ECE 276A Project 2: Particle Filter SLAM

1st Yifan Wu
UCSD ECE
yiw084@ucsd.edu)

***Abstract*—The goal of the project is to use a particle filter with a differential-drive motion model and scan-grid correlation observation model for simultaneous localization and occupancy-grid mapping.**

I. Introduction

Simultaneous localization and mapping (SLAM) is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an robot's location within the map. It is initially a chicken-and-egg problem whether mapping comes first or localization comes first. Current State of Art solutions includes the particle filter, extended Kalman filter and Graph SLAM.

Particle filter has a lot of advantages which make it very popular in modern robotics. [1]

- The particles approximate the posterior belief
- It works for any observation model and any motion model
- It works for high dimensional systems
- It is relatively easy to implement

In this project, we implement the solutions for SLAM based on particle filter and occupancy gird map. Also, we perform texture mapping which project colored points from stereo camera onto the occupancy grid map in order to color the cells in the occupancy grid with RGB values according to the projected points that belong to the its plane.

II. Problem Formulation

A. Mapping

Mapping is the problem defined as given the robot state trajectory $x_{0:T}$, build a map m of the environment from noisy and uncertain sensor observations z. Occupancy grid mapping is one of the simplest and most widely used representations. This algorithms usually assume that the cell values are independent conditioned on the robot trajectory:

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

A 2D occupancy grid map is defined as a vector $m \in R^2$, whose i-th entry indicates whether the i-th cell is free($m_i = -1$) or occupied ($m_i = 1$). The size of the grid map is determined by the maximum position of the robot in world frame, $(x_{min} - 80, x_{max} + 80)$ and $(y_{min} - 80, y_{max} + 80)$, where 80 is the max range of lidar. We model the map cells m_i as independent Bernoulli random variables. Since there is no prior information about the map, we assume the map prior is uniform which means occupied and free space are equally likely. During each update step, lidar scans will be convert in to maps cell and perform ray tracing algorithm such as breshenham2D to determine the occupy cell location and free cell location. If the cell is occupy add log-odds ratio to the occupied cell, otherwise subtract the log-odd ratio to the free space. After finish the entire process, recover map pmf from the log-odds map by logistic sigmoid function. The value in each cell represent the probability of being occupied. If the value is greater than 0.5, $m_i = 1$, if the value is less than 0.5, $m_i = -1$

B. Localization

Localization is the problem defined as given a map m, a sequence of control inputs $u_{0:T1}$, and a sequence of measurements $z_{0:T}$, infer the robot state trajectory $x_{0:T}$. In our approach, a particle filter is used to maintain the pdf $p(x_t|z_{0:t}, u_{0:t-1}, m)$. Each particle $\mu_{t|t}^{(k)}$ is a hypothesis on the state x_t with confidence $\alpha_{t|t}^{(k)}$. The particles specify the pdf of the robot state at time t:

$$p_{t|t}(x_t) = p(x_t|z_{0:t}, u_{0:t}, m) \approx \sum_{k=1}^{N} \alpha_{t|t}^{(k)} \delta(x_t; \mu_{t|t}^{(k)}) \tag{2}$$

There are two steps: prediction step and update step.

1) prediction Step: Using the motion model p_f to obtain the predicted pdf $p_{t+1|t}(x_{t+1})$. The motion model is defined as following:

$$x_{t+1} = f(x_t, u_t, w_t) \sim p_f(\cdot|x_t, u_t) \tag{3}$$

where w_t is the motion noise.

For every particle $\mu_{t|t}^{(k)}$, $k = 1, ..., N$ compute:

$$\mu_{t+1|t}^{(k)} = f(\mu_{t|t}^{(k)}, u_t, w_t) \tag{4}$$

$$\alpha_{t+1|t}^{(k)} = \alpha_{t|t}^{(k)} \tag{5}$$

where the particle weights remain unchanged.

2) Update Step: Using the observation model p_h to obtain the updated pdf $p_{t+1|t+1}(x_{t+1})$. The observation model is defined as following

$$z_t = h(x_t, m, v_t) \sim p_h(\cdot|x_t, m) \tag{6}$$

The particle poses remain unchanged but the weights are scaled by the observation model:

$$\mu_{t+1|t}^{(k)} = \mu_{t|t}^{(k)} \tag{7}$$

Update the particle weights using the laser correlation model:

$$\alpha_{t+1|t+1}^{(k)} \propto p_h(z_{t+1}|\mu_{t+1|t}^{(k)}, m)\alpha_{t+1|t}^{(k)} \tag{8}$$

where a laser correlation model will be be defined in technical approach .

C. Texture Mapping

Given a occupancy grid map m, the goal of texture mapping is to form a vector $m \in R^{n\times 3}$ where each cell in the vector is a RGB values according to the projected points that belong to the its plane.

Stereo camera were used to capture images from both left and right camera with total 2×1161 images. Each image has a size (560, 1280), where height is 560 and width is 1280. RGB values, $x \in \mathbb{R}^3$, in each pixel are transformed from pixel coordinate to world coordinate, then to grid map cell coordinate. Cells in the occupancy grid are colored with RGB values according to the projected points which belongs to its plane.

III. TECHNICAL APPROACH

Three measurement dataset, lidar, FOG and encoder, are provide. Since they all have different refresh rate, synchronization is perform to both FOG and encoder. Filling missing value with previous observation of linear velocity by the assumption that linear velocity between two consecutive encoder measurements is constant. By converting timestamps from nanosedonds into milliseconds, FOG, encoder and lidar data are merged with respect to the timestamps in FOG data. Leaving the missing value in lidar data unfilled.

A. Motion Model

A Discrete-time Differential-drive Kinematic Model is used to model the robot. The pose X_{t+1} at time $t+1$, is defined as :

$$X_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} \tag{9}$$

Where x is the x-position and y is the y-position and θ is the yaw angle(rotation angle around z-axis). X_{t+1} can be calculate from X_t by using Euler discretization over time interval of length τ:

$$X_{t+1} = X_t + \tau \begin{bmatrix} v_t \cos\theta_t \\ v_t \sin\theta_t \\ \omega_t \end{bmatrix} \tag{10}$$

where v_t is the linear velocity at time t obtained from wheel encoders, and w_t is the angular velocity at time t obtained from FOG.

B. Observation Model

An observation model is a function relating the robot state x and the environment m with the sensor observation z subject to measurement noise v:

$$z_t = h(x_t, m_t, v_t) \sim p_h(\cdot | x_t.m_t) \tag{11}$$

1) Encoder: The distance traveled during time τ for a given encoder count z, wheel diameter d, and 4096 ticks per revolution is:

$$\tau v \approx \frac{\pi d z}{4096} \tag{12}$$

Given τ linear velocity for both left wheel and right wheel can be calculate. Taking average of them will obtain the linear velocity of the robot.

2) FOG: directly provides delta yaw, θ_t, hence the angular velocity ω_t can be simply calculate by

$$\omega_t = \frac{\theta_t}{\tau} \tag{13}$$

Let FOG frame be the body frame to simply the transformation of measurements.

3) Lidar: collects data as ranges $r \in [2, 80]$ meters and angles $\theta \in [-5, 185]$ degrees with resolution 0.666 which is in Polar coordinate. we need to convert lidar data to Cartesian coordinate and transforming the point clouds to body frame.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r\cos\theta \\ r\sin\theta \\ 0 \end{bmatrix} \tag{14}$$

$$T_B = \begin{bmatrix} 1 & 0 & 0 & 0.8349 \\ 0 & 1 & 0 & -0.126869 \\ 0 & 0 & 1 & 1.76416 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

$$T_V = \begin{bmatrix} 1 & 0 & 0 & 0.335 \\ 0 & 1 & 0 & 0.035 \\ 0 & 0 & 1 & -0.78 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16}$$

$$\underline{s_B} = T_V T_L \underline{s_L} \tag{17}$$

where s_B and s_L are in homogeneous coordinate, $T_L \in R^{4\times 4}$ and $T_V \in R^{4\times 4}$ is transformation matrix from lidar frame to vehicle frame, and vehicle frame to body frame

C. Probabilistic Occupancy Grid Mapping

Dead Reckoning is perform to figure out the size of the map. The trajectory of the robot is predicted by using the differential driving model without noise. After considering the range of lidar, (x_{min}, x_{max}) and (y_{min}, y_{max}) are initiate as following:

$$\begin{aligned} &x_{max} = 1350 \\ &x_{min} = -100 \\ &y_{max} = 100 \\ &y_{min} = -1350 \\ &resolution = 1 \\ &mapsize : (1451, 1451) \end{aligned} \tag{18}$$

lidar scan, s_L, in word frame is in regular $[x, y, z]$ axis. In order to convert these points into 2D grid map, we need to project these points to the map by removing the z-axis. The occupancy grid map is a 2D matrix $m \in^{M\times N}$, where $M = ceil((xmax - xmin)/resolution + 1)$ and $N = ceil((ymax - ymin)/resolution + 1)$. Therefore, the coordinate in world frame can be transformed in to map coordinate:

$$\begin{aligned} i = ceil((x_w - x_min)/resolution) \\ j = ceil((y_w - y_min)/resolution) \end{aligned} \tag{19}$$

In our problem, map cells m_i are modeled as independent Bernoulli random variables

$$m_i = \begin{cases} 1 & \text{if occupied with prob. } \gamma_{i,t} \\ -1 & \text{if free with prob. } 1 - \gamma_{i,t} \end{cases} \tag{20}$$

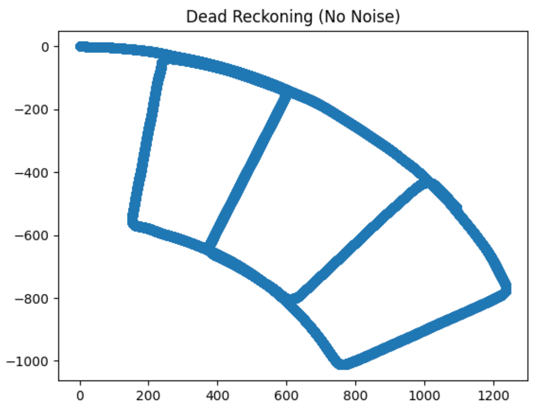


Fig. 1. *Dead Reckoning*

It is sufficient to track $\gamma_{i,t} := p(m_i = 1|z_{0:t}, x_{0:t})$ for each map cell.

Odds ratio of the Bernoulli random variable m_i updated via Bayes rule:

$$o(m_i|z_{0:t}, x_{0:t}) = g_h(z_t|m_i, x_t)o(m_i|z_{0:t-1}, x_{0:t-1}) \quad (21)$$

$$o(m_i|z_{0:t}, x_{0:t}) := \frac{p_h(z_t|m_i = 1, x_t)}{p_h(z_t|m_i = -1, x_t} \frac{\gamma_{i,t}}{1 - \gamma_{i,t}} \quad (22)$$

Assuming z_t indicates whether m_i is occupied or free, the inverse observation model odds ratio specifies the ratio of true positives versus false positives: $\Delta\lambda_{i,t} = 0.8/0.2 = 4$ Assuming, the map prior is uniform $\lambda_{i,0} = \log(1) = 0$

Log-odds of the Bernoulli random variable m_i:

$$\lambda_{i,t} = \log o(m_i|z_{0:t}, x_{0:t}) \quad (23)$$

Log-odds occupancy grid mapping:

$$\begin{aligned} \lambda_{i,t} &= \log \frac{p(m_i=1|z_t,x_t)}{p(m_i=-1|z_t,x_t)} - \lambda_{i,0} + \lambda_{i,t-1} \\ \lambda_{i,t} &= \lambda_{i,t-1} + \Delta\lambda_{i,t} - \lambda_{i,0} \\ \lambda_{i,t} &= \lambda_{i,t-1} \pm \log 4 \end{aligned} \quad (24)$$

The map pmf $\gamma_{i,t}$ can be recovered from the log-odds $\lambda_{i,t}$ via the logistic sigmoid function:

$$\gamma_{i,t} = \sigma(\lambda_{i,t}) = \frac{\exp(\lambda_{i,t})}{1 + \exp(\lambda_{i,t})} \quad (25)$$

D. Particle Filter

The particle filter is a histogram filter which allows its grid centers to move around and adaptively concentrate in areas of the state space that are more likely to contain the true state. The representation of the particle and weight is shown as below:

$$\{\mu_{t|t}^{(k)}, \alpha_{t|t}^{(k)}\} \quad (26)$$

where $k = 1, ..., N$

Each particle $\mu_{t|t}^{(k)} \in R^3$ represents a possible robot 2-D position (x,y) and orientation θ. Initially, set particles $\mu_{t|t}^{(k)} = [0, 0, 0]^T$ with uniform weights $\alpha_{t|t}^{(k)} = \frac{1}{N}$, where N is the number of particles.

1) Prediction Step: We predict every particle $\mu_{t|t}^{(k)}$, $k = 1, ..., N$, with the following equation in our implementation:

$$\mu_{t+1|t}^{(k)} = f(\mu_{t|t}^{(k)}, u_t, \epsilon_t) \quad (27)$$

$$\alpha_{t+1|t}^{(k)} = \alpha_{t|t}^{(k)} \quad (28)$$

- f(x,u) is the differential-drive motion model
- $u_t = (v_t, \omega_t)$ is the linear and angular velocity input
- $\epsilon_t \sim N(\begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_w^2 \end{bmatrix})$ is a 2-D Gaussian motion noise with covariance of v_t and ω_t

2) Update Step: the particle poses remain unchanged but the weights are scaled by the observation model:

$$\mu_{t+1|t}^{(k)} = \mu_{t|t}^{(k)} \quad (29)$$

Update the particle weights using the laser correlation model:

$$\alpha_{t+1|t+1}^{(k)} \propto p_h(z_{t+1}|\mu_{t+1|t}^{(k)}, m)\alpha_{t+1|t}^{(k)} \quad (30)$$

we choose the particle with largest weight and project the laser scan z_t to world frame and update the map log-odds. In our approach, we predict the true state of the robot by selecting the particle with the largest weight:

$$\begin{aligned} k* &= argmax(\alpha_{t+1|t}^{(k)}) \\ X_{t+1|t} &= \mu_{t+1|t}^{(k*)} \end{aligned} \quad (31)$$

where $k = 1, ..., N$

Laser Correlation Model sets the likelihood of a laser scan z proportional to the correlation between the scan's world-frame projection y= r(z, x) via the robot pose x and the occupancy grid m:

$$p_h(z|x, m) \propto \exp(corr(r(z, x), m)) \quad (32)$$

Defined a similarity function corr(r(z, x), m) between the transformed and discretized scan y and the occupancy grid m:

$$corr(r(z, x), m) = \sum_i 1\{y_i = mi\} \quad (33)$$

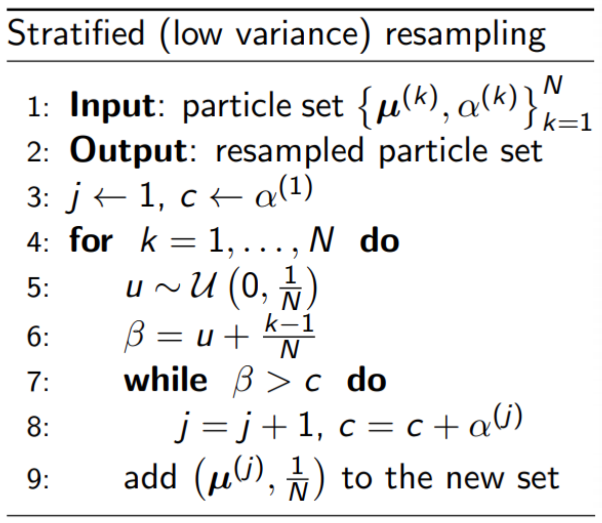
Transform the scan z_{t+1} to the world frame using $\mu_{t+1|t}^{(k)}$ and find all cells $y_{t+1|t}^{(k)}$ in m corresponding to the scan. The correlation $corr(y_{t+1|t}^{(k)}, m)$ is large if $y_{t+1|t}^{(k)}$ and m agree.

E. Resampling

Due to the limited finite number of particles, particle depletion is likely to happen. Most of the updated particle weights becomes close to zero. Resampling is used to prevent particle depletion. The particles with small weight are replaced by new particles with the higher weights.

The resampling is perform when the effective number of particles is less than a threshold:

$$N_{eff} = \frac{1}{\sum_{k=1}^{N} \alpha_{t|t}^{(k)}} \leq N_{threshold} \quad (34)$$



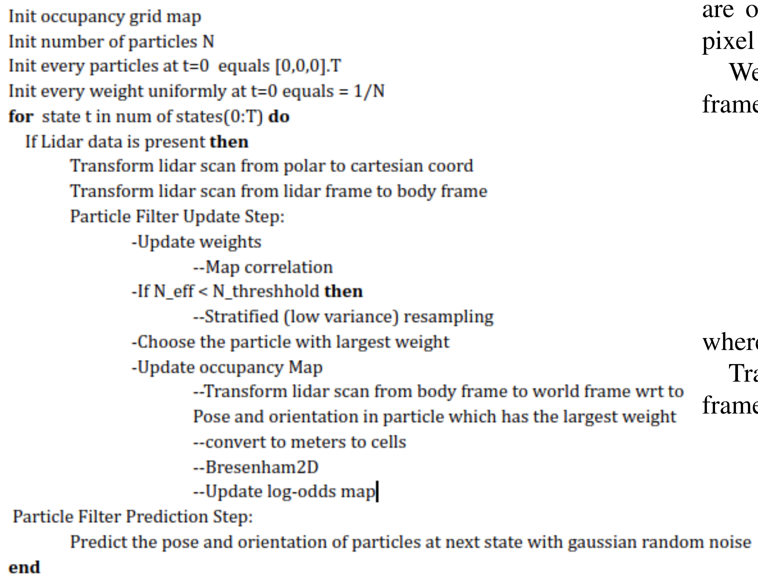
```
Stratified (low variance) resampling
1: Input: particle set $\{\mu^{(k)}, \alpha^{(k)}\}_{k=1}^{N}$
2: Output: resampled particle set
3: $j \leftarrow 1,\ c \leftarrow \alpha^{(1)}$
4: for $k = 1, \ldots, N$ do
5:     $u \sim \mathcal{U}\left(0, \frac{1}{N}\right)$
6:     $\beta = u + \frac{k-1}{N}$
7:     while $\beta > c$ do
8:         $j = j + 1,\ c = c + \alpha^{(j)}$
9:     add $\left(\mu^{(j)}, \frac{1}{N}\right)$ to the new set
```

Fig. 2. *Stratified Resampling*

1) Stratified Resampling: ensures that the samples with large weight will be selected at least once an samples with small weight will be select at most once. Therefore, stratified resampling has low variance. It has the desirable property that any particle with normalized weight greater than $1/N$ is guaranteed to be drawn. The algorithm is shown in figure 2.

F. SLAM

Particle filter SLAM combines the particle filter(prediction step and update step) and updating the occupancy grid map. The pesudo code for the overall algorithm.



```
Init occupancy grid map
Init number of particles N
Init every particles at t=0  equals [0,0,0].T
Init every weight uniformly at t=0 equals = 1/N
for  state t in num of states(0:T) do
  If Lidar data is present then
        Transform lidar scan from polar to cartesian coord
        Transform lidar scan from lidar frame to body frame
        Particle Filter Update Step:
                -Update weights
                        --Map correlation
                -If N_eff < N_threshhold then
                        --Stratified (low variance) resampling
                -Choose the particle with largest weight
                -Update occupancy Map
                        --Transform lidar scan from body frame to world frame wrt to
                        Pose and orientation in particle which has the largest weight
                        --convert to meters to cells
                        --Bresenham2D
                        --Update log-odds map
 Particle Filter Prediction Step:
        Predict the pose and orientation of particles at next state with gaussian random noise
end
```

Fig. 3. *SLAM Algorithm*

G. Texture Mapping

Stereo camera were used to capture images from both left and right camera with total 2×1161 images. Each image has a size (560, 1280), where height is 560 and width is 1280. RGB values, $x \in \mathbb{R}^3$, in each pixel are transformed from pixel coordinate to world coordinate, then to grid map cell coordinate. Cells in the occupancy grid are colored with RGB values according to the projected points which belongs to its plane.

The projection matrix from right stereo camera is the following:

$$P_R = \begin{bmatrix} 775.372 & 0 & 619.473 & -368.4171 \\ 0 & 775.372 & 257.180 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{35}$$

From the projection matrix and stereo camera parameters, we can obtain the following value.

TABLE I
STEREO CAMERA PARAMETERS

SLAM Configuration		
Parameter	Description	Value
b	Baseline of stereo cameras	0.475
fsu	Focal and pixel scaling	775.372
fsv	Focal and pixel scaling	775.372
cu	Coordinates of the principal point	619.473
cv	Coordinates of the principal point	257.180

The disparity can be calculated as:

$$d = u_L - u_R = \frac{1}{z} f s_u b \tag{36}$$

where u_L is the pixel coordinates location in left camera and u_R is the pixel coordinates location in right camera. Images are obtained in terms of pixels (u, v) with the origin of the pixel array typically in the upper-left corner of the image.

We are able to calculate the (x, y, z) location in optical frame:

$$z = \frac{f s_u b}{d} \tag{37}$$

$$y = \frac{v_L - c_v}{f s_v} \tag{38}$$

$$x = \frac{u_L - c_u}{f s_u} \tag{39}$$

where z is the depth we would like to obtain.

Transforming the point from optical frame to regular frame,and then to body frame as follow:

$$\underline{m_B} = T_V T_o \underline{m_o} \tag{40}$$

$$R_o = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \tag{41}$$

$$T_o = \begin{bmatrix} R_o & 0 \\ 0^T & 1 \end{bmatrix}^{-1} \tag{42}$$

At each update step, transform the pixel from body frame to world frame by rotating around z-axis in world frame with yaw angle from the particle with the largest weight. Then find the location of pixel in the gird map cell coordinate with respect to free cells. Consider only the pixels whose z coordinate in the world frame falls within a small threshold of the flat plane of the map, and associate the RGB values with the corresponding map cells

$$m_W = R_z(\theta) m_B + p \tag{43}$$

where $R_z(\theta)$ is the rotation around z-axis, p = $[x, y]^T$, with respect to the particle with the largest weight, $\mu_{t|t}^{(k)} \in R^3$, k= $argmax(\alpha_{t|t}^{(k)})$.

IV. Result

By converting timestamps into milliseconds merge FOG, encoder and lidar data with respect to the timestamps in FOG data. All of the test using the same merged dataset with number of timestamps = 1160508. The result of particle filter SLAM is performed by different number of particles. The parameter is described in Table 2.

TABLE II
Particle Filter SLAM Parameter

SLAM Configuration		
Parameter	Description	Value
Map[xsize]	Map width	1451
Map[ysize]	Map height	1451
Map[res]	Map resolution	1
$\Delta\lambda_{i,t}$	Log-odds ratio	$\pm log(4)$
$\lambda_{i,0}$	Prior odds ratio	log 1
N	Num of particles	{1,5,25,50,100}
$N_{threshold}$	N_{eff} threshold for re-sampling	{0.2×N, 5}
ϵ	Gaussian noise	diag($\sigma_{v_t}^2$), $\sigma_{\omega_t}^2$)

Adding noise and resample the weights is the key of particle filter. Use low variance resampling such as stratified, to prevent particle depletion and congregate more particles to one state.

- Approximately 6282 resampling performed every 100000 steps when N = 25, $N_{Threshold=5}$
- Approximately 8919 resampling performed every 100000 steps when N = 50, $N_{Threshold=5}$
- Approximately 14197 resampling performed every 100000 steps when N = 100, $N_{Threshold=5}$
- Approximately 3891 resampling performed every 100000 steps when N = 50, $N_{Threshold=10}$

The affect of $N_{Threshold}$ are compared by Figure 7 and 8.

Since we add Gaussian random noise, each trajectory is different from each other, but the overall path is similar. Adding noise to the pose appropriately will improve the mapping. A larger noise cause a larger gap in the path provides more degree of freedom to detect occupied and free cells. Adding noise also helps to prevent overfitting the map. As we can see in Figure 6, the first plot is brighter which has a small noise. A large number of edges might be overconfident. Below two plot in Figure has a larger noise which decrease the chance of overestimate the edge. Another method to avoid overconfident estimation is to constrain the log-odd in each cells by setting a minimum and maximum value $\lambda_{MIN} \leq \lambda_{i,t} \leq \lambda_{MA}$.

Testing with different number of particles, $\{10, 25, 50\}$, a small change in number of particles, for instance, 3 to 5, has negligible affect on the result. This might due to the N_{eff} is not easy to hit the $N_{Threshold}$ when number of particles is small. The change of number of partical might be positively proportional to the number of resampling particles. On the other hand, a large increase of number of particles do have obvious affect on the result. As shown in figure 5, there are more and more edges are connected as number of particles increase. The entire map with 50 particles becomes more clear than map with 10 and 25 particles. There are more and more small detail shown in the map.

The edge of the map is not smooth, since we are using finite number of particles to model infinite continuous environment. In order to further smooth the edge, we have two options. Decrease the map resolution so that one meter will be convert to multiple map cells. Another method which guaranteed improvement is to increase number of particles as larger as possible. This will definitely increase the accuracy of the map, because this will increase the number of resamples the particles need to congregate to one state. However, a large number of practicals will dramatically increase the run time of the process as well.

Especially, using the default breshenham2D ray tracing algorithm is very slow. Further improvement can be using cv2.findcontours. Since a lot of resampling will be performed, a larger number of particles will increase the number of iteration of the for loop in stratified resampling. A better low-variance resampling method is to use Systematic resampling which is similar to stratified resampling except that the same uniform is used for each piece. It only sampled once before the for loop shown in figure 2.

The limitation of particle filter SLAM is that despite vectorizing the code, a good particle filter still requires a large number of of particles to perform well which is slow in practice. Moreover, measuring particle filter performance is difficult. since the process is not deterministic, the results are distinctive even with the same input and parameter. There is no convenient way of relating accuracy to number of particles. Similarly, particle filters offer no measure of confidence in their readings.

1) Future Work: Due to the complexity and overall runtime of the project, texture mapping is conceptually performed, but the implementation is not complete yet. Since there are only 1161 images at each timestamp, the sample is relatively samll compared to the entire merged dataset. Synchronization between Lidar data and stereo camera data is difficult to perform, since the stereo camera has the highest refreshing rate.

References

[1] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit *et al.*, “Fastslam: A factored solution to the simultaneous localization and mapping problem,” *Aaai/iaai*, vol. 593598, 2002.

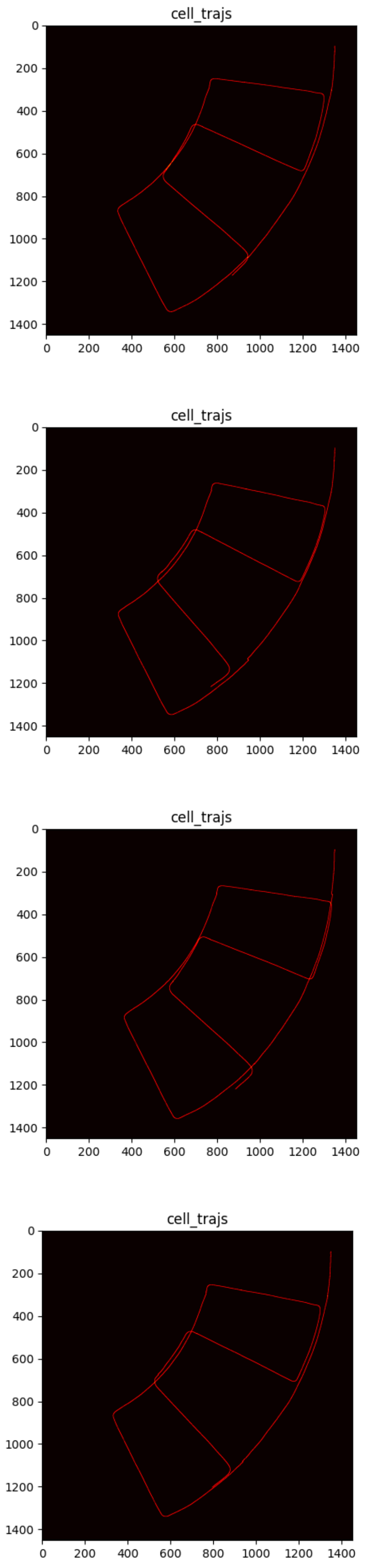


Fig. 4. *Comparison between Trajectories in Grid Map (N=10, 25, 50,100)*

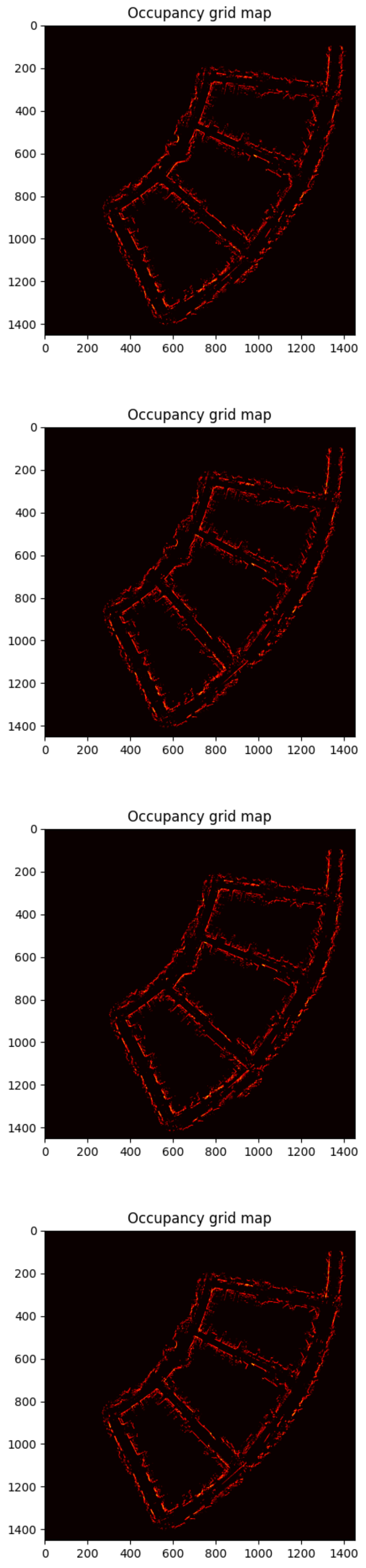


Fig. 5. *Comparison between Occupancy Grid Map (N=10, 25, 50, 100)*

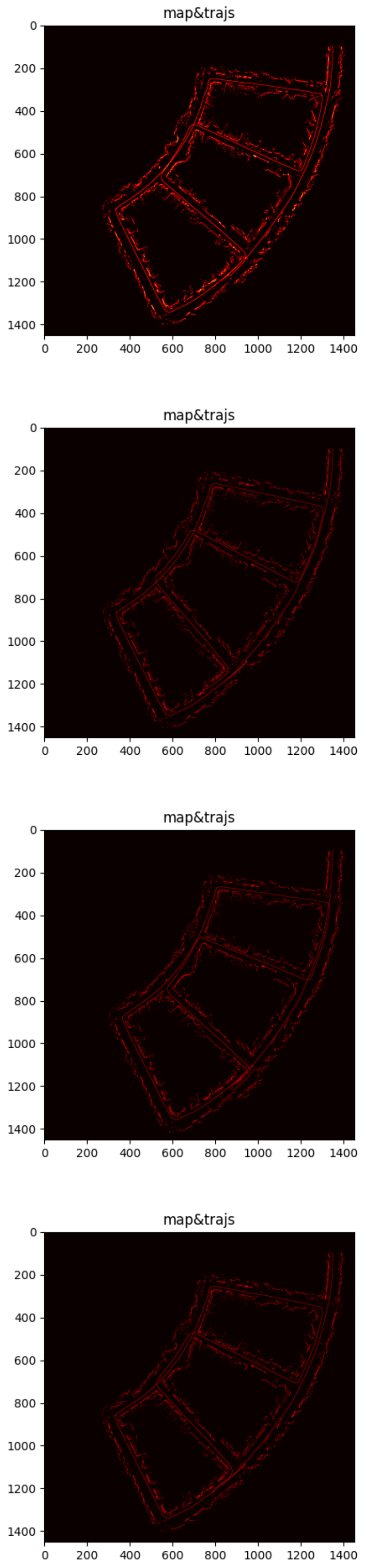


Fig. 6. *Comparison between Occupancy Grid Map and Trajectory (N=10, 25, 50, 100)*

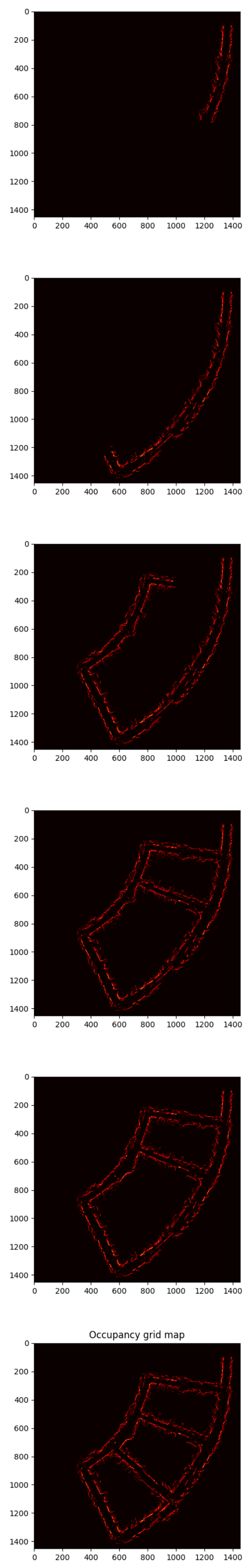


Fig. 7. *Occupancy Grid Map Creation Process (N=50, $N_{Threshold}$ = 10)*

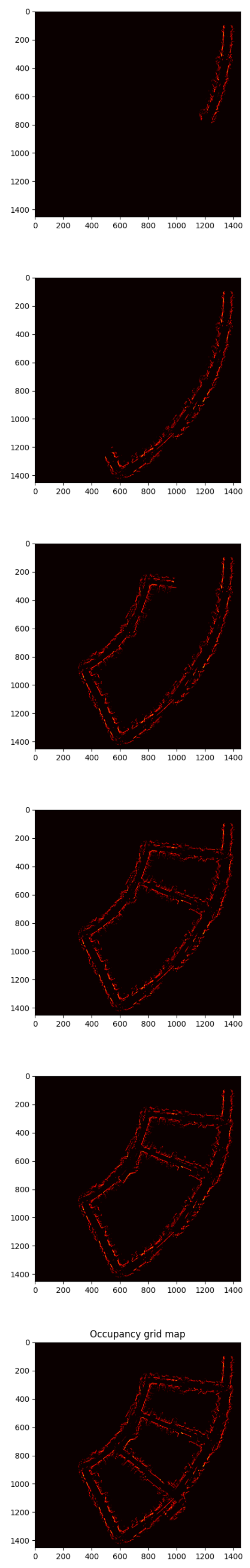


Fig. 8. *Occupancy Grid Map Creation Process (N=50, $N_{Threshold}$ = 5)*

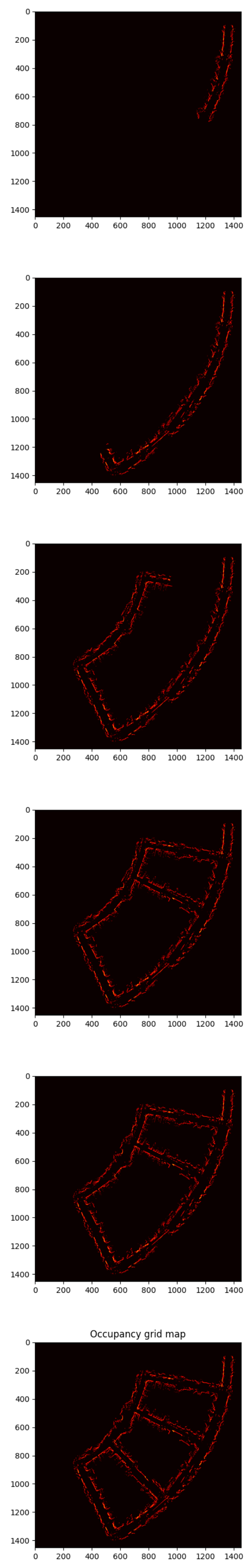


Fig. 9. *Occupancy Grid Map Creation Process (N=100, $N_{Threshold}$ = 5)*