Particle Filter SLAM

Hisu-Wen Yen A59010599 hsyen@ucsd.edu

I. Introduction

SLAM technique is used to build the map around robot and to localize robot's position, and map is very important element in robotics. If we have map, then we can require robot to perform some specific tasks. However, most of the time the map is unknown, then we need to use SLAM to discover the map. In this report contains how to use particle filter to perform SLAM.

II. Problem Formulation

In this problem, we use notation z as observation, u as control input and x as robot state.

A. Mapping

In real life map is scaled by meters and is continuous. However, in computer we can't store continuous value. Therefore, we use grid to discretize out map. In our problem, we will build 2D map, therefore, we can store 2D array m as our map.

$$m \in R^{mn} \tag{1}$$

mn is total amount of cells in grid map.

$$m_i = \begin{cases} 1, occupied\ grid \\ -1, free\ grid \end{cases} \tag{2}$$

We are not 100% sure about the map, so we use probability to represent whether the grid is occupied.

$$\gamma \in R^{mn} \tag{3}$$

$$\gamma_i \in [0,1] \tag{4}$$

Therefore, for mapping problem we want to compute

$$p(m|z_{0:t}, x_{0:t}) \text{ over time} \tag{5}$$

B. Localization

Besides mapping, we also want to know where the robot is now. Given the map m, inputs u and observations z, we want to estimate position of robot.

Still, we are not 100% sure where the robot is, so we use probability to represent.

Therefore, for localization problem, we want to compute

$$p_{t|t}(x_t|m, z_{0:t}, u_{0:t-1}) \text{ over time} \tag{6}$$

C. Texture mapping

With stereo camera, we can get colorful image with timestamp, and we can project a point on image into grid map to color the grid map.

Here I use RGB to represent color; set colored map cm.

$$cm \in Color^{mn} \tag{7}$$

$$Color = (R, G, B) \tag{8}$$

$$then\ cm_i = (r_i, g_i, b_i) \tag{9}$$

The problem becomes compute

$$p(cm \mid z_{0:t}, x_{0:t}) \tag{10}$$

III. Thectival Approches

In this report, we use particle filter to solve mapping and localization stated below.

$$p(m|z_{0:t}, x_{0:t}) \tag{11}$$

$$p_{t|t}(x_t|m, z_{0:t}, u_{0:t-1}) \tag{12}$$

A. Mapping

We assume each cell is independent to other cells, then:

$$p(m|z_{0:t}, x_{0:t}) = \prod p(m_i|z_{0:t}, x_{0:t}) \tag{13}$$

We can model m_i as independent Bernoulli random variables:

$$m_i = \begin{cases} 1(occupied), with\ prob.\ \gamma_{i,t} \\ -1(free), with\ prob.\ 1-\gamma_{i,t} \end{cases} \tag{14}$$

Apply Bayes rule and Markov assumption:

$$\begin{aligned} \gamma_{i,t} &= p(m_i = 1|z_{0:t}, x_{0:t}) \\ &= \frac{p(z_t|m_i = 1, z_{0:t-0}, x_{0:t})p(m_i = 1|z_{0:t-1}, x_{0:t})}{p(z_t|z_{0:t-1}, x_{0:t})} \\ &= \frac{p(z_t|m_i = 1, x_t)p(m_i = 1|z_{0:t-1}, x_{0:t-1})}{p(z_t|z_{0:t-1}, x_{0:t}) = \eta_t} \\ &= \frac{1}{\eta_t} Ph(z_t|m_i = 1, x_t)\gamma_{i,t-1} \end{aligned} \tag{15}$$

We can do similar way to $1-\gamma_{i,t}$, then

$$1 - \gamma_{i,t} = \frac{1}{\eta_t} Ph(z_t|m_i = -1, x_t)(1 - \gamma_{i,t-1}) \tag{16}$$

We create odd ratio:

$$o(m_i|z_{0:t}, x_{0:t}) = \frac{\gamma_{i,t}}{1-\gamma_{i,t}} = \frac{Ph(z_t|m_i=1,x_t)(\gamma_{i,t-1})}{Ph(z_t|m_i=-1,x_t)(1-\gamma_{i,t-1})} \quad (17)$$

$$gh(z_t|m_i, x_t) = \frac{Ph(z_t|m_i=1,x_t)}{Ph(z_t|m_i=-1,x_t)} \quad (18)$$

Take log to $o(m_i|z_{0:t}, x_{0:t})$

$$\lambda_{i,t} = \log o(m_i|z_{0:t}, x_{0:t}) \quad (19)$$

$$\lambda_{i,t} = \log(gh) + \lambda_{i,t-1} \quad (20)$$

Apply Bayes rule to $gh(z_t|m_i, x_t)$:

$$gh(z_t|m_i, x_t) = \frac{p(m_i=1|z_t,x_t)}{p(m_i=-1|z_t,x_t)} \frac{p(m_i=-1)}{p(m_i=1)} \quad (21)$$

$p(m_i|z_t, x_t)$ is inverse observation model, and we treat it as hyperparameter and it means that how much we trust the observations, plug this into (14), we get:

$$\lambda_{i,t} = \log \frac{p(m_i=1|z_t,x_t)}{p(m_i=-1|z_t,x_t)} - \lambda_{i,0} + \lambda_{i,t-1} \quad (22)$$

We can perform sigmoid function to change log-odd ratio to map pmf $\gamma_{i,t}$

$$\gamma_{i,t} = \frac{\exp(\lambda_{i,t})}{1+\exp(\lambda_{i,t})} \quad (23)$$

In this project, I transform each end point of lidar ray and robot's position to map grid coordinates, then apply provided bresenham2D to find all cells in the path of each lidar ray. Only treat last cell in the path as occupied cell and the rest as free cells to update my map.

B. Localization

To compute $p_{t|t}(x_t|m, z_{0:t}, u_{0:t-1})$ over time, we use Markov assumptions to derive Bayes Filter which only needs to keep track of two pdf (Updated pdf and Predicted pdf). Here we will use Particle Filter based on Bayes Filter.

In particle filter we set particle μ^k is a hypothesis on the state x_t with confidence a^k:

$$p_{t|t}(x_t) \coloneqq p(x_t|z_{0:t}, u_{0:t-1}, m) \approx \sum_{k=1}^{n} a_{t|t}^k \, \delta(x_t; \mu_{t|t}^k) \quad (24)$$

Plug (24) into Bayes Filter prediction step, we get:

$$\begin{aligned} p_{t+1|t}(x) &= \int Pf(x|s,u) p_{t|t}(s) ds \\ &= \int Pf(x|s,u) \sum_K a_{t|t}^k \delta(s-\mu_{t|t}^k) ds \\ &= \sum_K a_{t|t}^k \int Pf(x|s,u)\, \delta(s-\mu_{t|t}^k)\, ds \\ &= \sum_K a_{t|t}^k Pf(x|\mu_{t|t}^k, u) \end{aligned} \quad (25)$$

Try to view (25) with physical meaning.

Usually we use motion model Pf to obtain predicted pdf $p_{t+1|t}(x_{t+1})$, but in particle filter we only have discrete particles. Therefore, it will become applying motion model on each particles. All particles need to move along with the motion on their own frame, which we call prediction step:

$$\mu_{t+1|t}^k = f(\mu_{t|t}^k, u_t + e_t) \quad (26)$$

e_t is input error in time t

$$a_{t+1|t}^k = a_{t|t}^k \quad (27)$$

We only move particles, so the weights a^k remain the same.

Plug (24) into Bayes Filter update step, we get:

$$\begin{aligned} p_{t+1|t+1}(x) &= \frac{Ph(z_{t+1}|x) \sum_K a_{t+1|t}^k \delta(x-\mu_{t+1|t}^k)}{\int Ph(z_{t+1}|s) \sum_K a_{t+1|t}^k \delta(s-\mu_{t+1|t}^k)\, ds} \\ &= \sum_K \left[\frac{a_{t+1|t}^k Ph(z_{t+1}|\mu_{t+1|t}^k)}{\sum_K a_{t+1|t}^k Ph(z_{t+1}|\mu_{t+1|t}^k)} \right] \delta(x-\mu_{t+1|t}^k) \end{aligned} \quad (28)$$

In this step, we introduce observations to tune a^k. The particles poses remain unchanged but the weights are scaled by the observation model:

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

$$a_{t+1|t}^k \propto Ph(z_{t+1}|\mu_{t+1|t}^k, m) * a_{t|t}^k \quad (30)$$

We need to define lidar observation model $Ph(z_{t+1}|\mu_{t+1|t}^k, m)$, in this project we use Laser Correlation Model. Again, find all cells (Y_{t+1}^k) that will be occupied in this lidar ray, compare them with map and count how many cells are correct with map.

$$corr(Y_{t+1}^k, m) = \sum(y_i == m_i) \quad (31)$$

Then,

$$Ph(z_{t+1}|\mu_{t+1|t}^k, m) \propto corr(Y_{t+1}^k, m) \quad (32)$$

Finally, normalize $a_{t+1|t}^k$

$$normalized\ a_{t+1|t}^k = \frac{a_{t+1|t}^k}{\sum_1^K a_{t+1|t}^k} \quad (33)$$

C. Handeling asynchronous inputs

The data provided isn't synchronized, look this:

Time	0.1	0.2	0.3	0.4	0.5	0.6
Lidar	L_1				L_2	
Encoder		E_1		E_2		
FOG	F_1	F_2	F_3	F_4	F_5	F_6

In this project, I choose lidar timestamp as pivot. Keep predict v and ω until next lidar data time. So, the table will look like:

Time	0.1	0.2	0.3	0.4	0.5	0.6
Lidar	L_1				L_2	
Encoder				$v_{t=0.4}$		
FOG				$\omega_{t=0.4}$		

Following is a simple example for computation:
$v_{t=0.4}$ is velocity prediction based on E_1 and E_2.

$\omega_{t=0.4}$ is angular velocity prediction based on F_1 to F_4.

$$v_{t=0.4} = \frac{\pi d(E2-E1)}{resolution*(0.4-0.2)} \quad (34)$$

$$\omega_{t=0.4} = \frac{F1+F2+F3+F4}{(0.4-0.2)} \quad (35)$$

D. Vehicle Model

I use Differential-drive Kinematic Model as motion model in this project.

$$State\ x = (p, \theta), where\ p = (x, y) \in R^2 \quad (36)$$

$$Control\ u = (v, \omega), where\ v \in R, \omega \in R \quad (37)$$

Euler discretization over time interval of length τ :

$$x_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = f(x_t, u_t) \coloneqq x_t + \tau \begin{bmatrix} v_t \cos(\theta_t) \\ v_t \sin(\theta_t) \\ \omega_t \end{bmatrix} \quad (38)$$

We also need to consider each sensor's pose with respect to vehicle. We use Fig. (1) to derive their poses.

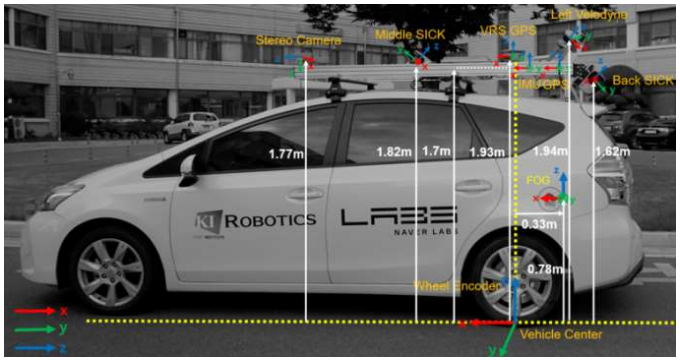


Fig. (1)

For FOG, it has the same orientation as vehicle, so

$$\omega_{FOG} = \omega_{vehicle} \quad (39)$$

For Lidar, we can see it has different orientation with the vehicle. We only have 2D map, so we just need to focus on rotation on z axis. Then:

$$vRs = R(z, -90°) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad (40)$$

To simplify calculation, I move vehicle center to the location of Lidar.

For Encoder, we treat vehicle as rigid body. Therefore, velocity in original center is same as velocity in Lidar's position.

$$v_{vehicle\ center} = v_{Lidar\ center} \quad (41)$$

E. Texture mapping

To compute the probability $p(cm \mid z_{0:t}, x_{0:t})$ requires huge memory; therefore, I choose deterministic way to compute Texture Map.

First, when I perform update step, I choose the particle with highest probability as true robot pose.

Second, based on timestamp from observation data find the closest stereo image as current stereo image.

Finally, using current stereo image to project the color on image into our grid map as texture mapping.

Next, we need to discuss how to perform color projection.

First, we need to convert pixel on image into coordinates in optical frame.

$$\begin{bmatrix} u_L \\ v_L \\ d \end{bmatrix} = \begin{bmatrix} fs_u & 0 & c_u & 0 \\ 0 & fs_v & c_v & 0 \\ 0 & 0 & 0 & fs_u b \end{bmatrix} \frac{1}{z_o} \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix} \quad (42)$$

Then, turn coordinates in optical frame to world frame based on robot pose.

$$oRr \left(rRv * \begin{bmatrix} x_v \\ y_v \\ z_v \end{bmatrix} + rTv \right) = \begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix} \quad (43)$$

$$\begin{bmatrix} x_v \\ y_v \\ z_v \end{bmatrix} = (oRr * rRv)^{-1} * \left(\begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix} - oRr * rTv \right) \quad (44)$$

$(oRr * rRv)^{-1} = vRo$ which is provided (rotation matrix in Vehicle2Stereo.txt).

rTv is provided (translation matrix in Vehicle2Stereo.txt).

$$oRr = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}$$

Now, we have coordinates in vehicle frame; however, our grid map is 2D only. We select specific interval of z_v to specify what point we want to project, then find its coordinates in world frame.

$$wRv * \begin{bmatrix} x_v \\ y_v \end{bmatrix} + wTv = \begin{bmatrix} x_w \\ y_w \end{bmatrix} \quad (45)$$

Set robot pose as $[x_t, y_t, \theta_t]$, then

$$wRv = \begin{bmatrix} \cos(\theta_t) & -\sin(\theta_t) \\ \sin(\theta_t) & \cos(\theta_t) \end{bmatrix} \quad (46)$$

$$wTv = \begin{bmatrix} x_t \\ y_t \end{bmatrix} \quad (47)$$

Finally, we only need to change the physic unit (meter) into cell index and trace back to original image to find the corresponding color, we can color our grid map.

IV. Results

Map parameters:

X : -1500 ~ 1500 (meters) →

Y : -1500 ~ 1500 (meters) ↑

Resolution : 2 meters

Total 1501 * 1501 cells

Vehicle initial pose

$$x_0 = \begin{bmatrix} x_0 \\ y_0 \\ \theta_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

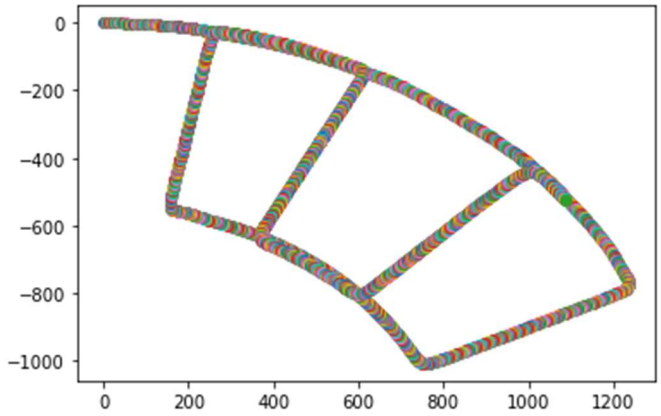
Unit : meters, meters, radian

Others hyperparameters

To speed up calculation, I didn't use all Lidar data. I use 20 rows interval to sample Lidar data.

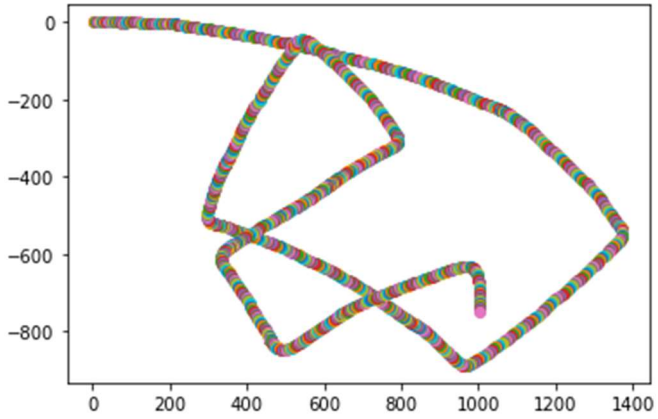
Log-odd for updating map is 4.

A. Motion without noise



We can expect when we use SLAM technique on noised input, the robot needs to move like this picture.

B. Pure motion with noise

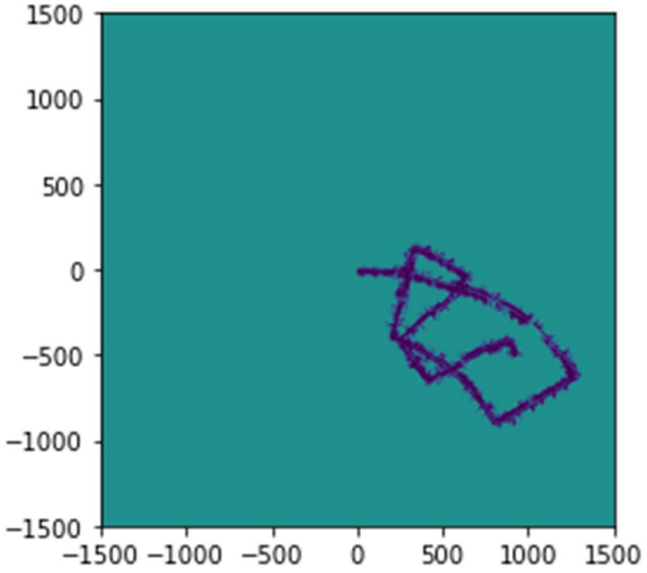


$$v_{noise} = Gaussian(0,0.1)$$
$$\omega_{noise} = Gaussian(0,0.03)$$

We can see that noised inputs make robot move differently. Next apply particle filter to check whether we can eliminate this uncertainty by introducing map and observations.

C. Particle Filter SLAM (1 Particle)

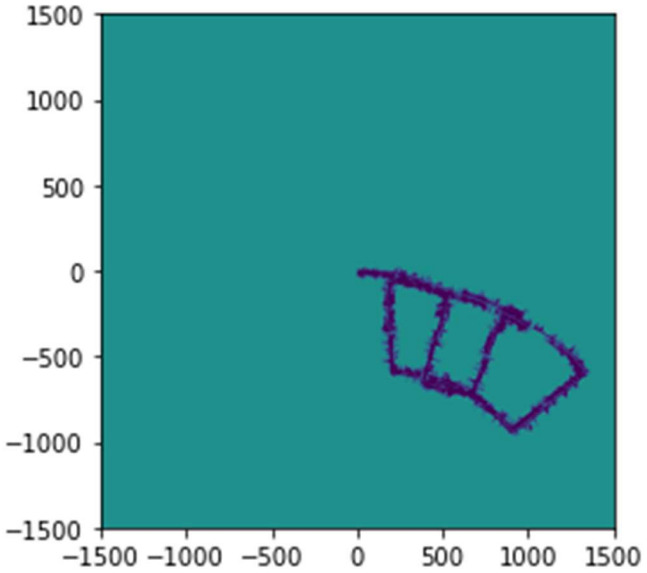
Log-odd map



We can see only one particle; the map isn't built properly.

D. Particle Filter SLAM (2 Particles)

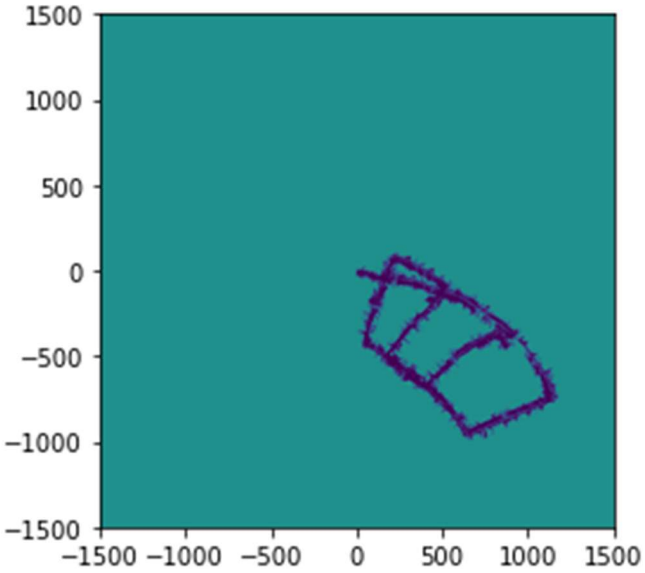
Log-odd map



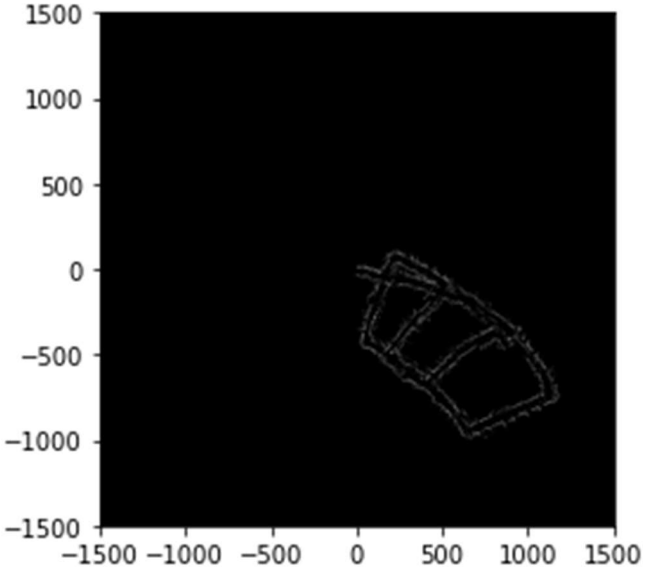
It gets slightly better.

E. Particle Filter SLAM (3 Particles)

Log-odd map

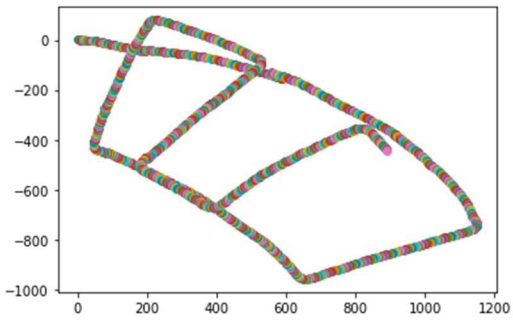


Binary map



*Note: $m_i = \begin{cases} 1, where\ \lambda_i > 0 \\ 0, where \lambda_i \leq 0 \end{cases}$

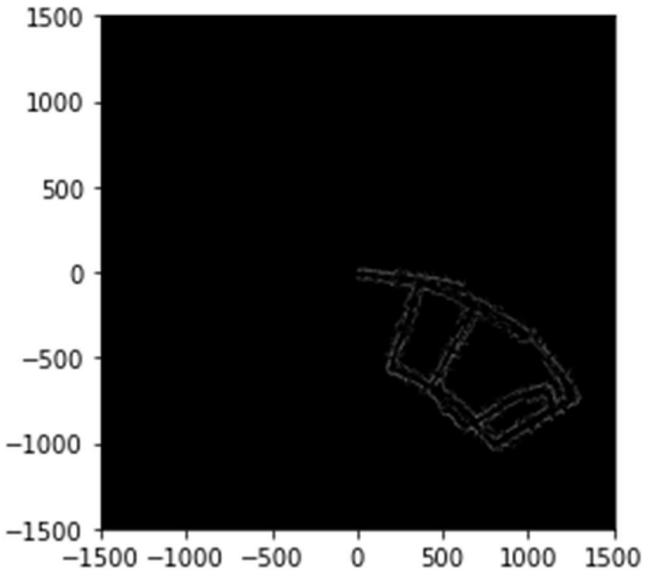
Robot trajectory



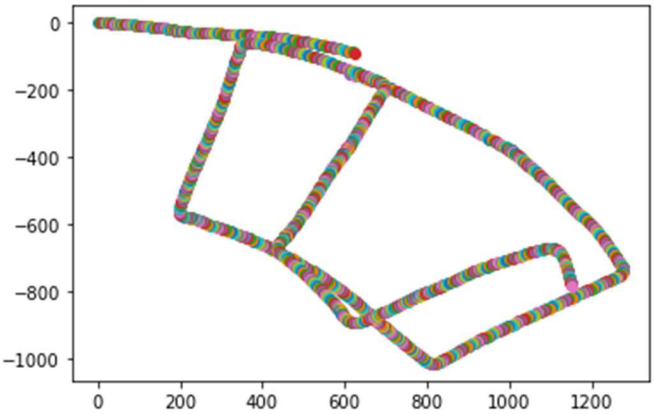
Three particles improve our map a lot, and we can expect with more particles more accurate the map is.

F. Particle Filter SLAM (50 Particles)

Binary map



Robot trajectory

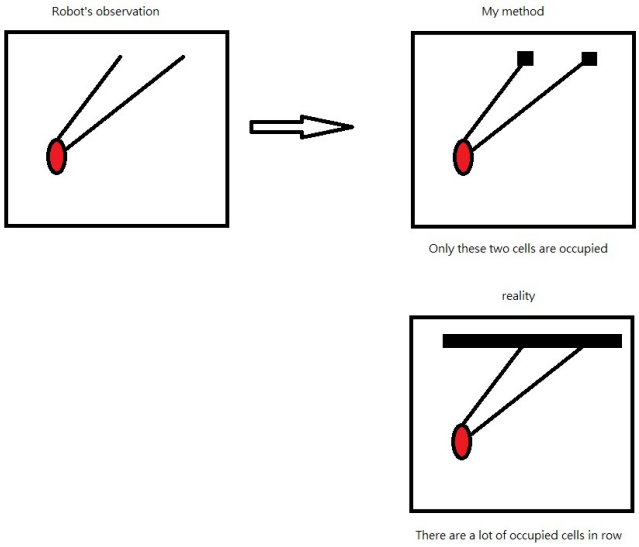


When number of particles increase to 50, there are still some problem.

First, the robot's position jumps at some point. This is because I only use the particle with highest probability to represent the robot. Therefore, when another particle locates

at more accurate position, the robot's position will jump to that particle.

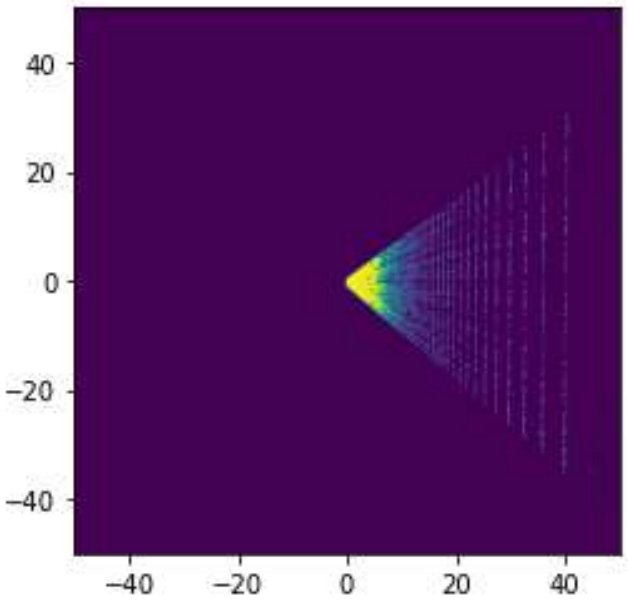
Second, the trajectory of robot isn't accurate at the end. I think this is because my map isn't built rigorously. I only use end point of each lidar ray as obstacle, however, it could not be such case. It might have big influence on mapcorrelation. As illustrated



Conclusion: more particles the map should be more accurate. However, without proper map-correlation function it is more easily get wrong mapping. It seems three particles are suitable for my code here.

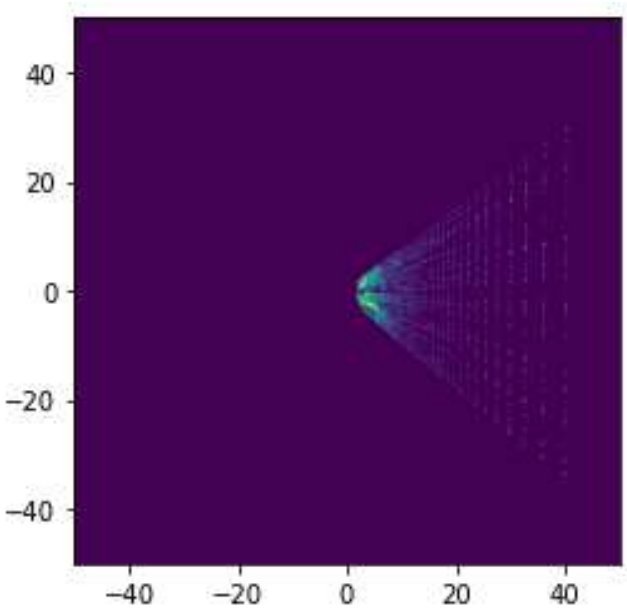
G. Texture mapping

Using robot pose (0,0,0) to test first stereo image, check whether we build the function correctly.



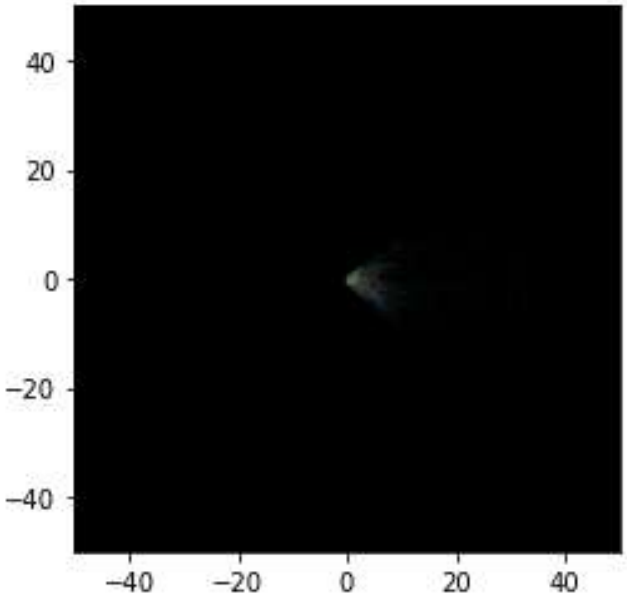
We can see the direction is correct, the function works properly.

Then we set desired z_v interval as [-0.5, 2.5] to see the result.



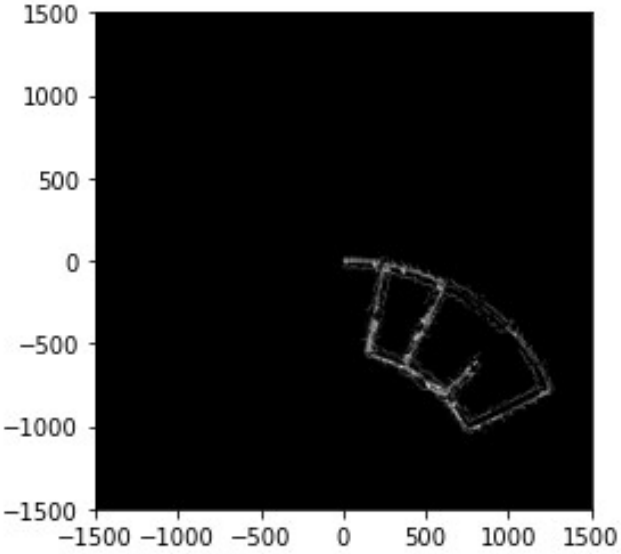
As expected, fewer points are projected.

Set real color to grid, it looks like

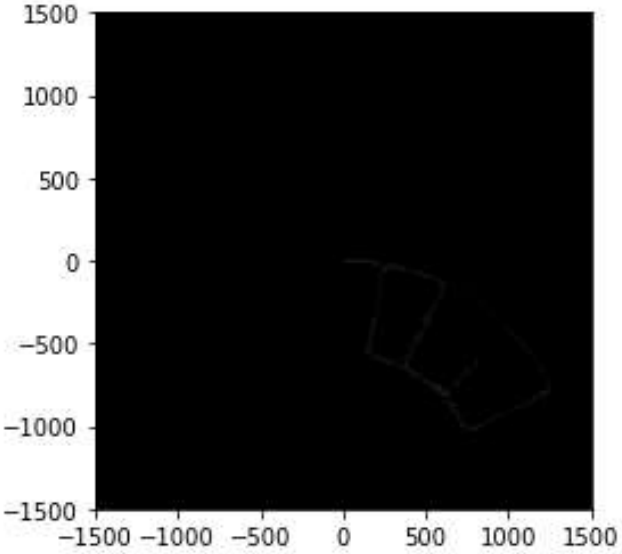


Now, we can run it for the whole map, but to save more time on computation, I only choose one particle and fewer iterations. Therefore, to get correct result I need to set v_{noise} and ω_{noise} 0.

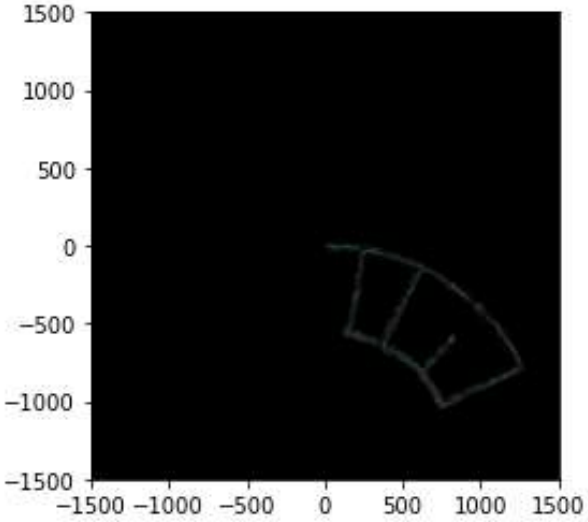
Binary map



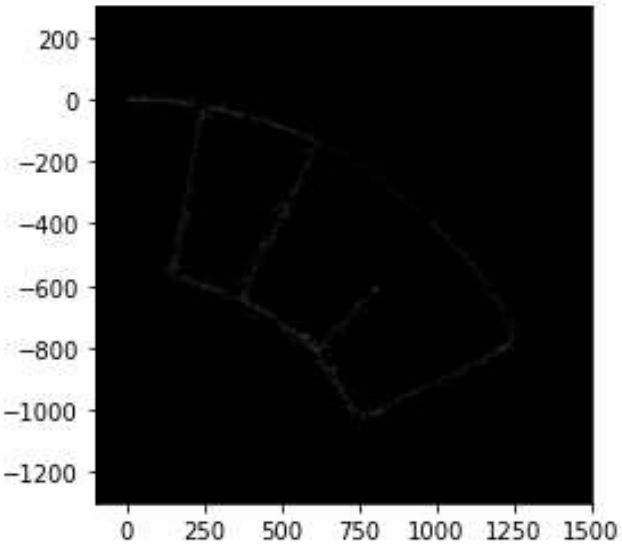
Texture map but only color on occupied cell



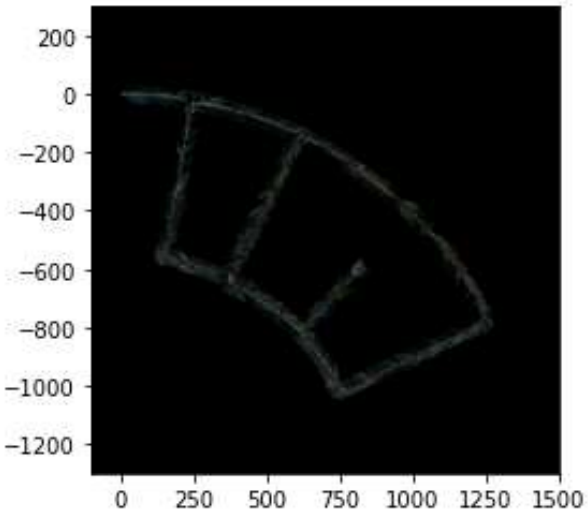
Texture map



Zoom in



Zoom in texture map



V. REFERENCE

1. Professor Nikolay Atanasov's slides (lecture 8~ 10)