Simultaneous Localization and Mapping using a Particle Filter

Zi He
Department of Electrical and Computer Engineering
University of California, San Diego
z1he@ucsd.edu

Abstract-**This is a project review of applying particle filtering to do simultaneous localization and mapping using odometry, inertial, 2-D Lidar scan and RGBD measurements from a differential drive robot. In this project. The IMU, odometer and laser measurements are utilized to localize the robot and build a 2-D occupancy grid map of the environment. The RGBD information is used to color the floor of the 2-D grid map.**

INTRODUCTION

In robotics, 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 agent's location within it [1]. Popular methods to approach this problem include particle filter, Kalman filter and GraphSLAM [2].

Nowadays, self-driving cars, domestic robots etc. are emerging unprecedentedly. Their wide applications greatly improve our life quality. To build such an intelligent robot, first of all it is significant that they are able to know where they are and how the surrounding world is like. SLAM lies at the key of solving this problem.

In this project, the approach I used is particle filtering. It mainly consists of three parts, which are localization prediction, localization update and mapping. First, I use a delta-mixture to represent the pdf of the robot state. In localization prediction, I use a differential drive motion model for the robot. In localization update, I use a laser correlation observation model. By going through these two steps iteratively as time goes on, the particle filter propagates the pdf over time.

PROBLEM FORMATION

Mapping and localization at the same time seems to be a chicken-and-egg problem. In order to build to map, we need to have the robot trajectory while in order to estimate its trajectory, we must be given a map of the environment.

A. Mapping problem

In this project, the map used is called occupancy grid mapping. The map and measurements are both unknown and uncertain. We can maintain a probability density function over the map as $p(m \mid z_{0:t}, x_{0:t})$. The map cells can be seen as independent Bernoulli random variables. Given occupancy measurements $z_{0:t}$, the distribution of cell m_i is:

$$m_i|z_{0:t} = \begin{cases} Ocuupied\ (1) & with\ prob\ \gamma_{i,t} := p\left(m_i = 1 | z_{0:t}, x_{0:t}\right) \\ free\ (0) & with\ prob\ 1 - \gamma_{i,t} \end{cases}$$

To have a probabilistic map representation, a grid of the occupancy probabilities should be kept. Using Bayes Rule, we can obtain that the map distribution γ is updated as follows:

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

Now the question remained is how can we choose an observation model that is suitable for this robot so that we can update our mapping.

B. Localization update problem

Once we have finished the first mapping and we have current particle positions and weights available as $\left(\mu_{t+1|t}^{(i)}, \alpha_{t+1|t}^{(i)}\right)$. It is time to update the particle weights with the current map and current Laser scan Zt+1. The question now is how to build an observation model that can tell us how good our current particle position is when comparing our current mapping with the Laser scan. The quantities need to be defined is a correlation parameter of mapping and Laser scan. Then we can update the particle weights using the following formula.

$$\begin{aligned}p_{t+1|t+1}(x) &= \frac{p_h(z_{t+1} \mid x) \sum_{k=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(k)} \delta\left(x; \mu_{t+1|t}^{(k)}\right)}{\int p_h(z_{t+1} \mid s) \sum_{j=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(j)} \delta\left(s; \mu_{t+1|t}^{(j)}\right) ds} \\ &= \sum_{k=1}^{N_{t+1|t}} \left[\frac{\alpha_{t+1|t}^{(k)} p_h\left(z_{t+1} \mid \mu_{t+1|t}^{(k)}\right)}{\sum_{j=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(j)} p_h\left(z_{t+1} \mid \mu_{t+1|t}^{(j)}\right)}\right] \delta\left(x; \mu_{t+1|t}^{(k)}\right)\end{aligned}$$

Where $p_h\left(z_{t+1} | \mu_{t+1|t}^{(k)}\right)$ is our observation model.

C. Transformation between frames

In the project, many objects have their own coordinate system. For example, the Hokuyo horizontal LIDAR scan is in its own frame. The scan provides us with ranges and angles. It is also located somewhere with respect to the robot according to the robot configuration. So we have to convert from ranges to (x,y) coordinates in the sensor frame, and then to the body frame of the robot and finally to world frame that we defined.

The transformation of the Kinect data, namely RGB camera images and disparity images is more complex compared with transforming the LIDAR ranges. We only have the depth camera pose with respect to the robot center while we do not know where the RGB camera is located. We have to use the following mapping to relate out depth image pixels to RGB image pixels:

$$\begin{aligned}dd &= (-0.00304 * d + 3.31) \\ \mathbf{depth} &= \frac{1.03}{dd} \\ rgbi &= (i * 526.37 + dd * (-4.5 * 1750.46) + 19276.0)/585.051 \\ rgbj &= (j * 526.37 + 16662.0)/585.051\end{aligned}$$

Although we have been provided with the mapping from depth to RGB. We still have to figure out how to transform from image pixel positions to optical frame, then to body frame of the robot and finally to the world frame.

TECHNICAL APPROACH

Generally there are many ways to tackle SLAM. In the project, I use particle filtering with a mixture of delta functions to represent the distribution of the particles (possible positions of the robot). The procedures run as prediction the next possible positions of all particles, update our particle weights with an observation model, do particle resampling to avoid particle depletion, update our mapping and finally when we have got the best particle pose at each timestamp, we can use it to draw a texture map.

A. mapping

Since for our mapping, we use occupancy grid cells. We can do a small trick to simplify our updating of mapping. So instead of pick some observation model and then update the γ distribution with it, we compute the ratio of γ and 1-γ, which give us odds ratio as follows:

$$\begin{aligned} o(m_i \mid z_{0:t}, x_{0:t}) :&= \frac{p(m_i = 1 \mid z_{0:t}, x_{0:t})}{p(m_i = 0 \mid z_{0:t}, x_{0:t})} = \frac{\gamma_{i,t}}{1-\gamma_{i,t}} \\ &= \underbrace{\frac{p_h(z_t \mid m_i = 1, x_t)}{p_h(z_t \mid m_i = 0, x_t)}}_{g_h(z_t \mid m_i, x_t)} \underbrace{\frac{\gamma_{i,t-1}}{1-\gamma_{i,t-1}}}_{o(m_i \mid z_{0:t-1}, x_{0:t-1})} \end{aligned}$$

To even simplify computation, we then introduce log odds by taking logarithm of odds ratio:

$$\begin{aligned} \lambda(m_i \mid z_{0:t}, x_{0:t}) :&= \log o(m_i \mid z_{0:t}, x_{0:t}) = \log\left(g_h(z_t \mid m_i, x_t) o(m_i \mid z_{0:t-1}, x_{0:t-1})\right) \\ &= \lambda(m_i \mid z_{0:t-1}, x_{0:t-1}) + \log g_h(z_t \mid m_i, x_t) \\ &= \lambda(m_i) + \sum_{s=0}^{t} \log g_h(z_s \mid m_i, x_s) \end{aligned}$$

Thus we do not have to specify the observation model that we will use. instead, now we only need to choose a value for the ratio between our observations. For example. We could pick:

$$g_h(1 \mid m_i, x_t) = \frac{p_h(z_t = 1 \mid m_i = 1, x_t)}{p_h(z_t = 1 \mid m_i = 0, x_t)} = \frac{80\%}{20\%} = 4 \qquad g_h(0 \mid m_i, x_t) = \frac{1}{4}$$

And the old odds can then be updated as:

$$\lambda_{i,t+1} = \lambda_{i,t} + \log g_h(z_{t+1} \mid m_i, x_{t+1})$$

B. Localization prediction

For this project, we employ a differential drive motion model for our robot. This model is as follows:

$$s_{t+1} = f(s_t, u_t) := s_t + \tau \begin{pmatrix} v_t \mathbf{sinc}\left(\frac{\omega_t \tau}{2}\right) \cos\left(\theta_t + \frac{\omega_t \tau}{2}\right) \\ v_t \mathbf{sinc}\left(\frac{\omega_t \tau}{2}\right) \sin\left(\theta_t + \frac{\omega_t \tau}{2}\right) \\ \omega_t \end{pmatrix}$$

Please notice that here what we use is a discrete time model. Next, we need to get data about the linear velocity and angular velocity of our robot from Encoder and IMU sensor. The problem here is that there is no such one-to-one correspondence between the timestamps of the two. Someone in the class pointed out that there is a approximately 2s delay in the IMU timestamp.

Here what I did is simply finding the closest two timestamps and then use the data at this timestamp for prediction.

The IMU data is quite noisy due to that a moving robot might have a lot of high frequency variations. So I applied a low-pass first order filter to the IMU data with the highest bandwidth at 10Hz.

C. Localization update

Here we used a Laser Correlation Model, which tells us the similarity between the LIDAR scan and our current mapping. The model is as follows:

$$\mathbf{corr}(y, m) := \sum_i \mathbb{1}\{m_i = y_i\}$$

It is large if the scan agrees with the map. Thus we have our observation model defined as:

$$p_h(z \mid x, m) = \frac{e^{\mathbf{corr}(y,m)}}{\sum_v e^{\mathbf{corr}(v,m)}} \propto e^{\mathbf{corr}(y,m)}$$

D. Texture mapping

Here the model we use for the RGB and depth camera is called Pinhole Camera Model. It assumes that the lens aperture is decreased to zero and all rays are forced to go through the optical center and remain undeflected such that diffraction is dominant. Under this model, the transformation from image pixel positions to optical frame is:

$$\underbrace{\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}}_{\text{pixels}} = \underbrace{\begin{bmatrix} fs_u & fs_\theta & c_u \\ 0 & fs_v & c_v \\ 0 & 0 & 1 \end{bmatrix}}_{\text{calibration: } K} \underbrace{\frac{1}{Z_o} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{canonical projection: } \Pi_0} \begin{pmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{pmatrix}$$

Then using the pose of the camera with respect to the robot origin, we can get the coordinates of a point from optical frame in body frame as:

$$\begin{pmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{pmatrix} = \begin{bmatrix} R_{oc} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_{cw} & p_{cw} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} = \begin{bmatrix} R_{oc}R_{wc}^T & -R_{oc}R_{wc}^T p_{wc} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

Where Roc is the rotation matrix from a regular to an optical frame, which is just to flip the image.

E. Computation efficiency

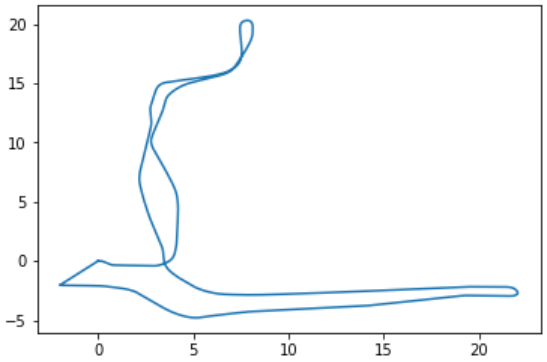
There are nearly 5000 timestamps for Encoder data that we have to iterate over. Also, commonly we have to use a number of particles as around 100 to get a relatively good result. Thus the algorithm can be very slow. Here it is very important to use vectorized computation as much as possible to reduce computation time. For example, during the transformation from LIDAR frame to world frame, and also the transformation from RGB and depth image frame to world frame, vectorized computation saves a lot of time. The whole process can be finished in 20 mins for any of the dataset!

RESULTS

A. Training results

ii. SLAM system over time

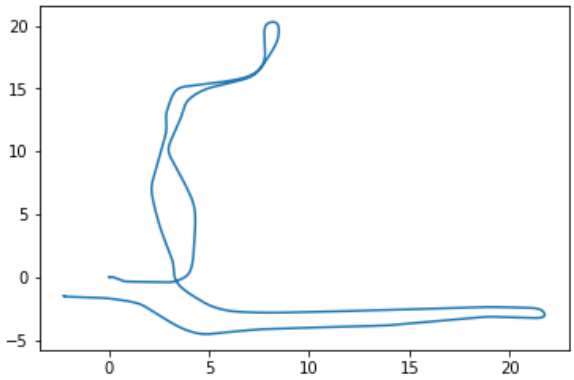
First of all, I tried to plot the robot trajectory using dead reckoning for dataset 20:

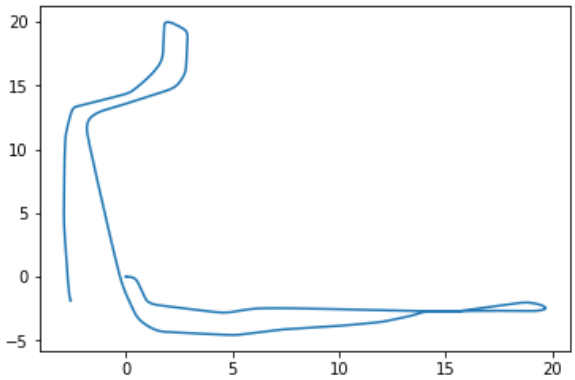


I would like to highlight that I made three GIFs for each of the dataset. Please refer to them to see a real-time SLAM! Below are static robot trajectory, occupancy grid map and texture map with number of particles as 100:

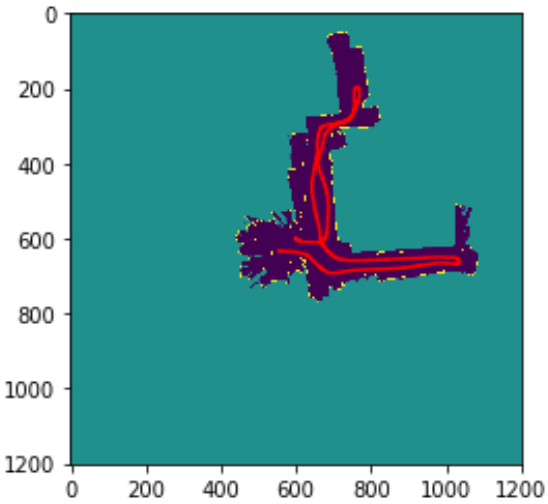
Dataset 20 **Dataset 21**

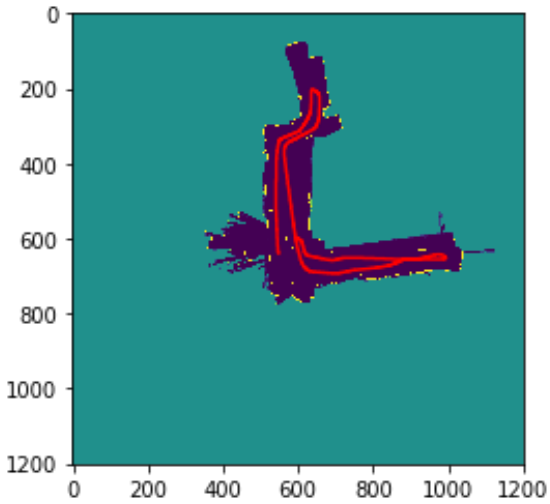
trajectory



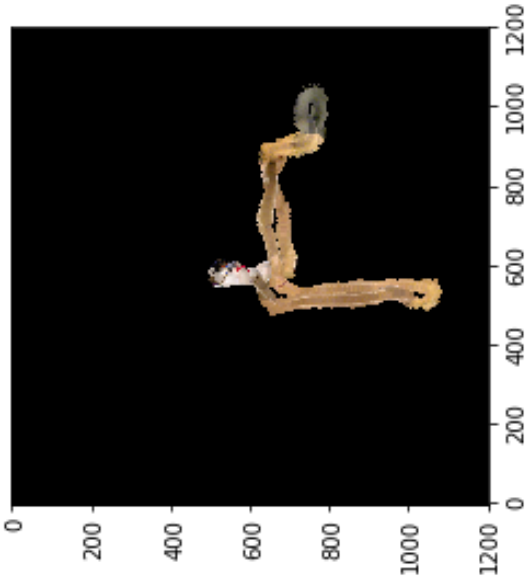


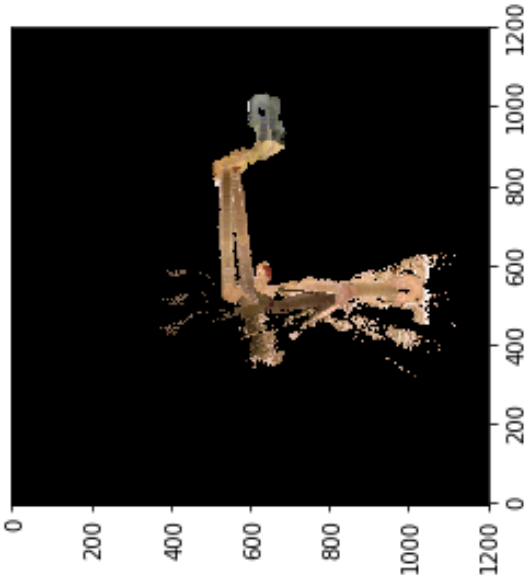
Mapping with trajectory





ii. Texture mapping

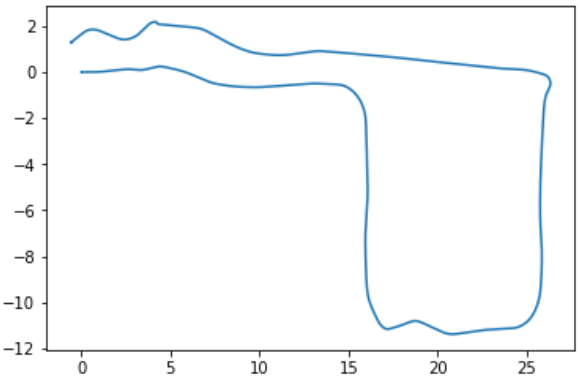




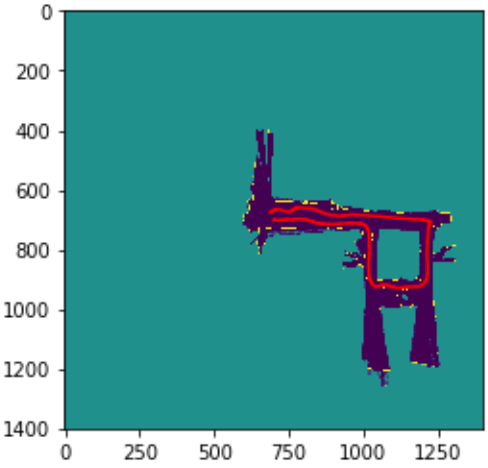
B. Testset

SLAM system over time: please refer to the GIF for real-time SLAM!!!

trajectory



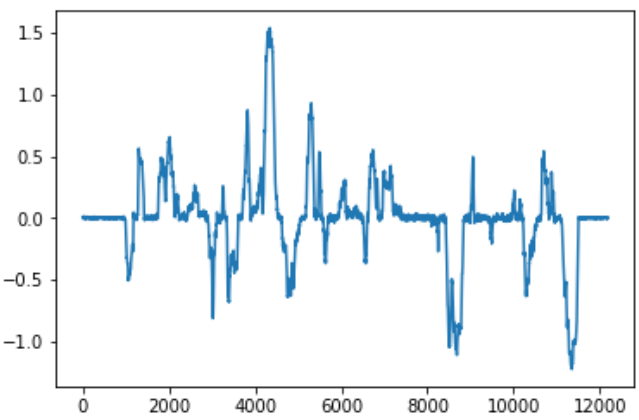
trajectory & mapping



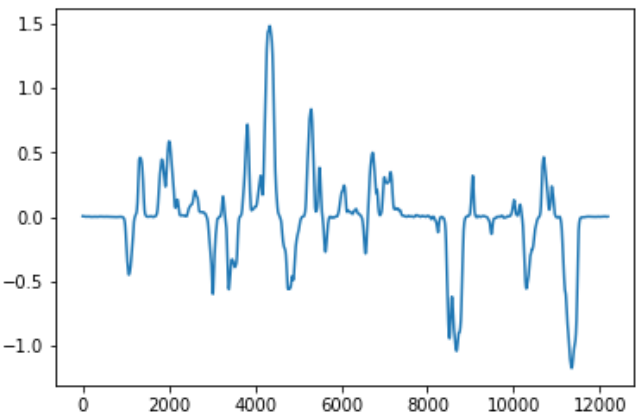
C. Low-pass filtering of IMU data

From the images below, it is clearly that after filtering, we get rid of those high frequency noise from the angular velocity.

Before filtering



After filtering



CONLUSION

In all, the implementation of simultaneous localization and mapping (including texture mapping) has been realized successfully. However, things can still be further improved to achieve better performance due to time limit. For example, adding variations to yaw value when computing the map correlation between current map and Laser scan. This perhaps would give us even more accurate value for the correlation at different particle positions.

REFERENCES

1. Durrant-Whyte, H.; Bailey, T. (2006). "Simultaneous localization and mapping: part I". IEEE Robotics & Automation Magazine. 13 (2): 99–110.
2. https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping