Particle Filter SLAM and Texture Mapping

Saqib Azim
ECE Department
UC San Diego
sazim@ucsd.edu

Abstract

In this project, we have implemented a SLAM system to localize and map the environment simultaneously using particle filters. The system uses data from multiple sensors such as encoders, LIDAR, IMU, and RGBD Kinect camera. The project is divided into two parts - in the first part, we use particle filter algorithm to localize and map the environment using only LIDAR, encoder and IMU sensors. In the second part, we add texture to the created map using data from RGBD Kinect camera attached to the robot and the optimal robot trajectory obtained from particle filter algorithm in the first part.

Index Terms: *SLAM, particle filter (PF), robot trajectory, mapping, environment.*

1. Introduction

SLAM is a widely used technique in the world of robotics for localizing a loving robot and mapping an environment simultaneously and it has seen significant improvements over the past decades. In the present world, it has numerous applications such as, in autonomous driving, in self-localization of humans in indoor or outdoor environments, personal robots for homes, cafes, for creating sparse or dense maps of an unknown environment, etc. In our specific situation, we are given data from multiple sensors which includes IMU data (that provides linear and angular acceleration), encoder data attached to the robot wheel (that provides wheel velocity). An obvious question might be - we already have robot velocity and acceleration data using which we can compute robot location at every time steps and therefore its entire trajectory. Then why do we need particle-filter SLAM (PF-SLAM) ?? The caveat is that all these measurements have noise and therefore without any estimation technique, the computed trajectory might have large drift and noise. Therefore, particle-filter SLAM alleviates this issue as it uses Bayes theorem to combine observations from multiple sensors.

In this project, we are given multi-modal data from multiple sensors - encoder, IMU, 2-D LIDAR scanner - that are attached to a differential-drive robot, and it is moving in an environment. Using this data, we are required to estimate the trajectory of the robot and simultaneously map the environment. We perform particle-filter-based SLAM approach to achieve this objective. The particles are referred to robot's pose hypothesis in the world frame. The particle filter has broadly these steps:

- We initialize the particle poses to $(x, y, \theta) = (0, 0, 0)$ and assign equal weights to each particle.
- We predict particle's poses at the next time-step using differential-drive model of the robot. We add random noise to each particle's linear velocity and angular velocity obtained from the encoder and IMU sensor respectively.
- Using these predicted poses, we update the particle weights by computing the correlation between the LIDAR observations based on each of these predicted particle's poses with the current environment map. Higher the correlation between the predicted particle pose based observations and current map, higher would be the assigned weight to that particle.
- We update the robot trajectory at current time step using the best particle pose which we have taken as the particle whose LIDAR observations result in the highest correlation with current map. This step can be thought of as localization.
- Finally, we update our map using the LIDAR observations transformed to the world coordinates using the robot location at current timestep.

2. Problem Formulation

2.1. Particle-Filter SLAM

Some common notation used in this report:
T - total number of timesteps

$x_{0:T}$ - trajectory of the robot
$z_{0:T}$ - observations from the LIDAR scanner
$u_{0:T-1} = [v_{0:T-1}, \omega_{0:T-1}]$ - control input to the robot
$w_{0:T}$ - motion noise
$v_{0:T}$ - observation noise
$x_{t+1} = f_d(x_t, u_t, w_t)$ - motion model of the differential drive robot. In discrete time, the prediction of the next state can be represented using -

$$x_{t+1} = f_d(x_t, u_t) = x_t + \tau_t \begin{bmatrix} v_t \text{sinc}(\frac{\omega_t \tau_t}{2}) \cos(\theta_t + \frac{\omega_t \tau_t}{2}) \\ v_t \text{sinc}(\frac{\omega_t \tau_t}{2}) \sin(\theta_t + \frac{\omega_t \tau_t}{2}) \\ \omega_t \end{bmatrix} \quad (1)$$

$z_t = h(x_t, v_t)$ - observation model
m - occupancy grid map of the environment generated using the LIDAR range observations
$\{\mu[k], \alpha[k]\} \quad k = 1, 2, ...N$ - particle hypothesis states (2D x-y position and orientation θ) and particle weights

Starting from an initial robot state $(x, y, \theta) = (0, 0, 0)$ and given the observations $z_{0:T}$ from the LIDAR scanner and control inputs $u_{0:T-1}$ measured using the encoder and IMU sensor attached to the robot, predict the robot trajectory $x_{0:T}$ based on robot's motion model and simultaneously create environment map using the LIDAR range measurements and observation model.

PF Prediction Problem: Given prior robot state distribution at any time step t given by $p_{t|t}(x_t)$ and control input $u_t = [v_t, \omega_t]^T$, we need to estimate the distribution of robot states for the next time step $p_{t+1|t}(x_{t+1}) = p(x_{t+1}|z_{0:T}, u_{0:T})$ and thus predict the robot motion using the differential-drive motion model in equation 1. In particle filter, we represent this distribution using a PMF over discrete states.

$$p_{t|t}(x_t) = \sum_{i=1}^{N} \alpha_{t|t}[k] \delta(x_t - \mu_{t|t}[k] \quad (2)$$

PF Update Problem: Given a probability distribution of states for next time step $p_{t+1|t}(x_{t+1})$ and a range observation z_{t+1} from LIDAR scanner, we need to use the observation model p_h to obtain the updated distribution of robot states $p_{t+1|t+1}(x_{t+1}) = p(x_{t+1}|z_{0:t+1}, u_{0:t})$. Specifically, we need to estimate the PMF weights $\alpha_{t+1|t}[k]$ for each particle given the updated particle states after every predict step.

PF Mapping Problem: Given observations $z_{0:t}$ from the LIDAR scanner and robot trajectory $x_{0:t}$, we need to compute a 2D occupancy grid map of the environment m_t that can represent free space, occupied space (obstacles) and not visited space. Mathematically, estimate $p(m_t|z_{0:t}, x_{0:t}, m_{0:t-1})$ at every timestep t.

2.2. Texture Mapping Problem

Given RGBD (RGB and disparity) images from the Kinect RGBD camera $I_{0:T}, D_{0:T}$ attached to the robot and a robot trajectory $x_{0:T}$ estimated using the particle filter SLAM, we need to estimate a 2D color map of the floor surface on which the robot is moving.

3. Technical Approach

In this section, we describe in detail our approach to solving the Particle Filter SLAM (PF-SLAM) problem given multimodal data.

3.1. Data Processing:

The data from IMU, encoder and LIDAR scanner are not time synchronized. In addition, their measurement frequencies are different. To process them together, we first matched the timestamps of IMU and LIDAR data with respect to the encoder data. For each encoder timestamp, we find the closest timestamp in the IMU and LIDAR timestamps and store their indices. We access IMU and LIDAR data corresponding to these indices and use them for all future purposes. Therefore, after this matching process, the IMU and LIDAR data have the same number of samples as the encoder data with closest matching timestamps.

3.2. LIDAR data conversion:

The LIDAR data is a scan range measurement (in meters) over 1081 scan angles (for dataset 20) ranging from $-135°$ to $135°$ (for dataset 20). We first convert these range measurements from distances in meters to cartesian coordinates (x, y, z) in the LIDAR frame. The LIDAR origin is taken as the LIDAR geometric center with axes direction matching the robot coordinate frame. Thus, in the robot frame, the LIDAR origin is at $(x, y, z) = [133.23, 0, 514.35]$ mm. Next, we convert these coordinates from the LIDAR frame to the robot body frame. The robot origin is chosen as the geometric center of the robot body in X-Y horizontal plane and Z=0 is chosen to be the horizontal plane passing through the wheel center. The robot axes directions are chosen as follows: X - forward, Y - left, and Z - up. The same axes direction is also chosen for the LIDAR frame and the world frame as well. The world frame origin is the robot starting position at $t = 0$. These cartesian coordinates transformed from the LIDAR frame to the robot frame represent 3D points where the rays emitted from the LIDAR scanner hits an obstacle. Due to measurement noise in the LIDAR scanner, there is noise present in these cartesian coordinates of the obstacles as well.

3.3. IMU data processing

The IMU sensor provides linear acceleration $a_0 : T$, and linear angular velocity (roll, pitch, and yaw angles). Since

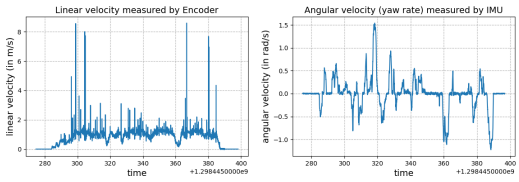

Figure 1. Figure (on the right) shows the raw angular velocity (yaw) with time measured using IMU sensor

the robot is moving in a 2D horizontal plane, we only consider the yaw angle as relevant data to predict robot motion. Therefore, we have ignored the other measurements. We also do not use linear acceleration in our computation. We applied a low-pass Butterworth filter with a cut-off frequency of 10 Hz to reduce measurement noise in the yaw rate.

3.4. Encoder data processing

The encoder provides rotation counts for each of the 4 wheels at 40 Hz in this format - [Forward Right (FR), Forward Left (FL), Rear Right (RR), Rear Left (RL)]. We use the following equations to compute the average velocities for the right v_R and left wheels v_L and average them to get the robot linear velocity $v = \frac{v_R+v_L}{2}$ where v_R and v_L are given by -

$$v_R = \frac{0.5(FR+RR)\pi d}{360\tau_t} \quad , \quad v_L = \frac{0.5(FL+RL)\pi d}{360\tau_t} \tag{3}$$

where d is wheel diameter = 0.254 m, τ_t is the time difference between successive encoder measurements provided in the data.

3.5. Map Generation

For generating an occupancy grid map, we use LIDAR range scan data. As described earlier, we convert obstacle range data along 1081 scan angles to cartesian coordinates in the LIDAR frame and then to the robot frame (RF). Next, based on robot motion, we convert these coordinates in the robot frame to the world frame (WF) using robot pose estimated at each timestep. Therefore, if the robot SE(3) pose in the world frame at time t is $\begin{bmatrix} {}_{WF}R_{RF}(t) & {}_{WF}p_{RF}(t) \\ 0^T & 1 \end{bmatrix}$ then we convert the LIDAR observation synchronized at timestep t from the robot frame to the world frame using: $X_{WF} = {}_{WF}R_{RF}(t)X_{RF} + {}_{WF}p_{RF}(t)$. Once we have the LIDAR obstacle coordinates in the world frame, these represent the 3D points measured as obstacles by the LIDAR. We create a 2D occupancy grid map that can represent physical distances from -30m to 30m in the world frame X-Y axes with map resolution being 0.05m. Thus, the map size is 1201×1201. We use the Breeshenham Rasterization algorithm to trace all the coordinate points in the map along any ray starting from LIDAR and ending at an obstacle. The breshenham algorithm takes the start and end points of a ray and provides all the intermediate points along the ray.

We maintain a log-odds map λ_t of size 1201×1201 in which we add $\log(1/4)$ to all map cells along any ray that are free. Otherwise, we add $\log(4)$ to the endpoint along a ray that marks an obstacle or an occupied cell. We do this process for all 1081 scan angles one-by-one and update the log-odds map λ_t by adding $log(1/4)$ for all the free cells and $\log 4$ to all occupied cells in the map. Then we on to the next trajectory step and repeat this process. At the end of the trajectory, we bound our log-odds map to be strictly between some $\lambda_{min} = -100$ and $\lambda_{max} = 50$ (hyperparameters). Then we convert this 1201×1201 log-odds map to probability values using the softmax operator representing the probability of a map cell being free $P(m[i,j] = \text{free}) = \frac{1}{1+\exp(\lambda_t)}$. Finally, we assign all cells with probability > 0.5 as white free cells (1), all cells with probability < 0.5 as black occupied cells (0), and all cells with probability = 0.5 as gray (0.5) or not visited cell.

3.6. Particle-Filter SLAM

The particle filter algorithm uses particles $\mu[k] \in R^3$ that represents a potential robot state (2D position in X-Y and orientation θ) and uses discrete weights $\alpha[k]$ for $k = 1, 2, ..., N$ to represent the probability distribution $p_{t|t}$ and $p_{t+1|t}$ as below.

$$p_{t|t}(x_t) = \sum_{k=1}^{N} \alpha_{t|t}[k]\delta(x_t - \mu_{t|t}[k]) \tag{4}$$

$$p_{t+1|t}(x_{t+1}) = \sum_{k=1}^{N} \alpha_{t+1|t}[k]\delta(x_{t+1} - \mu_{t+1|t}[k]) \tag{5}$$

The particle filter algorithm is divided into two major steps: Prediction and Update. Initially, we initialize all particles state to $(x, y, \theta) = (0, 0, 0)$. In the prediction stage, we predict the particle state at the next timestep (t+1) given the current state using the differential-drive motion model as described in equation 1. We add some random noise to the measured linear and angular velocity (from encoder and IMU yaw) to get random variation in each particle's state. Specifically, for every particle, we sample from $N(0, \sigma_v^2)$ and $N(0, \sigma_{\omega^2})$. We apply this to each particle $\mu_{t|t}[k]$ in order to obtain $\mu_{t+1|t}[k] \sim p_f(.|\mu_{t|t}[k], \mu_t)$ and then we approximate the probability distribution at next time step

using below equation:

$$p_{t+1|t}(x_{t+1}) = \sum_{k=1}^{N} \alpha_{t|t}[k] p_f(x_{t+1}|\mu_{t|t}[k], \mu_t)$$
$$= \sum_{k=1}^{N} \alpha_{t|t}[k] \delta(x_{t+1} - \mu_{t+1|t}[k])$$

Thus, the prediction step only changes the particle locations and not their weights.

Next, we move on to the update step where we use the particle states computed in the prediction step x_{t+1} to update the particle weights $\alpha[k]$. We use the LIDAR observations to compute correlation between each particle and current map, and use it to update weights. We transform the LIDAR scan from robot frame to world frame using each particle's pose estimated from the prediction step. Then we compute the map correlation for each particle. We add some noise perturbation to the particle's state by considering a $m \times m$ (m = 9 in our case) grid around each particle's XY position and assume the particle to be in each of those grids. For each of those m^2 particle state perturbations, we transform the LIDAR obstacle coordinates from robot frame to world frame. Then we compute the correlation between the current occupancy map and each of the scans by just adding the occupancy map in each of those m^2 locations across all scan angles. The idea is that the best particle state among all particles should most agree with the observations and existing map. Therefore, for each particle, we get a $m \times m$ correlation map, from where we take the highest correlation value. We repeat this process for each particle.

Finally, we update the particle weights proportional to each particle's correlation value with the map using the below equations -

$$\alpha_{t+1|t+1}[k] = \frac{\text{corr}(y_{t+1}[k], m_t)\alpha_{t+1|t}[k]}{\sum_{k=1}^{N} \text{corr}(y_{t+1}[k], m_t)\alpha_{t+1|t}[k]} \quad (6)$$

After the update step, we choose the particle with the highest weight as the robot's optimal state at current time step. Finally, we update the occupancy map using this optimal state using the process described in sec 3.5. After every update step, we resample the particles if the number of effective particles N_{eff} is less than a threshold (we took it to be N/10 as given in the slides). For resampling, we do weighted sampling from a pool of indices 1,2,...,N where weights are the particle weights and N is the number of particles. For the chosen index j, we choose $\mu_{t+1|t+1}[j]$ as one of the particle states. Then we repeat this process N times to fully resample all particle states. After resampling, we update all particle weights to 1/N.

4. Texture Mapping

In this problem, we are required to map the texture of the floor surface given RGBD data from KINECT sensor and optimal robot trajectory estimated using PF-SLAM approach. We are given RGB images and disparity images captured from KINECT camera attached to the robot. Once again, the RGB and disparity images are captured at different time instances and not time synchronized. To synchronize them, for each RGB timestamp/image, we find the closest timestamp in the disparity images and match them together.

Next, due to x-axis offset between RGB and depth camera (not in the same location inside KINECT sensor), there is a correspondence mismatch between RGB and disparity images. We find pixel correspondences between RGB and disparity images using the equations provided in the problem statement. That is, for each pixel (i,j) in the disparity image with value d, corresponding pixel location in the RGB image is given by -

$$dd = -0.00304d + 3.31 \quad (7)$$
$$rgbi = (526.37i + (-4.5 \times 1750.46)dd + 19276)/585.051 \quad (8)$$
$$rgbj = (526.37j + 16662)/585.051 \quad (9)$$

and corresponding depth value at pixel (i,j) is depth = $\frac{1.03}{dd}$ We also ignore invalid pixel locations in the above computed (rgbi, rgbj) which are outside the image bounds. Now, that we know pixel correspondences between RGB and depth image, we convert these pixel coordinates from pixel coordinates to camera optical frame using the below relation.

$$\begin{bmatrix} X_o \\ Y_o \\ Z_o \end{bmatrix} = K^{-1} \begin{bmatrix} rgbi \\ rgbj \\ 1 \end{bmatrix} * depth \quad (10)$$

where K is the camera intrinsic matrix provided in problem statement. Next, we convert from each pixel location in camera optical frame to regular frame using the below relation -

$$\begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} X_o \\ Y_o \\ Z_o \end{bmatrix} \quad (11)$$

Next we transform these pixel coordinates from regular frame to robot body frame using KINECT sensor pose information in robot frame. We use the roll, pitch, and yaw angle of the Kinect sensor to get the rotation matrix and the position of Kinect wrt the robot origin to get Kinect SE(3) pose in robot frame. The KINECT origin position in robot

frame is ${}_{RF}p_K = [167.66, 0, 380.01]$ mm.

$$\begin{bmatrix} X_{RF} \\ Y_{RF} \\ Z_{RF} \end{bmatrix} = {}_{RF}\, R_K \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} + {}_{RF}\, p_K \tag{12}$$

Finally, we now have for each RGB pixel, the corresponding 3D point in the robot frame. We convert these to world frame using the robot's optimal trajectory estimated using PF-SLAM algorithm.

$$\begin{bmatrix} X_{WF} \\ Y_{WF} \\ Z_{WF} \end{bmatrix} = {}_{WF}\, R_{RF}(t) \begin{bmatrix} X_{RF} \\ Y_{RF} \\ Z_{RF} \end{bmatrix} + {}_{WF}\, p_{RF}(t) \tag{13}$$

Next, we only consider those 3D points whose Z-coordinates matches the floor surface. In my case, since the Z=0 is taken to be horizontal plane passing through robot wheel center, we only consider points whose $0.122 \leq Z_{WF} \leq 0.132$ m. We taken these points and convert them to map coordinates similar to described in 3.5. Finally, we put the RGB intensity value at pixel (rgbi, rgbj) in RGB image to the map coordinates computed using $[X_{WF}, Y_{WF}]$. Thus, we have our resulting floor texture map.

5. Conclusion

- In most experiments, I used number of particles to be 100 and ran the particle filter for 1 trajectory step.
- We get acceptable results for low noise standard deviation. For $\sigma_v = 0.01, 0.1, 0.5$ and $\sigma_\omega = 0.01, 0.05$.
- Increasing noise standard deviation for angular velocity component $\sigma_\omega \geq 0.1$ and for linear velocity $\sigma_v \geq 1$ separates the particles too much from the current robot pose and results in a very inflated environment map.
- We tried increasing the number of particles which resulted in some improvements in the generated map but it took almost double the amount of time as compared to with N=100.

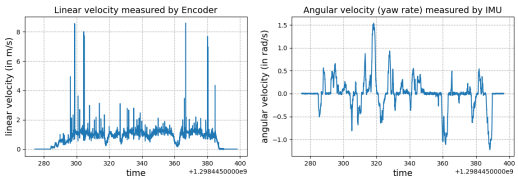

Figure 2. Figure showing the robot linear (computed using encoder) and angular velocity (from IMU) with time

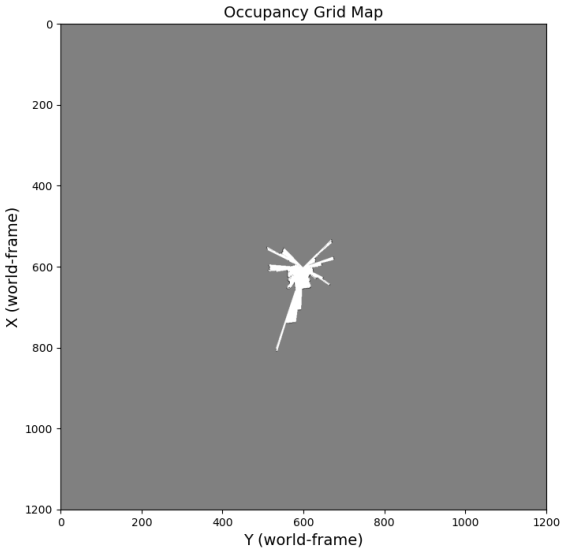

Figure 3. Figure showing map of the initial (robot pose = (0,0,0)) LIDAR scan measurement where the center of the grid map corresponds to the XY-origin of world frame. White corresponds to free space, black means occupied space, and gray represents space not visited yet

6. Results - Dataset 20

figures/ds20/pfslam/map_pf_N1000_stdv0.5_stdw

Figure 55. Figure showing final map after PF-SLAM with 1000 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$

Figure 4. Figure showing the full map of the environment generated using (almost) all LIDAR scan measurements transformed to world frame using robot trajectory. The robot trajectory (shown in red) was generated using the differential-drive motion model with 0 noise

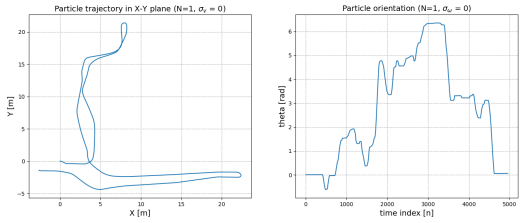

Figure 5. Figure (Left) showing a particle trajectory in the X-Y plane with 0 noise added to linear and angular velocity $\sigma_v = 0$, $\sigma_\omega = 0$. (Right) shows particle orientation variation with time

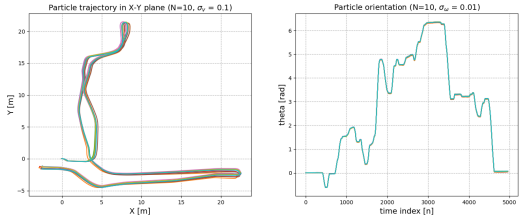

Figure 6. Figure (Left) showing a 10 particle trajectory in the X-Y plane with $\sigma_v = 0.1$, $\sigma_\omega = 0.01$. (Right) shows particle orientation variation with time

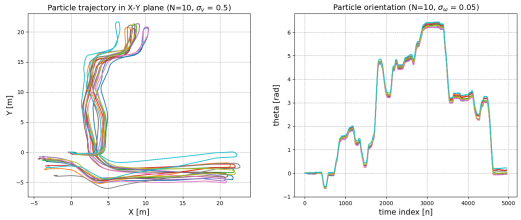

Figure 7. Figure (Left) showing a 10 particle trajectory in the X-Y plane with $\sigma_v = 0.5$, $\sigma_\omega = 0.05$. (Right) shows particle orientation variation with time

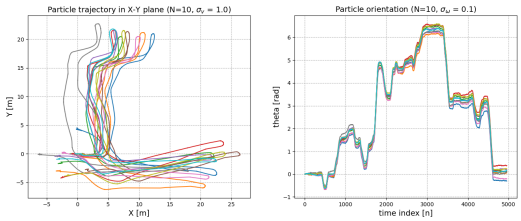

Figure 8. Figure (Left) showing a 10 particle trajectory in the X-Y plane with $\sigma_v = 1.0$, $\sigma_\omega = 0.1$. (Right) shows particle orientation variation with time

Figure 9. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 500

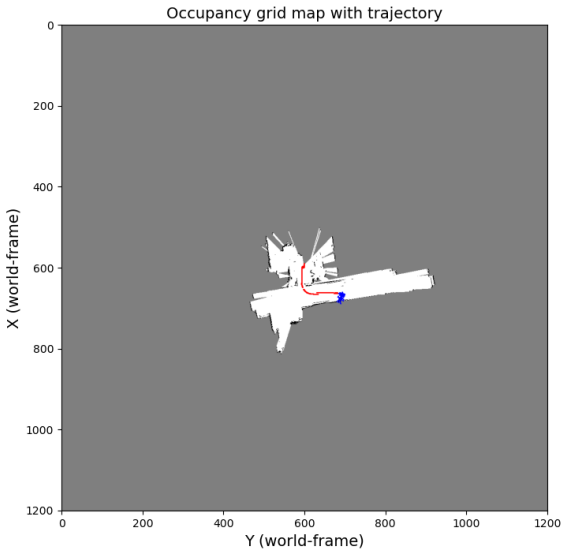

Figure 10. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 1000

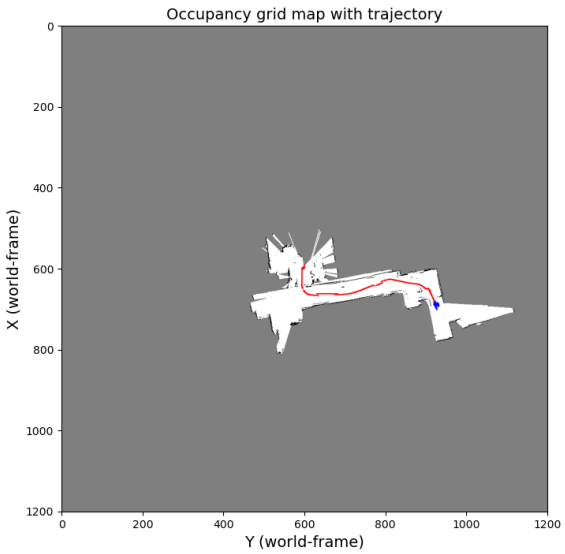

Figure 11. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 1500

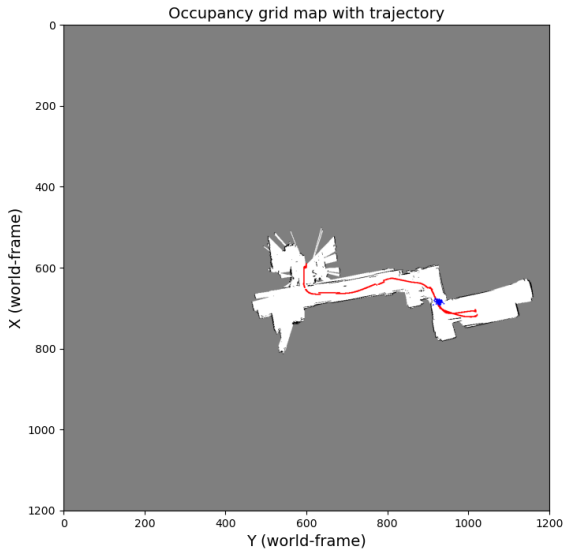

Figure 12. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 2000

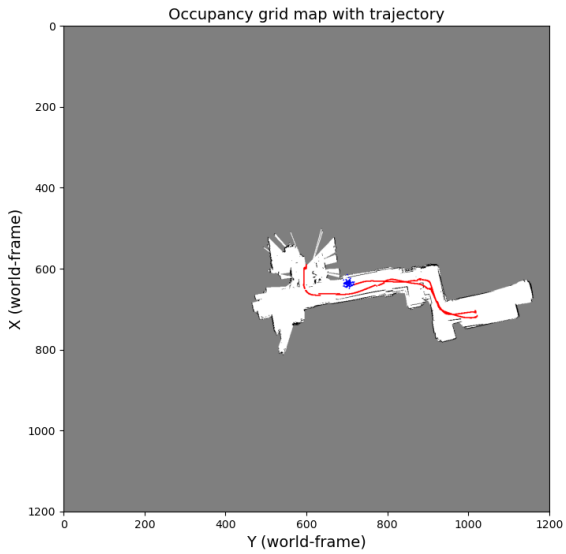

Figure 13. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 2500

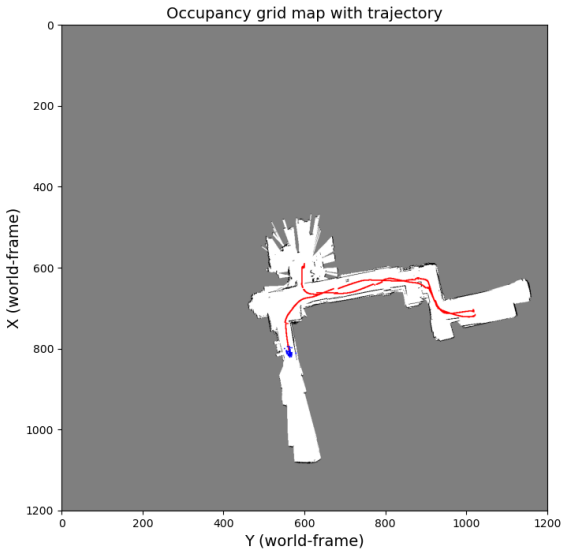

Figure 14. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 3000

Figure 15. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 3500

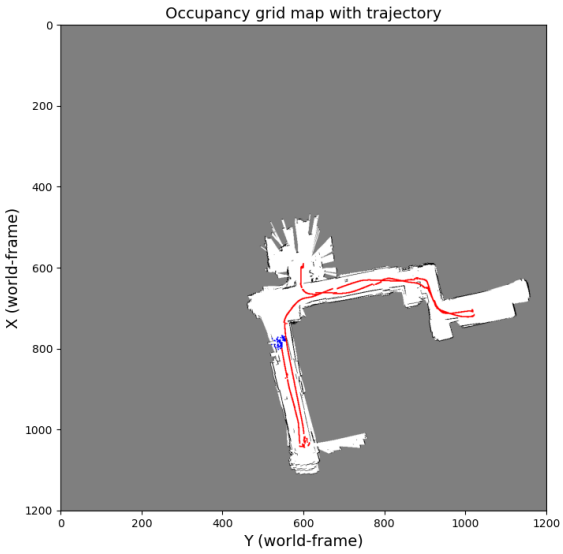

Figure 16. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 4000

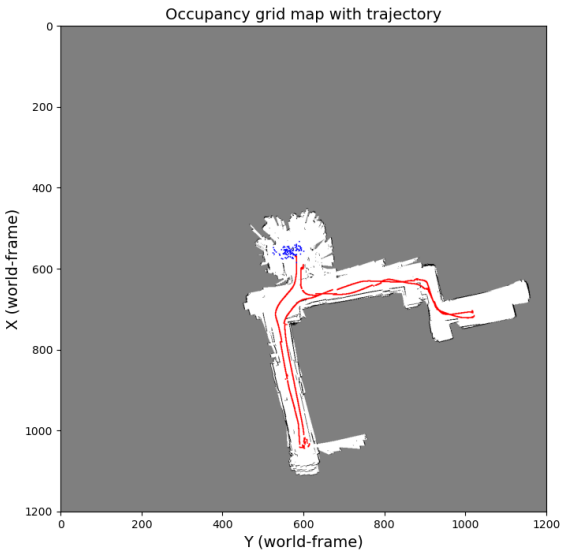

Figure 17. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 4500

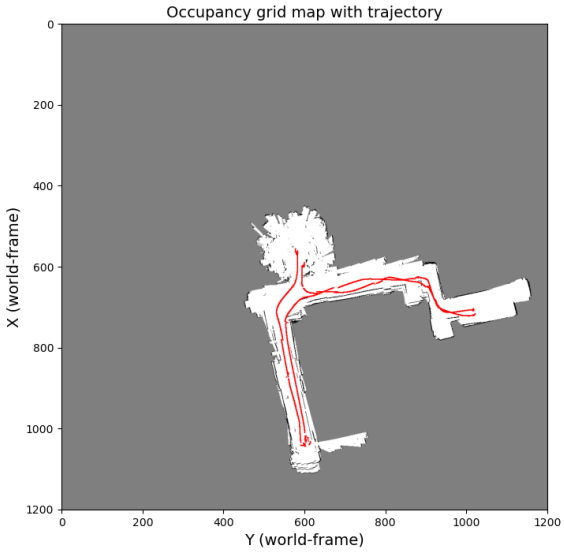

Figure 18. Figure showing final map after PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$

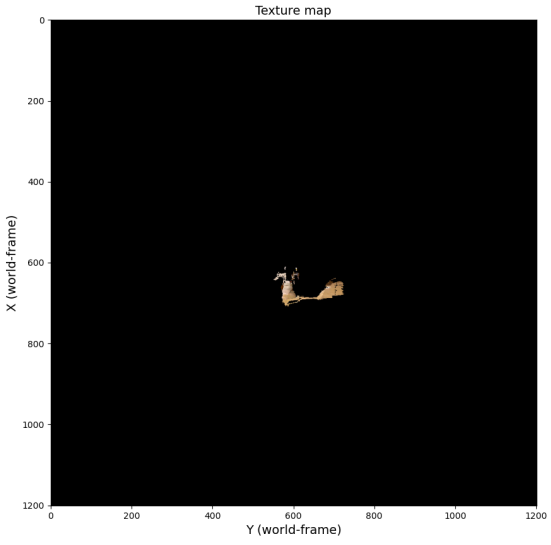

Figure 19. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, step 400

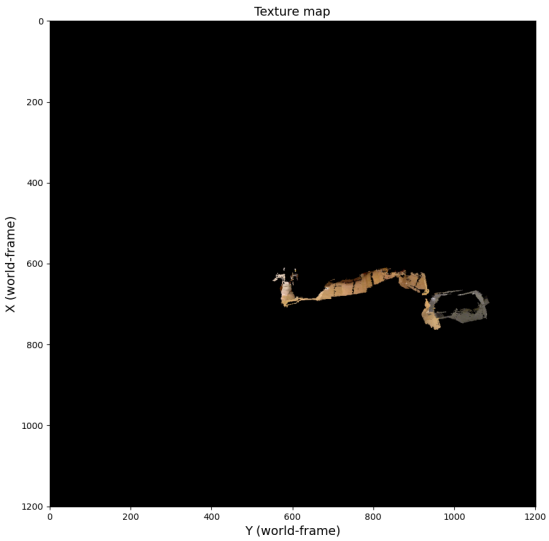

Figure 20. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, step 800

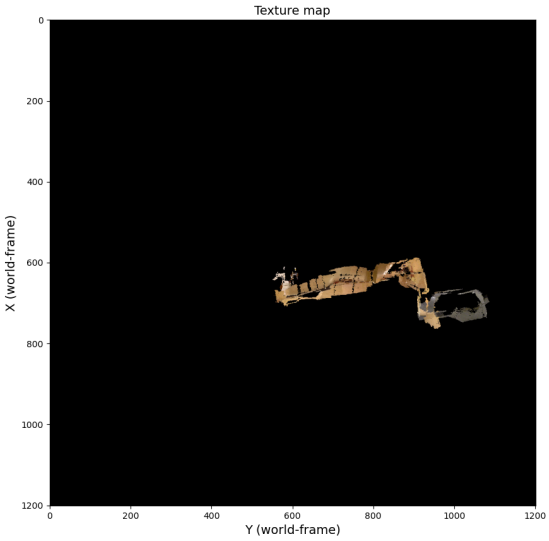

Figure 21. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, step 1200

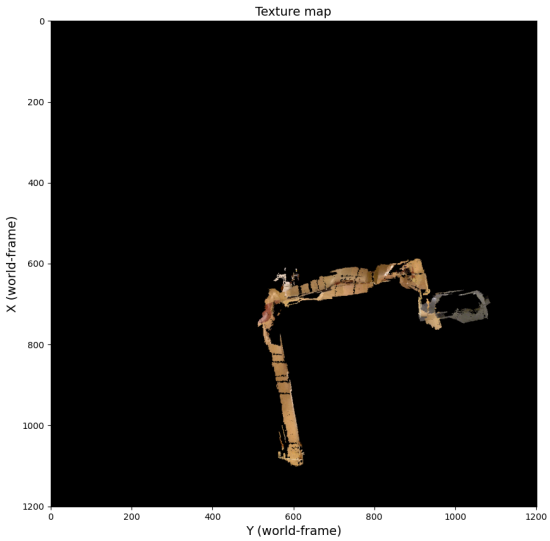

Figure 22. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, step 1600

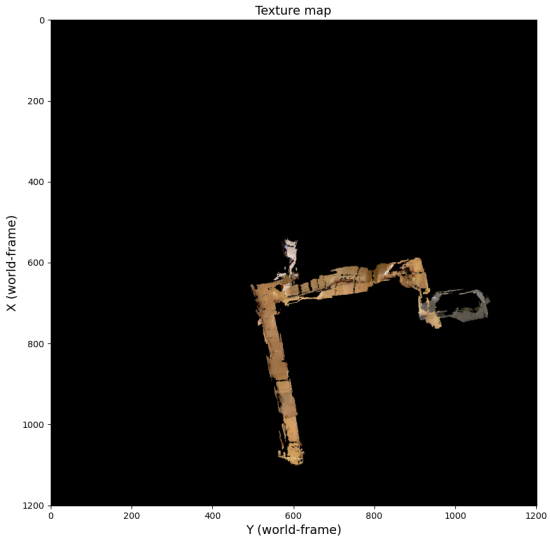

Figure 23. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, step 2000

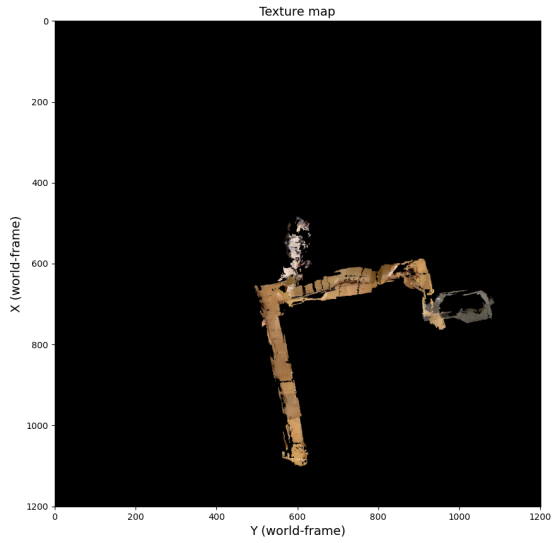

Figure 24. Figure showing final texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$

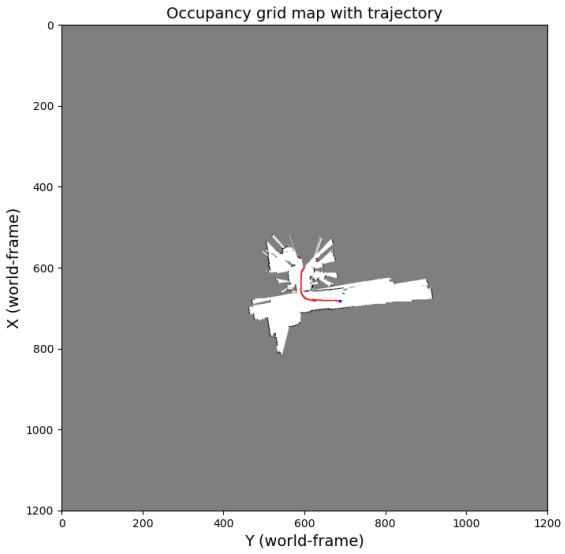

Figure 26. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.1, \sigma_\omega = 0.01$, trajectory step 1000

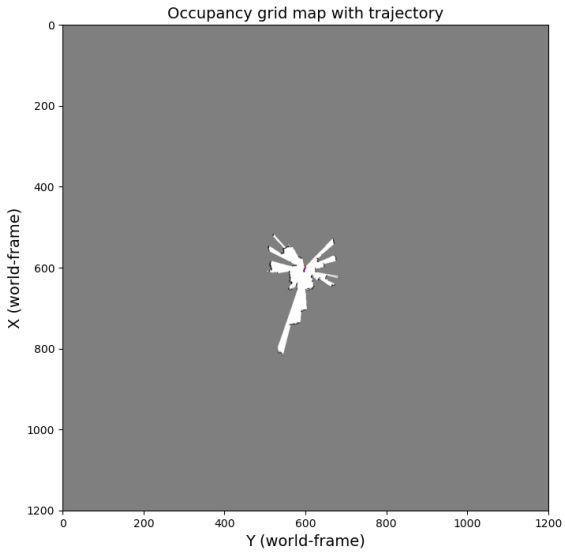

Figure 25. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.1, \sigma_\omega = 0.01$, trajectory step 500

Figure 27. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.1, \sigma_\omega = 0.01$, trajectory step 1500

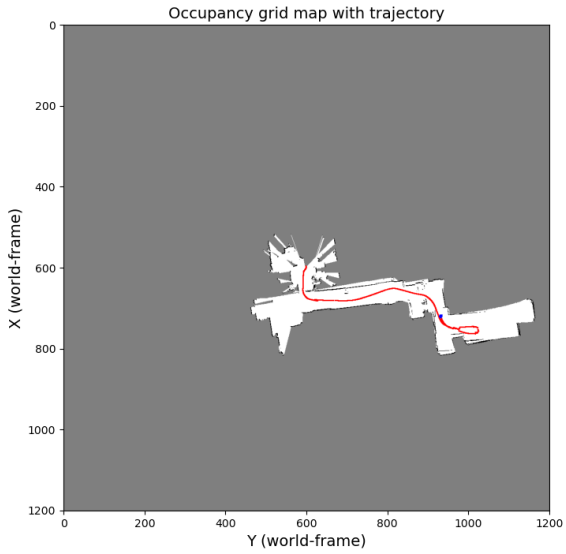

Figure 28. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.1, \sigma_\omega = 0.01$, trajectory step 2000

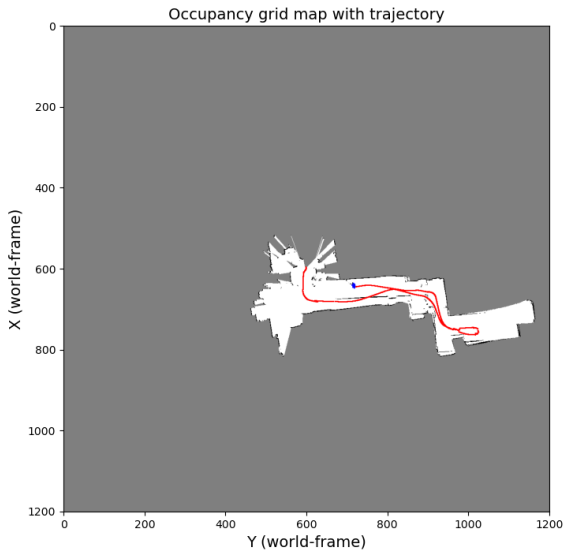

Figure 29. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.1, \sigma_\omega = 0.01$, trajectory step 2500

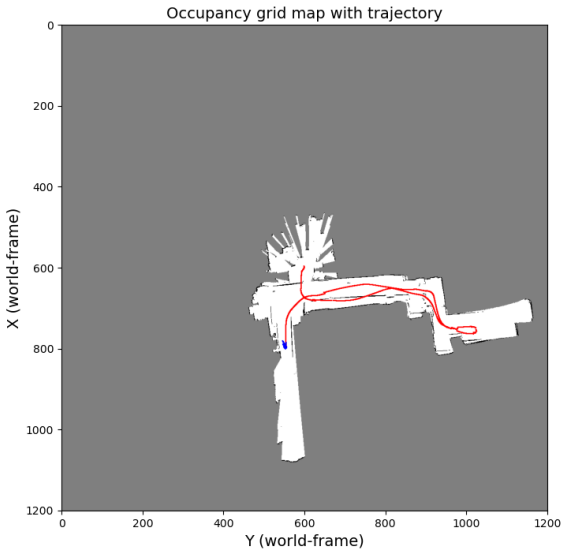

Figure 30. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.1, \sigma_\omega = 0.01$, trajectory step 3000

Figure 31. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.1, \sigma_\omega = 0.01$, trajectory step 3500

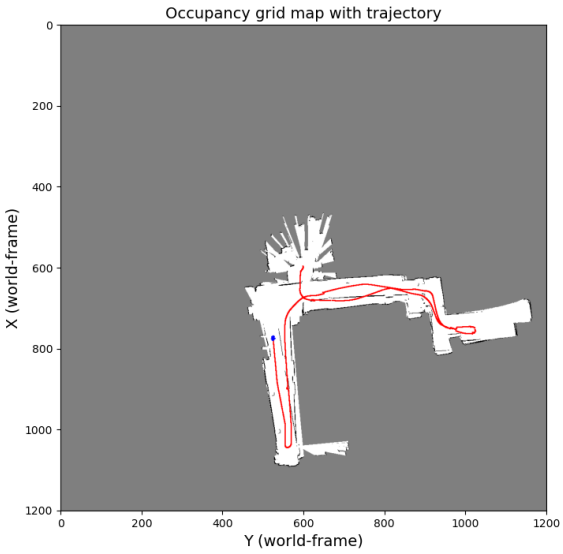

Figure 32. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.1, \sigma_\omega = 0.01$, trajectory step 4000

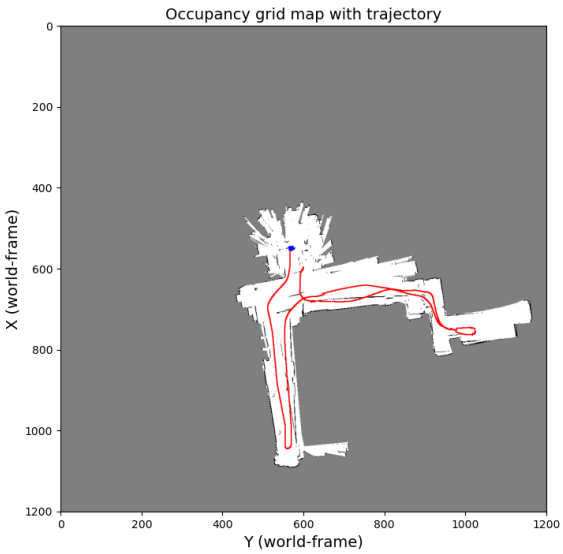

Figure 33. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.1, \sigma_\omega = 0.01$, trajectory step 4500

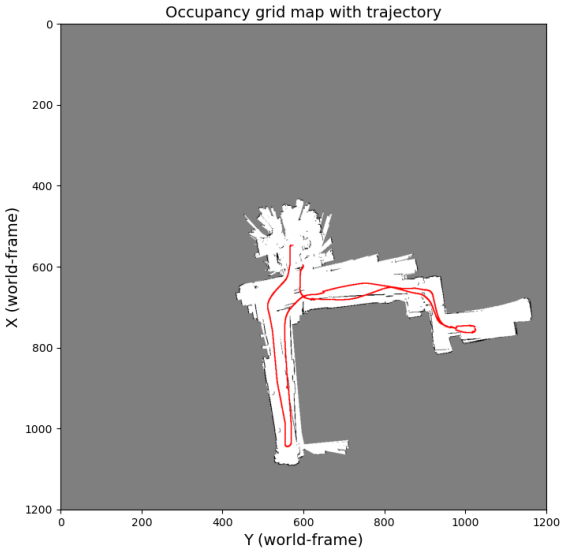

Figure 34. Figure showing final map after PF-SLAM with 100 particles, $\sigma_v = 0.1, \sigma_\omega = 0.01$

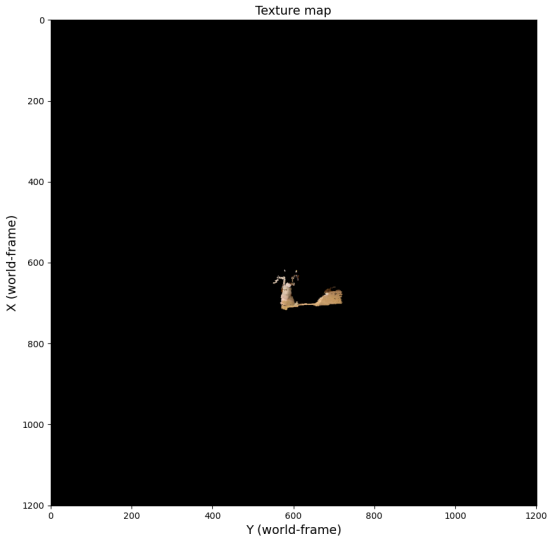

Figure 35. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.1, \sigma_\omega = 0.01$, step 400

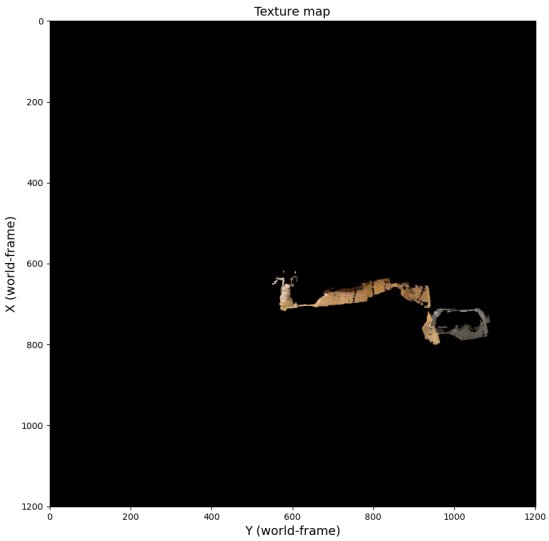

Figure 36. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.1, \sigma_\omega = 0.01$, step 800

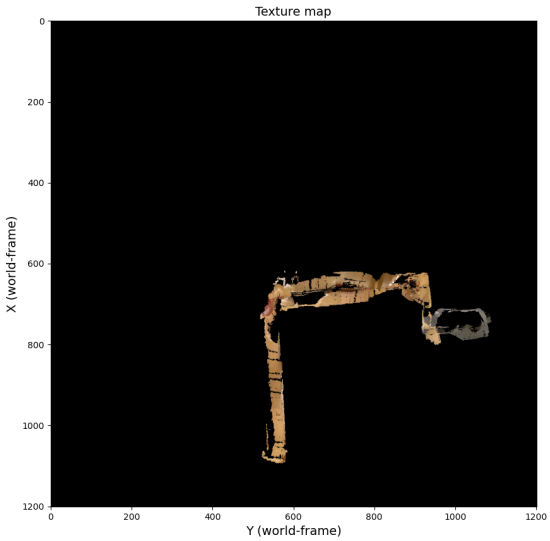

Figure 38. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.1, \sigma_\omega = 0.01$, step 1600

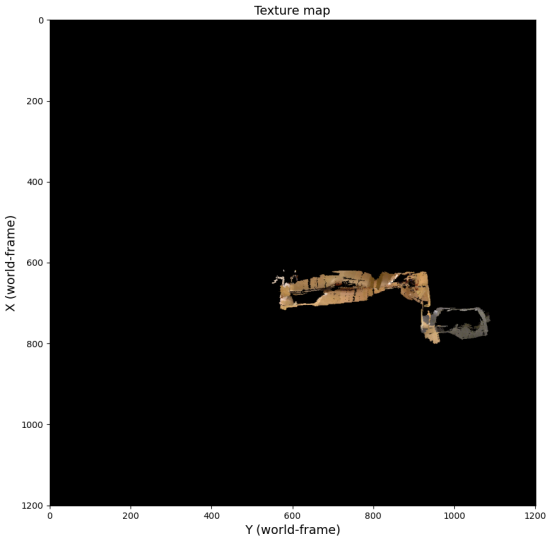

Figure 37. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.1, \sigma_\omega = 0.01$, step 1200

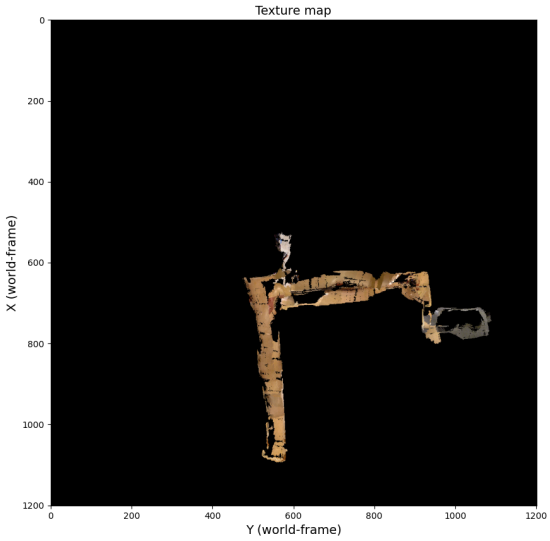

Figure 39. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.1, \sigma_\omega = 0.01$, step 2000

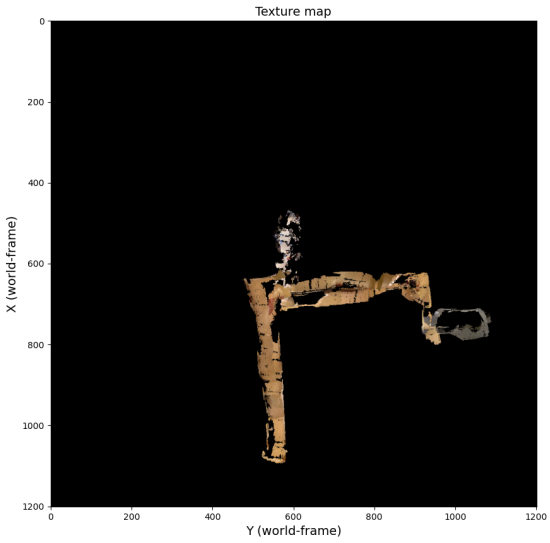

Figure 40. Figure showing final texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.1, \sigma_\omega = 0.01$

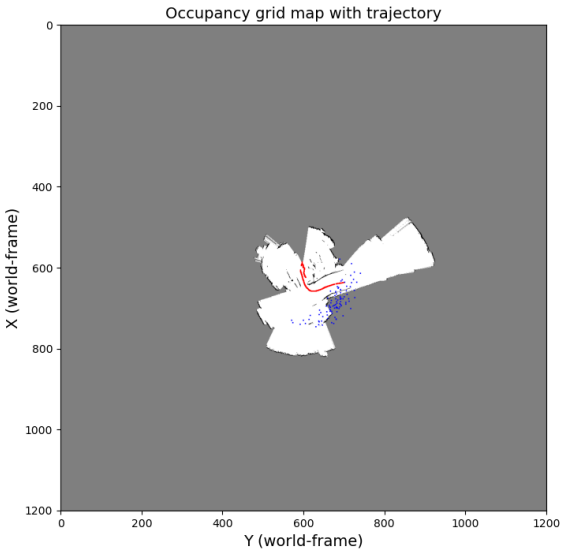

Figure 42. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 1.0, \sigma_\omega = 0.1$, trajectory step 1000

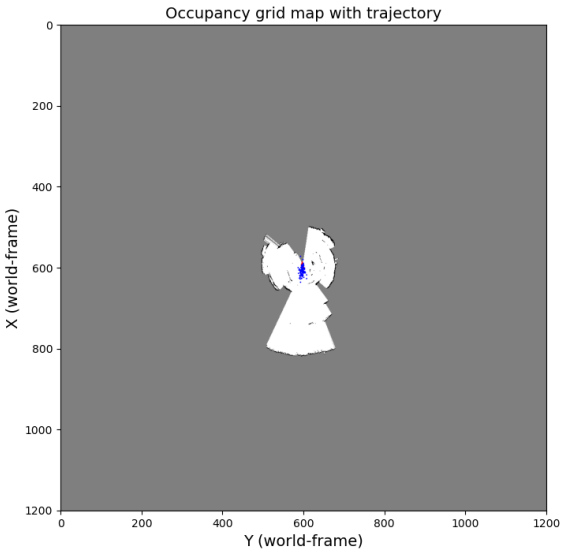

Figure 41. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 1.0, \sigma_\omega = 0.1$, trajectory step 500

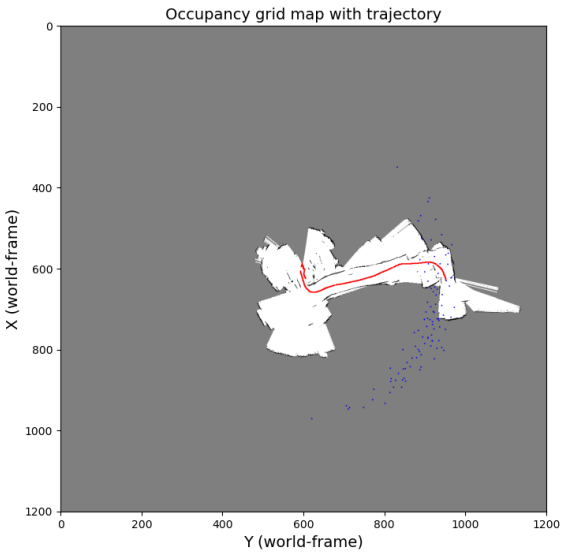

Figure 43. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 1.0, \sigma_\omega = 0.1$, trajectory step 1500

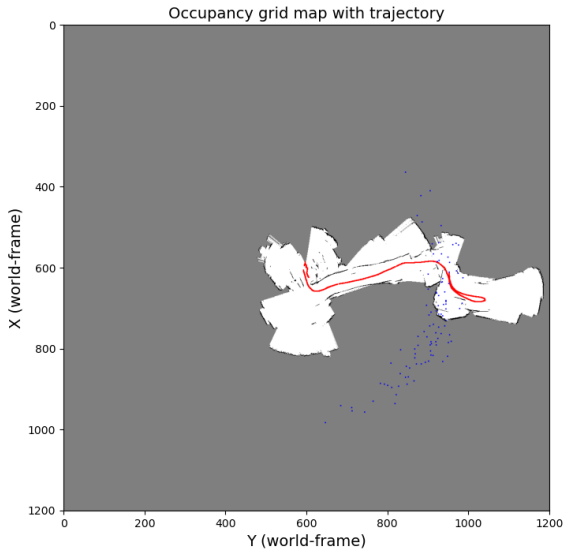

Figure 44. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 1.0, \sigma_\omega = 0.1$, trajectory step 2000

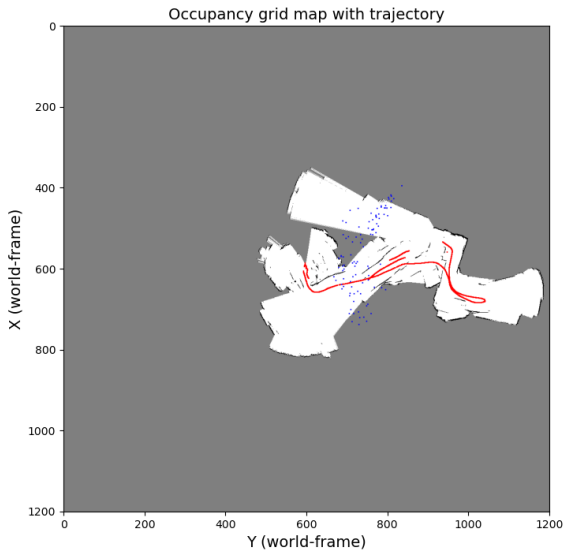

Figure 45. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 1.0, \sigma_\omega = 0.1$, trajectory step 2500

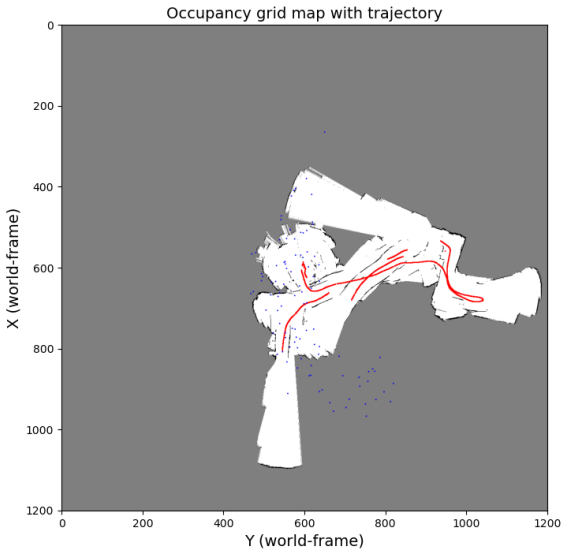

Figure 46. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 1.0, \sigma_\omega = 0.1$, trajectory step 3000

Figure 47. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 1.0, \sigma_\omega = 0.1$, trajectory step 3500

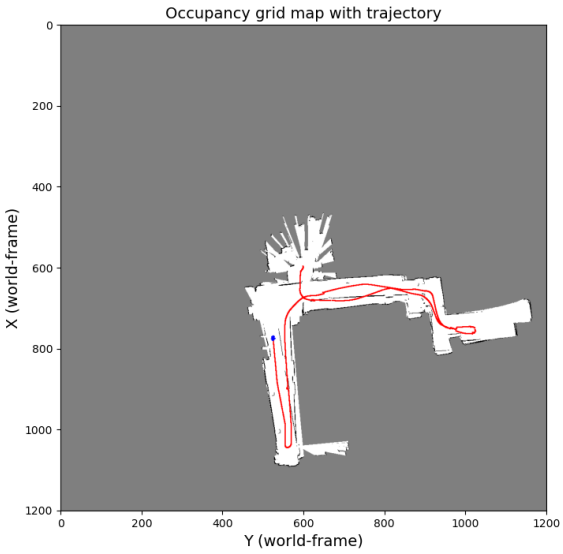

Figure 48. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 1.0, \sigma_\omega = 0.1$, trajectory step 4000

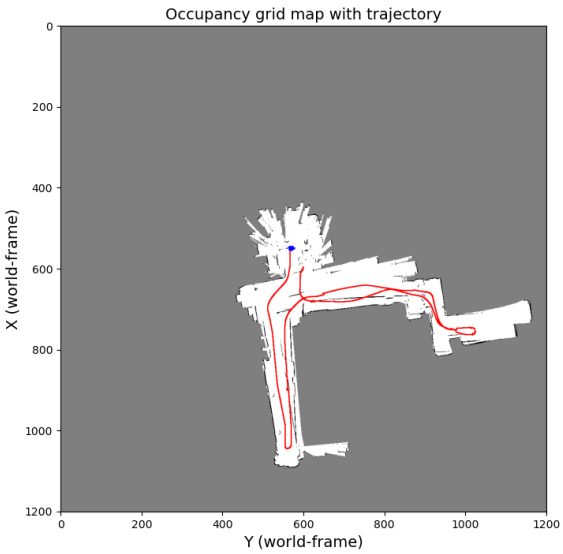

Figure 49. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 1.0, \sigma_\omega = 0.1$, trajectory step 4500

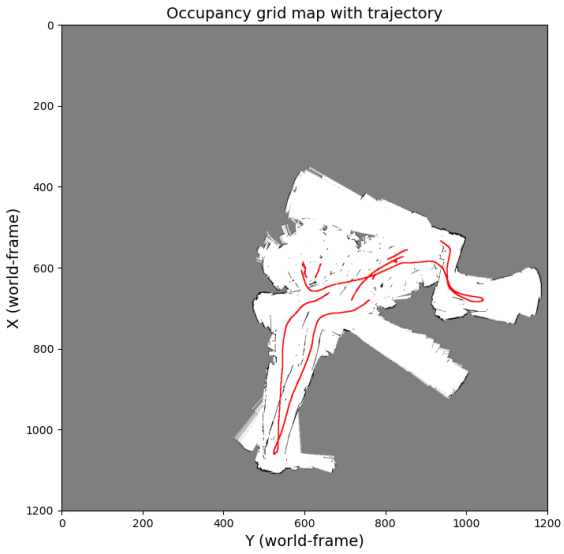

Figure 50. Figure showing final map after PF-SLAM with 100 particles, $\sigma_v = 1.0, \sigma_\omega = 0.1$

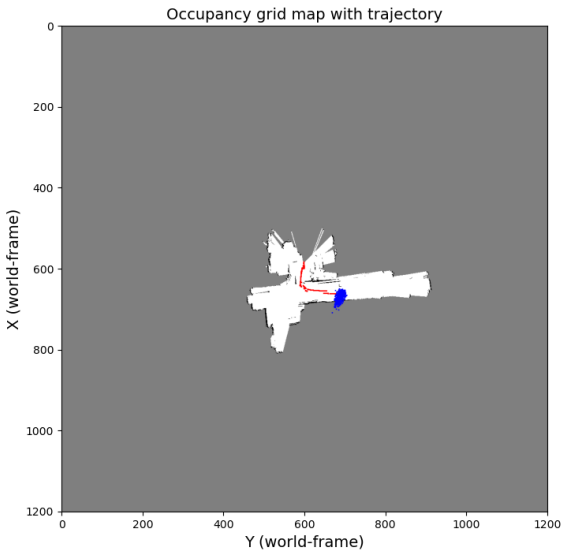

Figure 51. Figure showing map during PF-SLAM with 1000 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$, trajectory step 1000

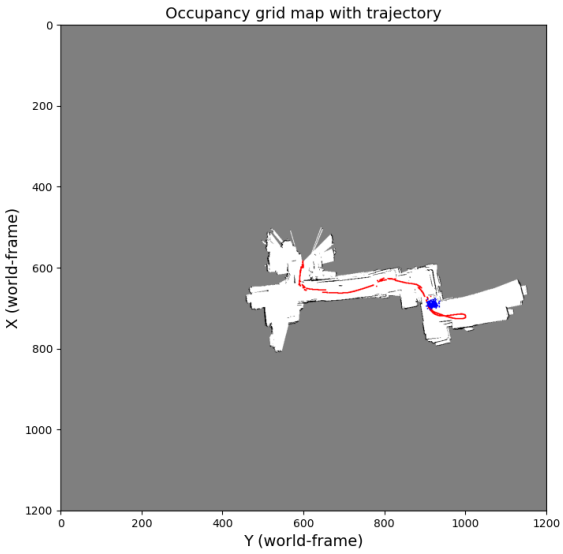

Figure 52. Figure showing map during PF-SLAM with 1000 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$, trajectory step 2000

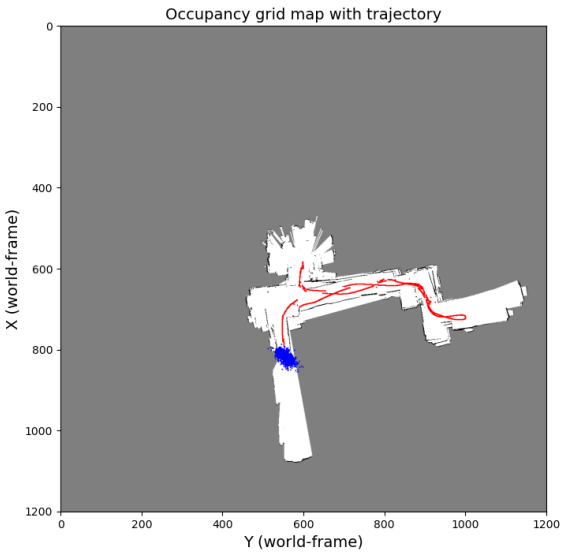

Figure 53. Figure showing map during PF-SLAM with 1000 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$, trajectory step 3000

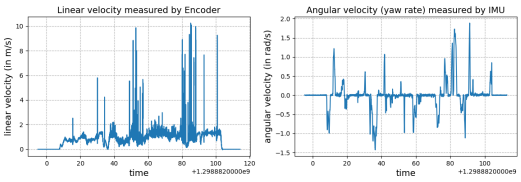

Figure 59. Figure showing the robot linear (computed using encoder) and angular velocity (from IMU) with time

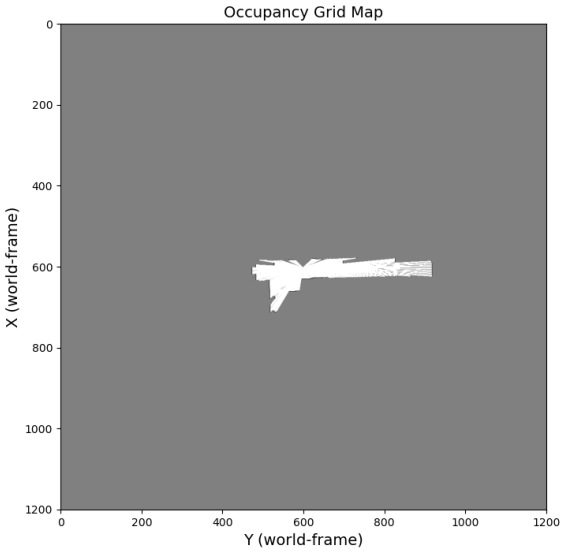

Figure 60. Figure showing map of the initial (robot pose = (0,0,0)) LIDAR scan measurement where the center of the grid map corresponds to the XY-origin of world frame. White corresponds to free space, black means occupied space, and gray represents space not visited yet

7. Results - Dataset 21

Figure 61. Figure showing the full map of the environment generated using (almost) all LIDAR scan measurements transformed to world frame using robot trajectory. The robot trajectory (shown in red) was generated using the differential-drive motion model with 0 noise

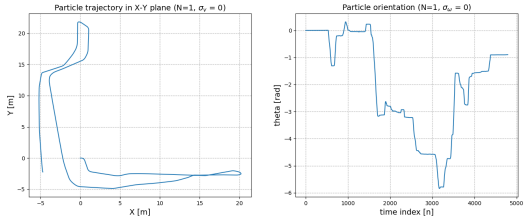

Figure 62. Figure (Left) showing a particle trajectory in the X-Y plane with 0 noise added to linear and angular velocity $\sigma_v = 0$, $\sigma_\omega = 0$. (Right) shows particle orientation variation with time

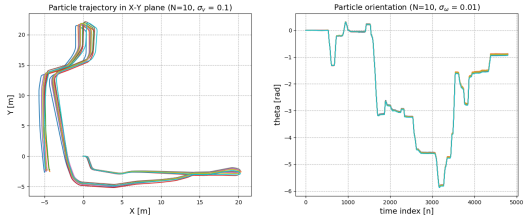

Figure 63. Figure (Left) showing a 10 particle trajectory in the X-Y plane with $\sigma_v = 0.1$, $\sigma_\omega = 0.01$. (Right) shows particle orientation variation with time

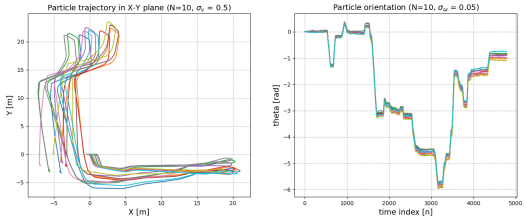

Figure 64. Figure (Left) showing a 10 particle trajectory in the X-Y plane with $\sigma_v = 0.5$, $\sigma_\omega = 0.05$. (Right) shows particle orientation variation with time

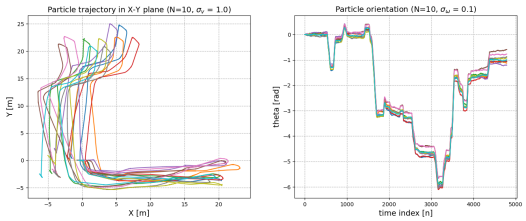

Figure 65. Figure (Left) showing a 10 particle trajectory in the X-Y plane with $\sigma_v = 1.0$, $\sigma_\omega = 0.1$. (Right) shows particle orientation variation with time

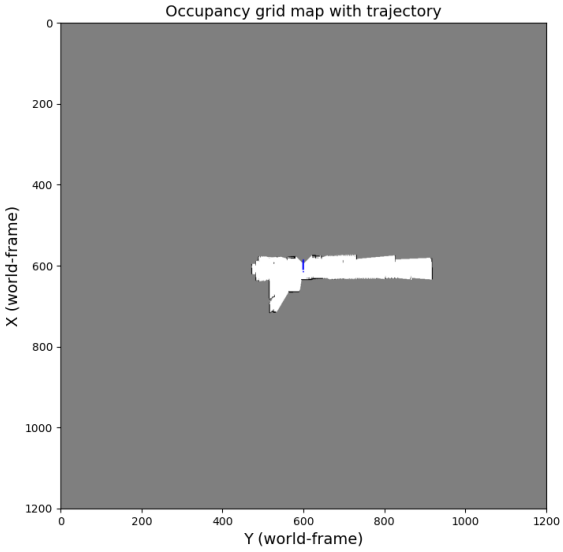

Figure 66. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 500

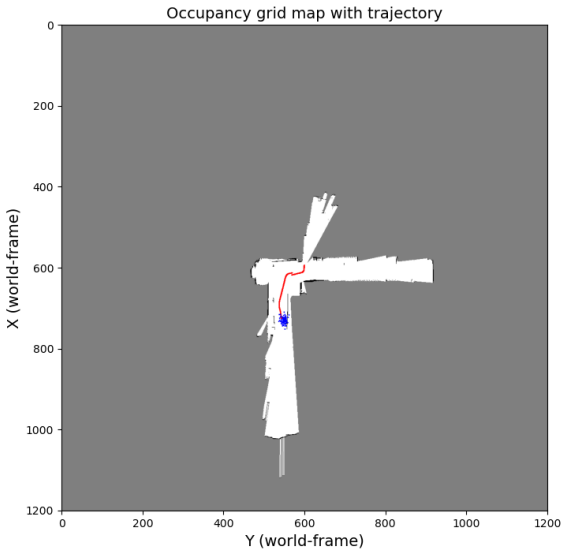

Figure 67. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 1000

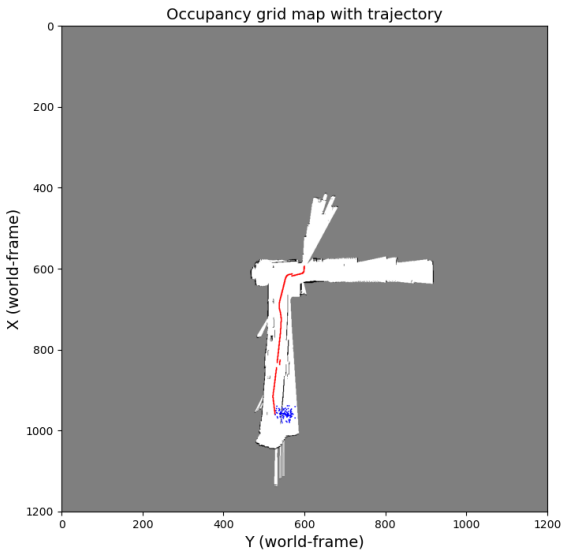

Figure 68. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 1500

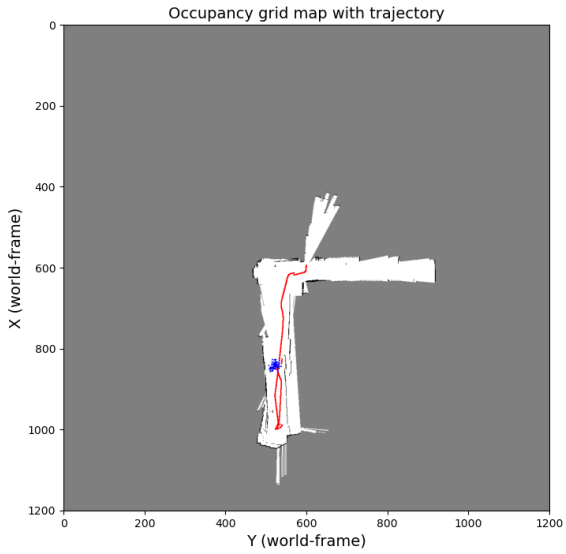

Figure 69. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 2000

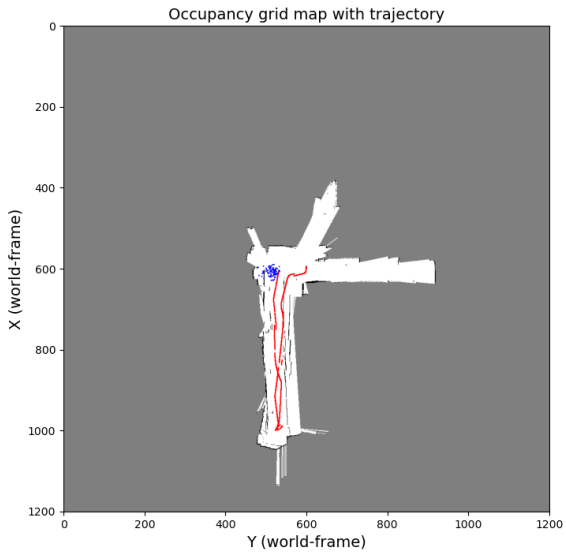

Figure 70. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 2500

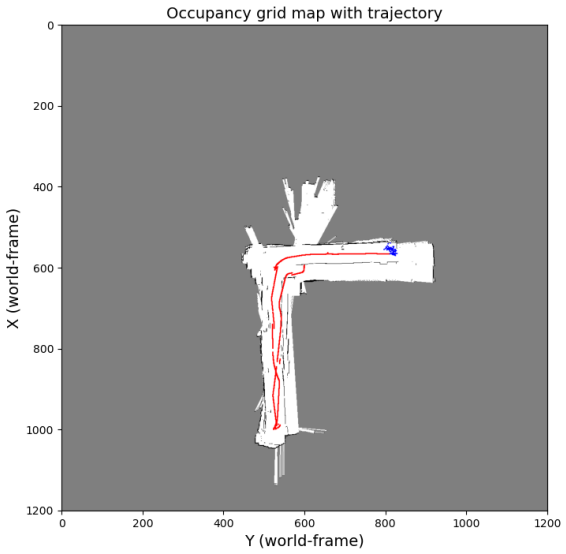

Figure 71. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 3000

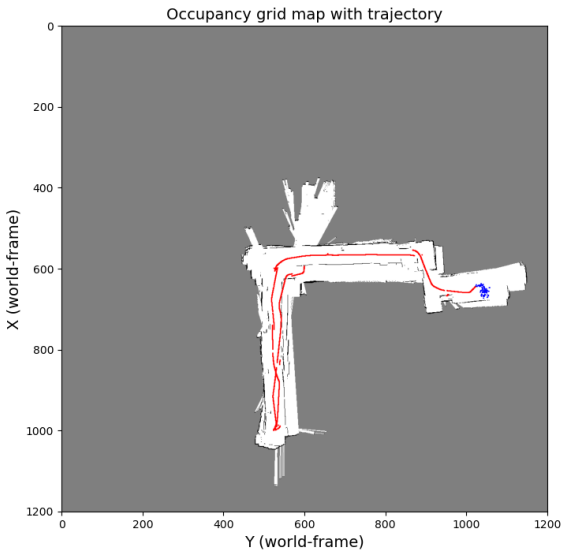

Figure 72. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 3500

Figure 73. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 4000

Figure 74. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, trajectory step 4500

Figure 75. Figure showing final map after PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$

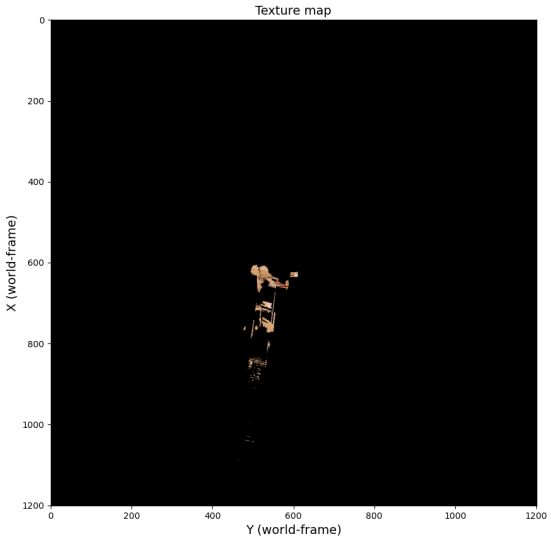

Figure 76. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, step 400

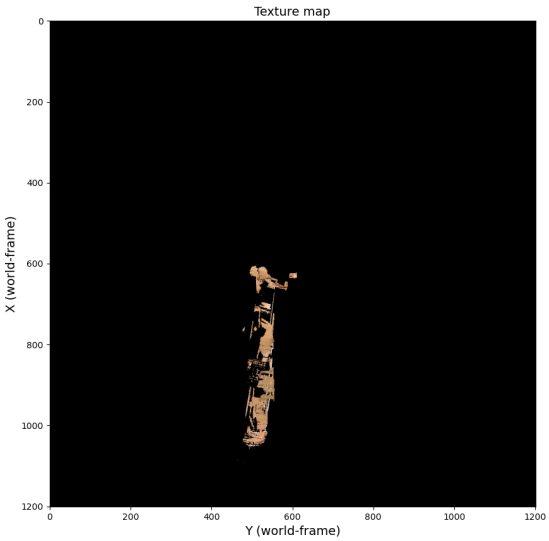

Figure 77. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, step 800

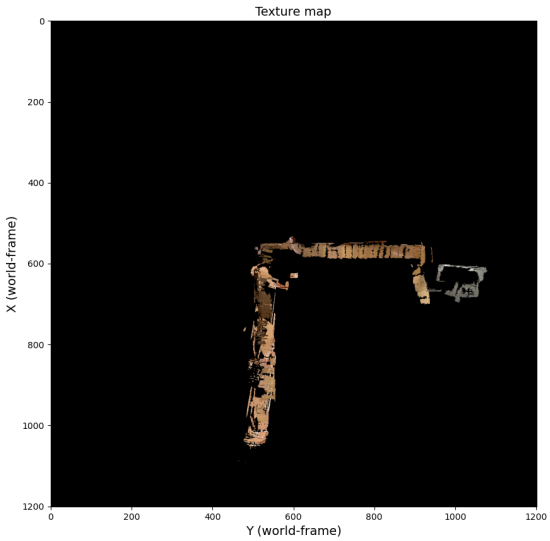

Figure 79. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, step 1600

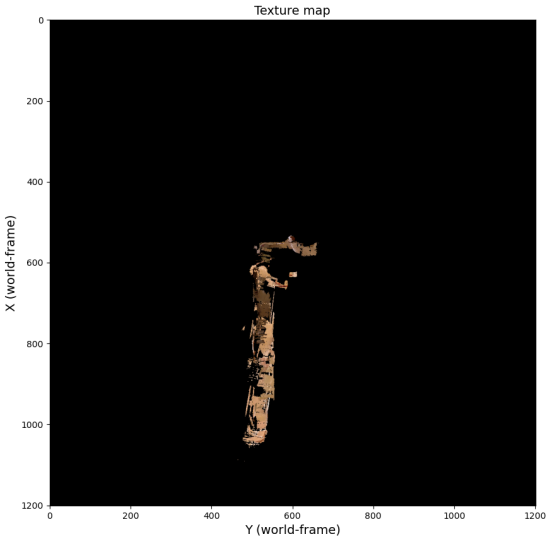

Figure 78. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, step 1200

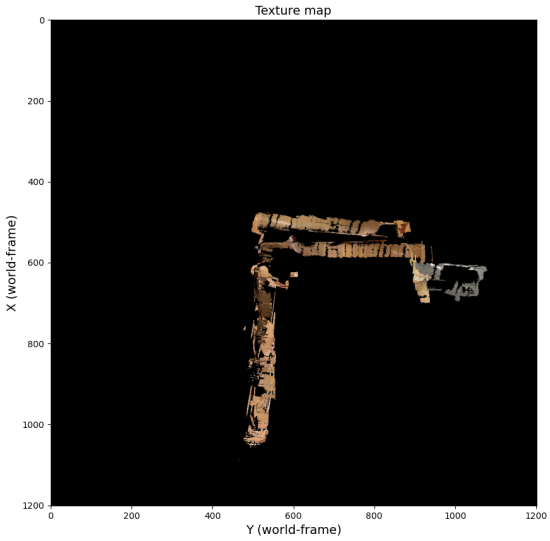

Figure 80. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$, step 2000

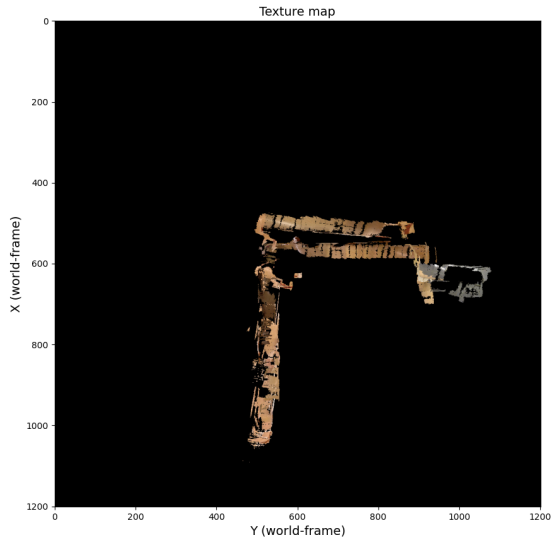

Figure 81. Figure showing final texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.05$

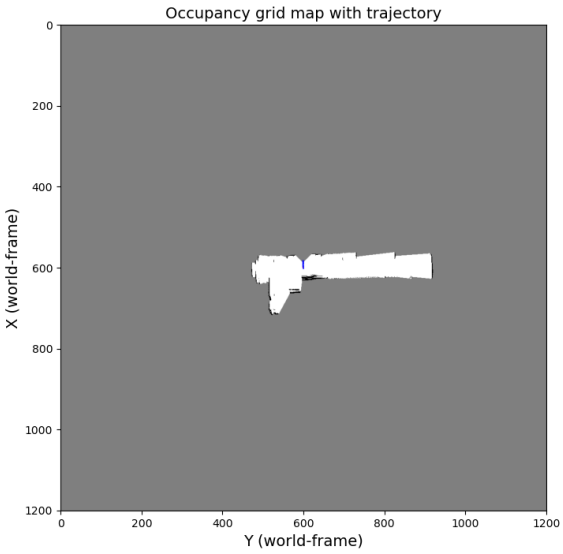

Figure 82. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$, trajectory step 500

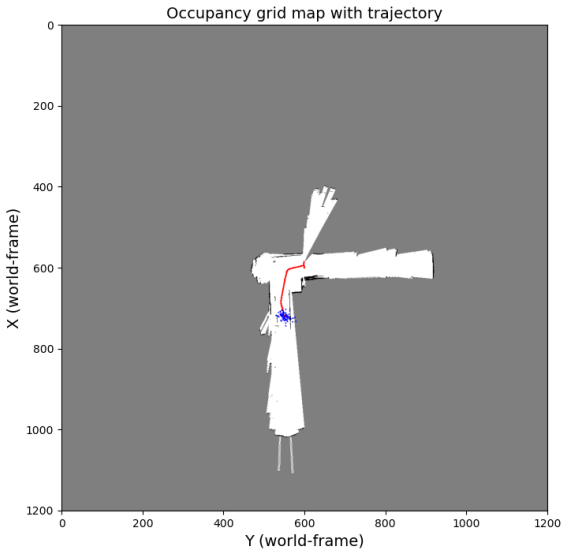

Figure 83. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$, trajectory step 1000

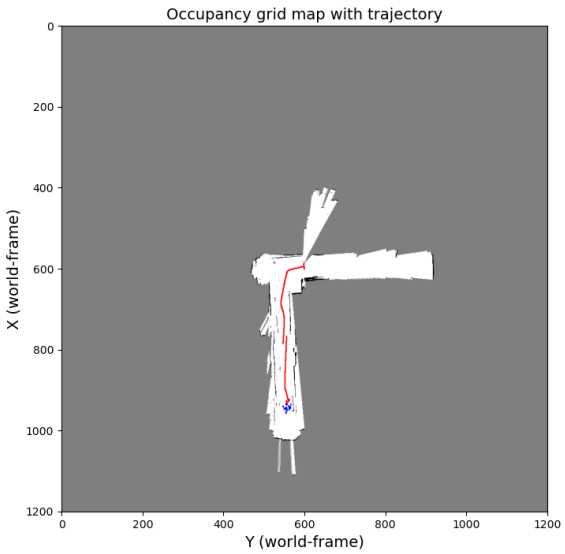

Figure 84. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$, trajectory step 1500

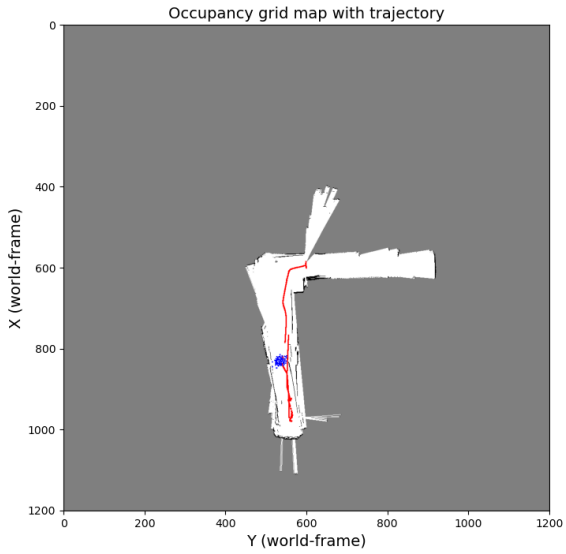

Figure 85. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$, trajectory step 2000

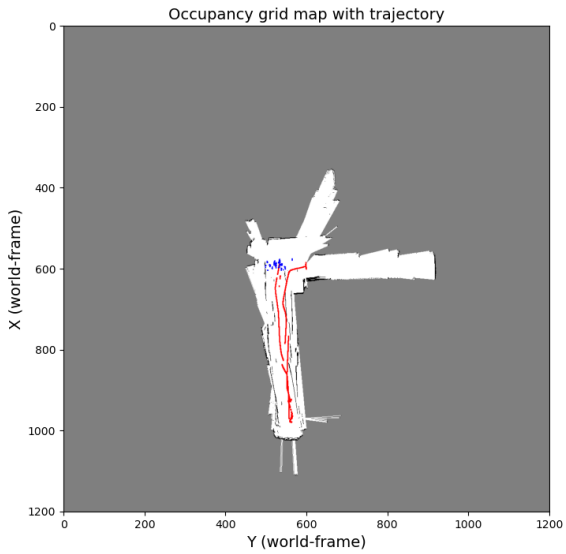

Figure 86. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$, trajectory step 2500

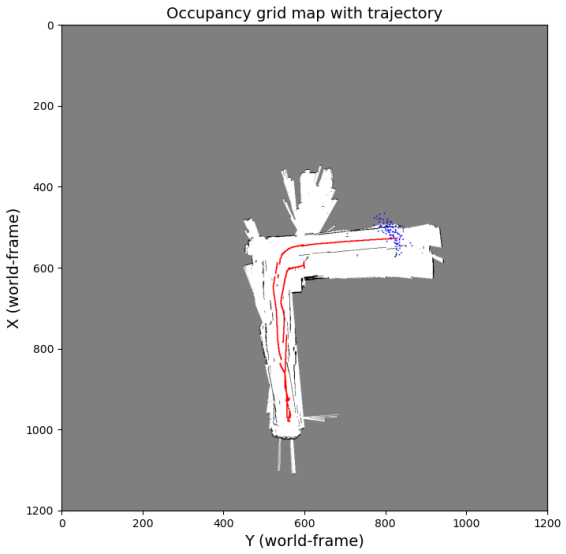

Figure 87. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$, trajectory step 3000

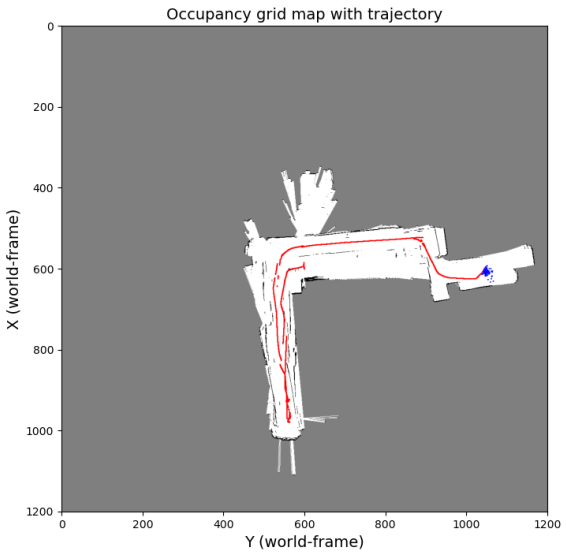

Figure 88. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$, trajectory step 3500

Figure 89. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$, trajectory step 4000

Figure 90. Figure showing map during PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$, trajectory step 4500

Figure 91. Figure showing final map after PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$

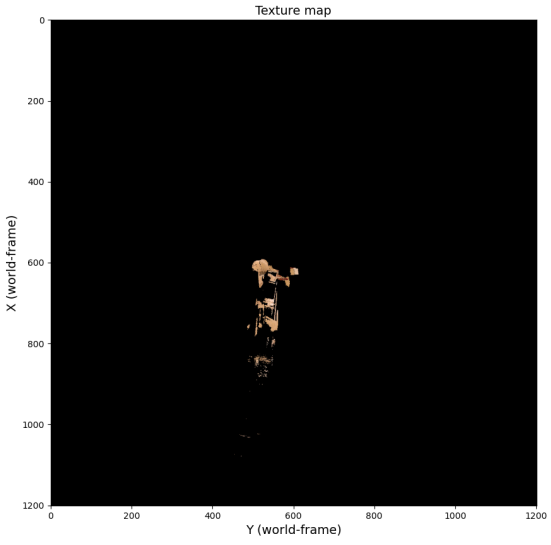

Figure 92. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$, step 400

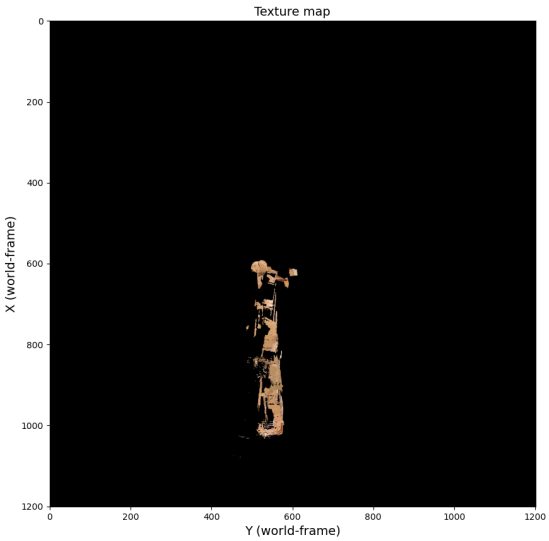

Figure 93. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$, step 800

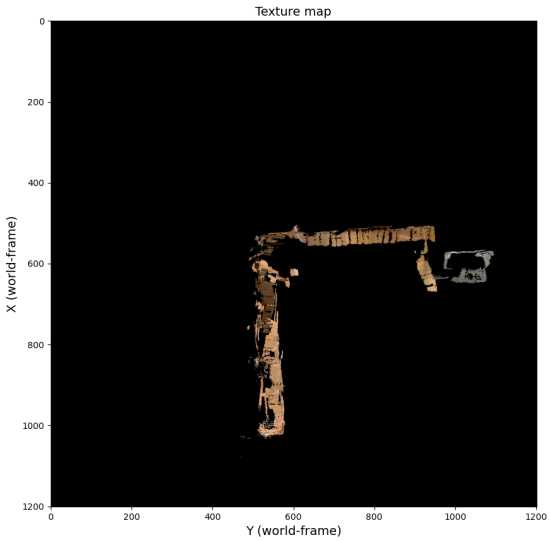

Figure 95. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$, step 1600

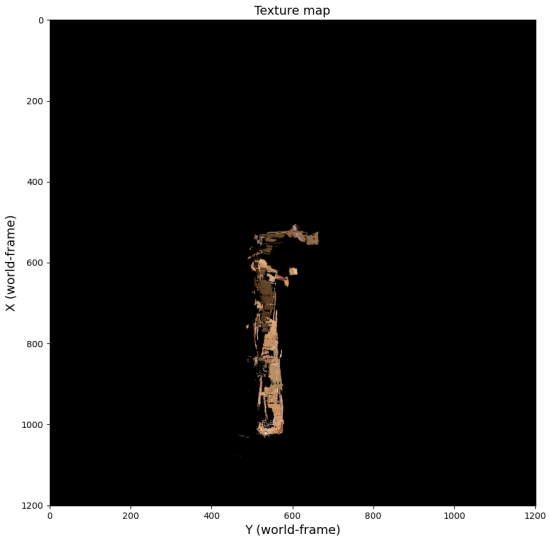

Figure 94. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$, step 1200

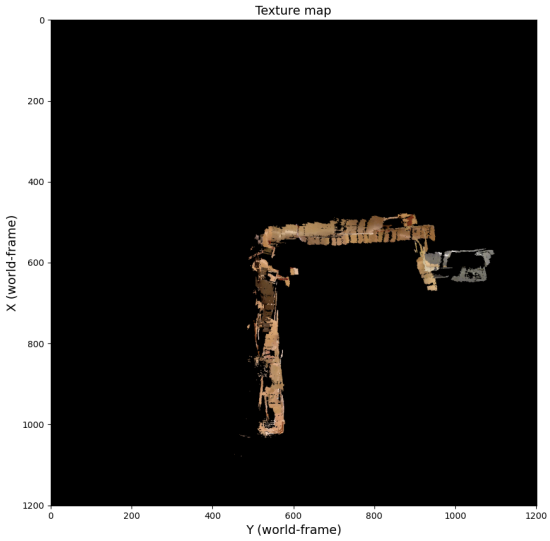

Figure 96. Figure showing texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$, step 2000

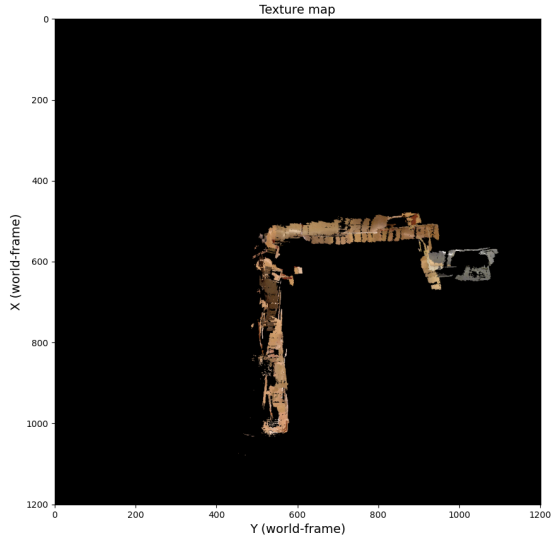

Figure 97. Figure showing final texture map generated using optimal robot trajectory from PF-SLAM with 100 particles, $\sigma_v = 0.5, \sigma_\omega = 0.1$