}^{(}k)$ of $1/N$ where N is number of particles. Over time the particles concentrate on locations which have higher weight. The theoretical formulation is given in Fig 2. Using the differential-drive model with linear velocity from the encoders and angular velocity from the gyroscope the next state of each particle was predicted in this step. The particle position was changed, but their weights remain the same. As shown in eq (3), the linear velocity ν is dependent on dimater of wheel d,counts in time τ, z and encoder resolution 4096. The linear velocities were calculated separately for the two wheels and their average was taken. τ, the difference in time was obtained from the encoder timestamps.

$$\tau\nu = \frac{\pi dz}{4096} \tag{4}$$

Dead reckoning was done to estimate the initial trajectory. In Dead reckoning only one particle's predicted steps were mapped, without adding any noise. The dead reckoning is shown in Fig 5.

E. Updation

The theoretical formulation for updation is given in Fig 3. In this step, using the lidar scans the particle weights were updated using the laser correlation model. Here $y = r(z, x)$ is the lidar scan's world frame projection converted to grid cells and x is vehicle pose and m is map.

$$p_h(z_{t=1}|\mu_{t|t}^{(k)}, m) \propto corr(r(z, x), m) \tag{5}$$

Laser correlation model calculates the correlation between the particles world state position as a cell y_i and it's observation on the map m_i.

$$corr(y, m) = \sum_i 1(y_i = m_i) \tag{6}$$

The process involves converting laser scan to world coordinates using the particles, and find correlation between these occupied/free cells and the current binary map cells. The

$$\mu^{(k)}_{t+1|t+1} = \mu^{(k)}_{t+1|t} \qquad \alpha^{(k)}_{t+1|t+1} \propto p_h(\mathbf{z}_{t+1} \mid \mu^{(k)}_{t+1|t}, \mathbf{m})\alpha^{(k)}_{t+1|t}$$

Fig. 3. Update Step (from slides)

$$\mathbf{x}_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = f(\mathbf{x}_t, \mathbf{u}_t) := \mathbf{x}_t + \tau \begin{bmatrix} v_t \cos(\theta_t) \\ v_t \sin(\theta_t) \\ \omega_t \end{bmatrix}$$

Fig. 4. Differential Drive Motion Model for Discrete Time (from slides)

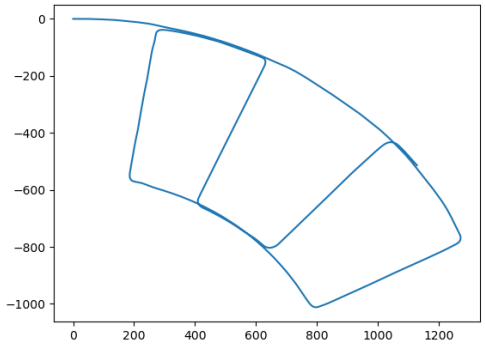


Fig. 5. Dead Reckoning

correlation is not calculated at each particle position, but a small 9×9 perturbation is considered. The particles with max correlation have their weights updated in this step. The particles are the best particles (largest weights)

F. Mapping

To test out the frame transformations (Lidar to Vehicle, Vehicle to world), the first lidar scan was projected on the occcupancy map - the faint black at the centre of square.(Fig 6) The given lidar readings were converted from polar to cartesian coordinates. Valid ranges < 70 and > 2 was selected after tuning this parameter. The map cells were initialized using a uniform prior (each cell is equally likely to be free or occupied). To update the map at every iteration, the lidar scans were converted to world frame using pose of the best particle (particle with largest weight α). The world coordinates of the end points are then passed through bresenham2D which gives the occupied and free cells. If a cell is observed free, decrease log odds ratio $\lambda_{i,t}$ by $log(4)$ and increase it if cell is occupied. Lastly, the PMF of map can be obtained by by using a sigmoid function.

$$m_i = \begin{cases} +1 \text{ if i cell is occupied w.p } \gamma_{i,t} \\ -1 \text{ if i cell is free w.p } 1 - \gamma_{i,t} \end{cases} \tag{7}$$

$$\lambda_{i,t} = \lambda_{i,t-1} \pm log(4) \tag{8}$$

G. Re-sampling

Resampling was done after each update step when N_{eff} was less than 60% of the particles. The sampling algorithm used was Stratified Resampling (Fig. 7). This is to prevent number of particles from reducing when their weight becomes less. The number of particles with larger weights are increased and their weights are normalized to $1/N$. Stratified Re-sampling ensures particles with larger weights are sampled at least once. This reduces variance in the final particle distribution.

H. Texture Mapping

This step has been implemented in code partially. It involved the following process

Get the disparity between the left and right stereo images using StereoBM_create.

Use the focal length d, baseline value$b$ pixel value u_l, v_l of left camera and disparity d to get x,y and z in optical or camera frame. These baseline and scaling values are given in the projection matrices.

$$z = \frac{fs_u b}{d} \tag{9}$$

z is depth of the image

$$y = \frac{(v_l - c_v) * z}{fs_v} \tag{10}$$

$$x = \frac{(u_l - c_u) * z}{fs_u} \tag{11}$$

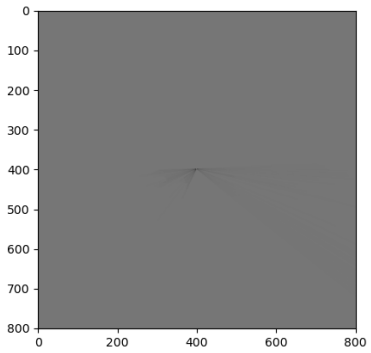


Fig. 6. First Lidar Scan at (0,0)

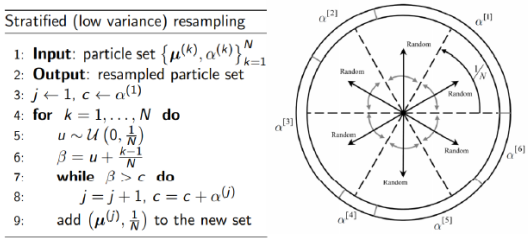


Fig. 7. Stratified Resampling

TABLE I
SLAM PARAMETER VALUES

Parameter	Description	Values
Map[xsize]	Width of MAP	1400
Map[ysize]	Height of MAP	1400
Map[res]	Resolution of MAP	1
$nparticles$	Number of Particles	[1,5,10,20,50]
$N_{threshold}$	$N_{eff} < N_{threshold}$	$0.6 * nparticles$
Gaussian Noise	Noise added to particle	$\phi(0, diag(\sigma_{vt}, \sigma_{wt}))$
Lidar Downsample	Downsample rate of lidar	10

1) After getting the optical coordinates, the pose of the camera is used to project the coordinates to vehicle frame and then to world frame(using particle pose). This is done at every update step.
2) The world frame coordinates of pixel are then converted to occupancy map cell coordinates. Here only those pixels are considered whose z value lies within a threshold of the map and assign those cells the associated RGB values.

I. Noteworthy Observations

- Skipping an iteration of SLAM when linear velocity is zero sped up the process. This works because at the two intersections where the car encounters a red light, the encoder and lidar readings are the same.
- Downsampling Lidar decreases the time for update step. Every 10 measurements of lidar are loaded. Out of these further filtering out measurements which are too close or too far reduces number of scans.
- Adding Gaussian noise with variance equal to the change in position from x_{t-1} to x_t allows the noise to be limited, but not too less.

IV. RESULTS

The final trajectory and occupancy maps are shown at the end of the report. The final SLAM parameters are shown in Table I. The trajectories were mapped for 1,5,10,20 and 50 particles. The images in this report show the trajectory for 20 particles at various iterations. Some iterations are skipped as the vehicle encounters red lights and has to stop. So the trajectory does not vary for those periods. (There are 2 such periods) In general, as the particle count was increased, the trajectory became more detailed with less noise. (Fig 10.) SLAM ran for 127635 iterations, with Final Predictions = 64927, Final Updations = 11587. As number of particles increase, the time for SLAM to run also increases. This places a computational and time constraint on this approach. Adding Gaussian Noise to the particle pose improved the localisation as particles can spread out more and detect more cells being occupied/free. Adding noise also makes the system less overconfident in any one position. This can also be done by restricting log odds between certain values $\lambda_{MIN} \leq \lambda_{i,t} \leq \lambda_{MAX}$ Re-sampling ensured that the particles would not be depleted, this makes the final trajectory have less error.

As seen in Fig 10. adding more particles makes the map "smoother". Another way to do this would be to increase resolution of the map so more world coordinates can be mapped to individual cells. However, on decreasing resolution, the computation time increases significantly.

Lastly, texture mapping was implemented partially in the update step (due to time constraints). The pixel coordinates of the images were converted to real world coordinates (stereo images => disparity => depth => optical coordinates => real world coordinates => select an image plane => map RGB colors to cell.

REFERENCES

[1] https://natanaso.github.io/ece276a/ref/ECE276A10ParticleFilterSLAM.pdf

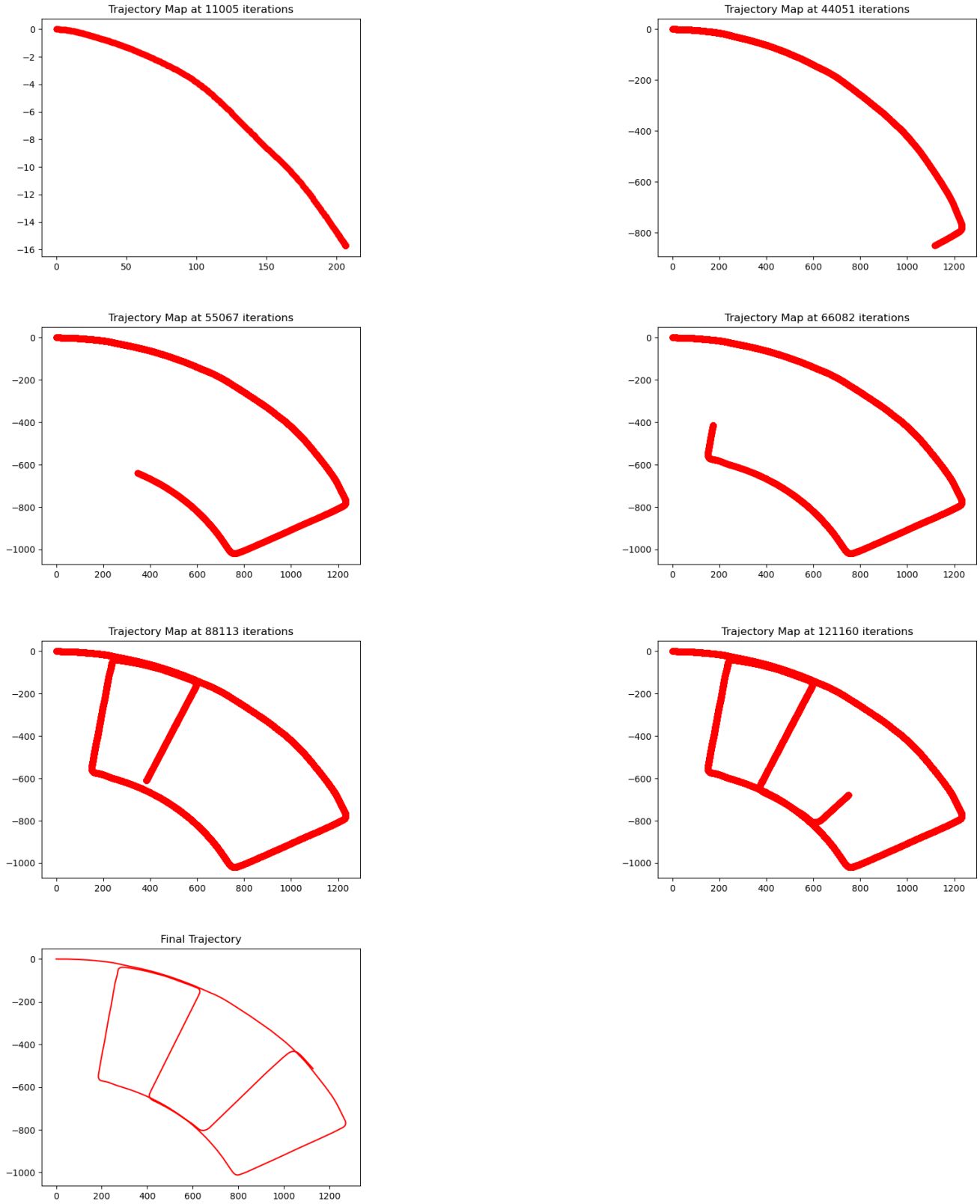


Fig. 8. Trajectories at various iterations

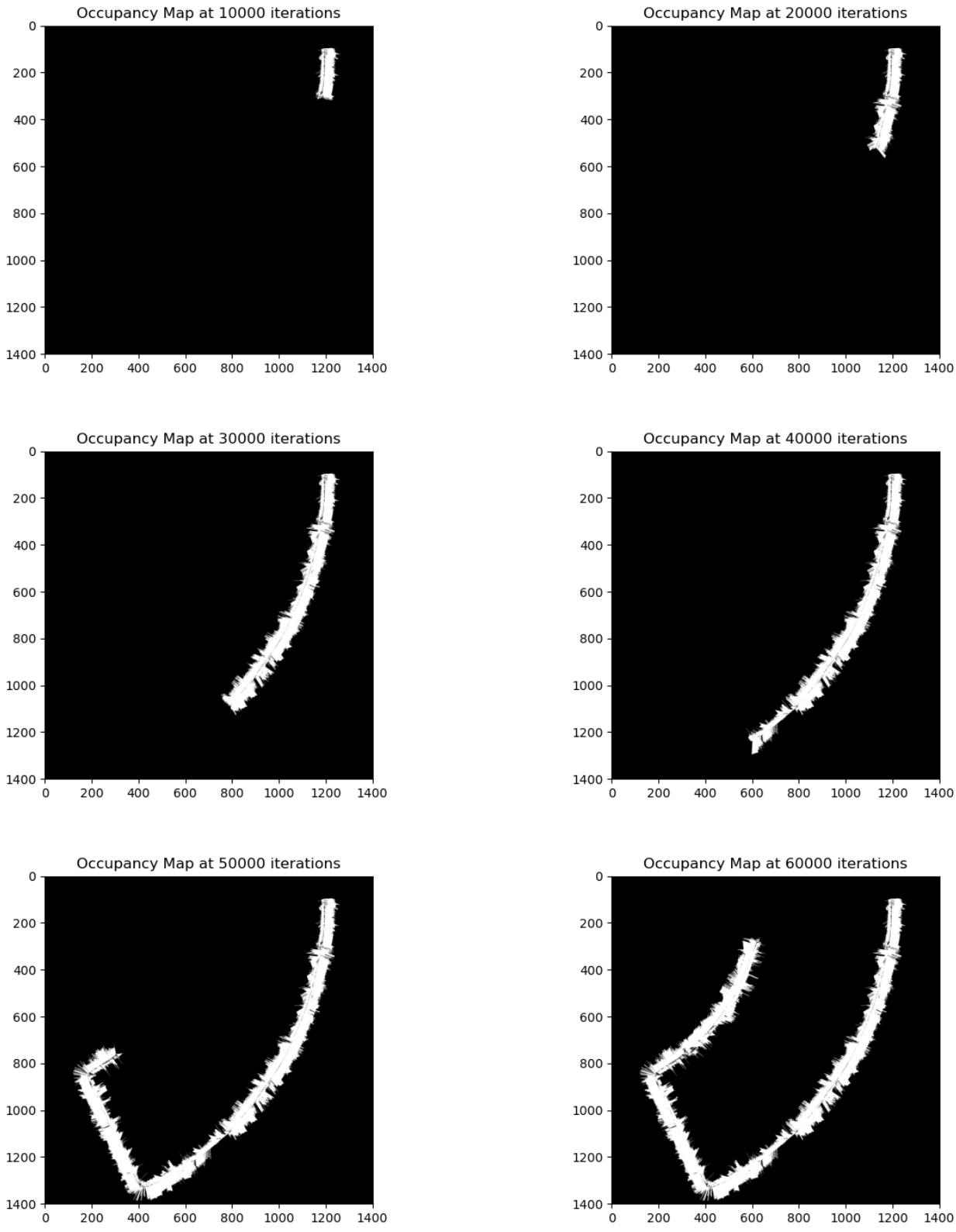
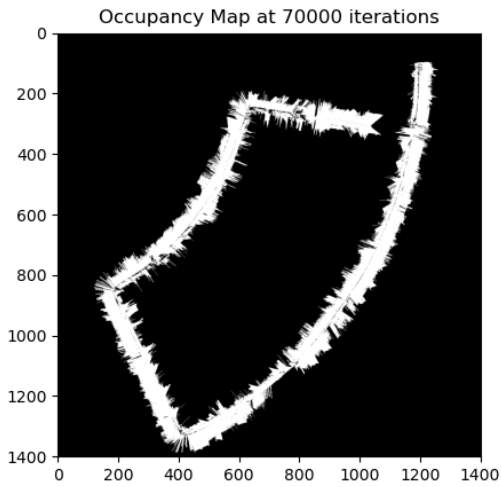
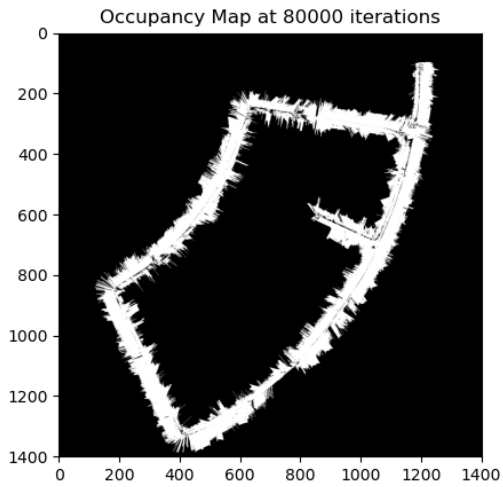
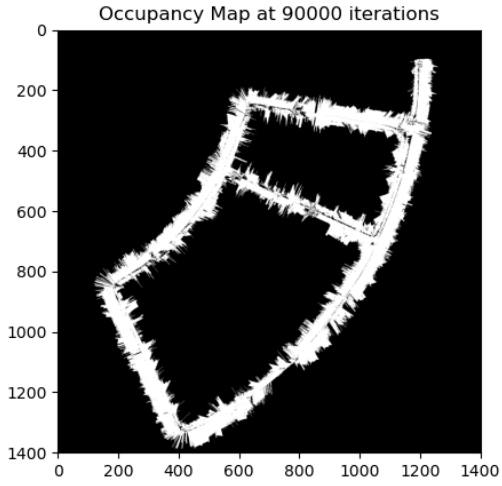
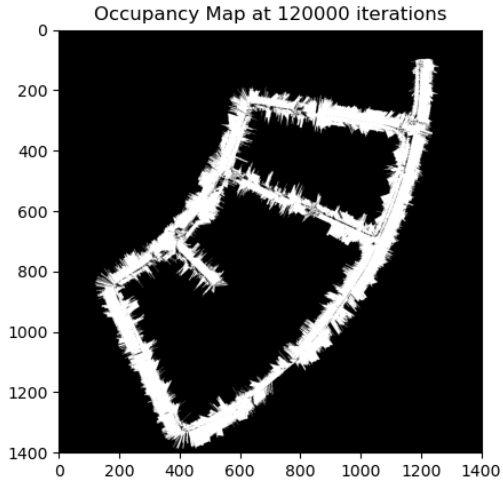


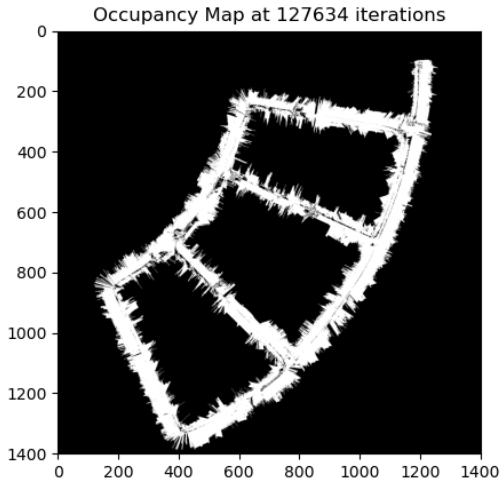
Fig. 9. Occupancy Maps at 10k,20k,30k,40k, 50k and 60k iterations for 20 particles











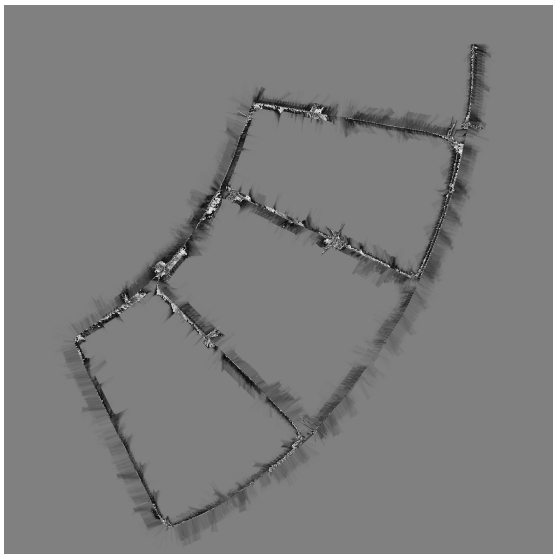
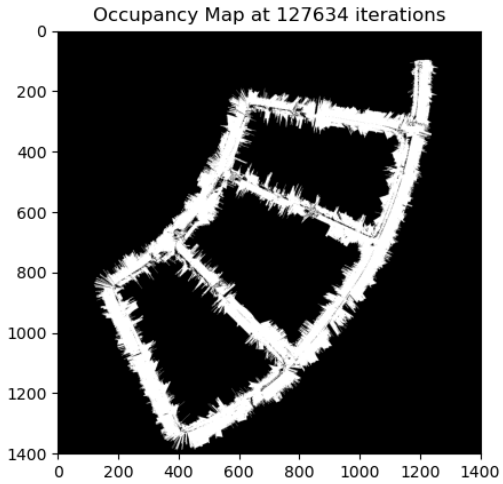
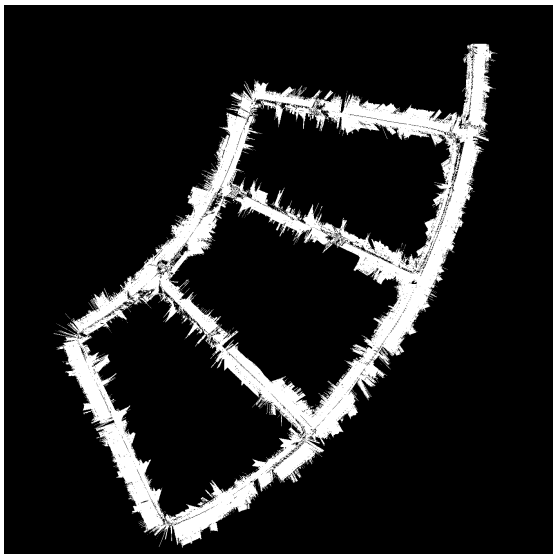


Fig. 10. Occupancy Maps at 70k,80k,90k,120k, 50k and last iteration for 20 particles





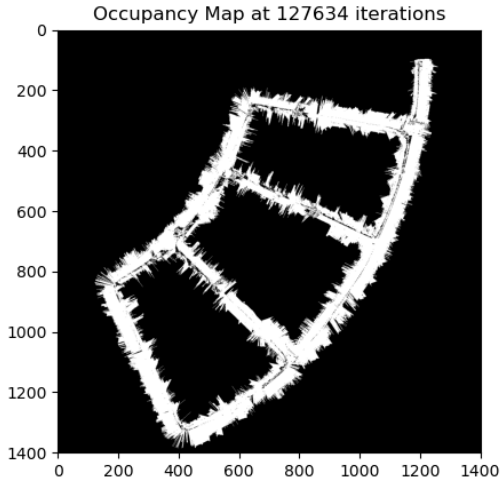


Fig. 11. Occupancy Maps for 1, 20 and 50 particles