Simultaneous Localization and Mapping (SLAM)

Hsiao-Chen Huang
Department of Electrical and Computer Engineering
University of California San Diego
hsh030@eng.ucsd.edu

***Abstract*—This project aim to solve simultaneous localization and mapping problem by using particle filtering approach. Our goal is to use a differential-drive robot with IMU, odometry, and laser measurements to build a 2-D occupancy grid map of the environment. Also, use the RGBD information from the camera to color the floor of the 2-D map. In this work, we basically followed the Bayes filter probabilistic assumption for estimating the state of dynamic system. Then, try to map the environment from the given state. The results are shown in the final section.**

I. Introduction

One of the approaching in autonomous driving is building the high-definition (HD) map to help recognizing the environment while navigation. For a self-driving fleet to work efficiently, a highly-reliable infrastructure needs to exist for not just building the maps, but also updating the maps. Therefore, localization in real-time is also an important part because we must know where we are while mapping the environment information to our global map.
In this project, we drive through a differential-drive robot with the encoders, IMU, Lidar and RGBD camera in a space. By using measurements obtained by these sensors, we aim to map the environment and localize the robot on the map at each time stamp. For the localization part, we use the particle filtering approach. We can define the bunch of problems into the following outlines:
1) Mapping, update the values in the log-odds map from the given laser scan rays and plot the grid map with occupied/free ceils along each ray. 2) Prediction, update the pose of each particles using the motion model. Use the wheel odometry and the yaw gyro measurements to estimate the location of particles (robot trajectory) before correcting it with the laser readings. 3) Update, use scan matching to correct the robot pose. Given the scans from each particle, we compute the correlation of the each scan and the current map. Update the particle weights and resample the particles. Choose the best particle, project the laser scan, and update the map log-odds 4) Texture map, project colored points from the RGBD sensor onto your occupancy grid in order to color the floor. Find the ground plane in the transformed data via thresholding on the height.

II. Problem Formulation

A. *Mapping*

Given the robot state in the world frame, $x_t = (x, y, \theta) \in \mathbb{R}^3$. We need to build a map of the environment via the sensors' information, such as laser and camera. However, the measurements are obtained by the sensor with the sensor frame. We need to transfer those data into body frame and to world frame. Let B be a body frame whose position and orientation with respect to the world frame W are $p \in R^3$ and $R \in SO(3)$, respectively. The coordinates of a point $s_B \in R^3$ in the body frame B can be converted to the world frame by first rotating the point and then translating it to the world frame: $s_W = R_{s_B} + p$. This transformation is not linear but affine. So, it can be converted to linear by appending 1 to the coordinates of a point s:

$$\begin{bmatrix} s_W \\ 1 \end{bmatrix} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} s_B \\ 1 \end{bmatrix}$$

The trnsformation matrix can thus be described by a matrix

$$SE(3) := \left\{ T := \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} | R \in SO(3), p \in \mathbb{R}^3 \right\} \in \mathbb{R}^{4\times4}$$

Let the transformation from robot to world frame be

$$_{\{W\}}T_{\{B\}} := \begin{bmatrix} _{\{W\}}R_{\{B\}} & _{\{W\}}p_{\{B\}} \\ 0 & 1 \end{bmatrix}$$

And the transformation from sensor to robot frame be

$$_{\{B\}}T_{\{L\}} := \begin{bmatrix} _{\{B\}}R_{\{L\}} & _{\{B\}}p_{\{L\}} \\ 0 & 1 \end{bmatrix}$$

The relative transformation from the sensor frame to the world frame can be represent as

$$_{\{W\}}T_{\{L\}} =_{\{W\}} T_{\{B\}} \times_{\{B\}} T_{\{L\}}$$
$$= \begin{bmatrix} _{\{W\}}R_{\{B\}\{B\}}R_{\{L\}} & _{\{W\}}R_{\{B\}\{B\}}p_{\{L\}} +_{\{W\}} p_{\{B\}} \\ 0 & 1 \end{bmatrix}$$

B. *Localization*

We solve the robot localization problem with particle filter. This is a special case of Bayes filtering. First, we represent the theoretical principle of Bayes filter in the following:
Define t is the discrete time, x_t is robot state, u_t is

control input, z_t is observation, m_t is environment state, w_t is motion noise, v_t is observation noise.
Motion Model:

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

Observation Model:

$$z_t = h(x_t, m_t, v_t) \sim p_h(\cdot|x_t, m_t)$$

Our main problem of localization is try to estimate the robot state that could maximize the posterior density $p_{t+1|t+1}(x)$ over x_{t+1}.

$$p_{t+1|t+1}(x_{t+1}) = p(x_{t+1}|z_{0:t+1}, u_{0:t})$$

To simplify this equation, we need to combine evidence from control inputs and observations using Markov assumptions and Bayes rule. Under Markov assumption, we can rewrite the joint distribution of the state, observation and control input as

$$\begin{aligned} p(x_{0:T}, &z_{0:T}, u_{0:T-1}) = \\ &p_{0|0}(x_0) \prod_{t=0}^{T} p_h(z_t|x_t) \prod_{t=1}^{T} p_f(x_t|x_{t-1}, u_{t-1}) \end{aligned}$$

And the Bayes rule is

$$p(x|y,z) = \frac{p(y|x,z)p(x|z)}{\int p(y,s|z)ds} = \frac{p(y|x,z)p(z|x)p(x)}{p(y|z)p(z)}$$

Hence, the posterior density can be represent as

$$\begin{aligned} &p_{t+1|t+1}(x_{t+1}) = p(x_{t+1}|z_{0:t+1}, u_{0:t}) \\ &= \frac{1}{\eta_{t+1}} p(z_{t+1}|x_{t+1}, z_{0:t}, u_{0:t}) p(x_{t+1}|z_{0:t}, u_{0:t}) \\ &= \frac{1}{\eta_{t+1}} p_h(z_{t+1}|x_{t+1}) p(x_{t+1}|z_{0:t}, u_{0:t}) \\ &= \frac{1}{\eta_{t+1}} p_h(z_{t+1}|x_{t+1}) \int p(x_{t+1}, x_t|z_{0:t}, u_{0:t}) dx_t \\ &= \frac{1}{\eta_{t+1}} p_h(z_{t+1}|x_{t+1}) \int p(x_{t+1}|z_{0:t}, u_{0:t}, x_t) p(x_t|z_{0:t}, u_{0:t}) dx_t \\ &= \frac{1}{\eta_{t+1}} p_h(z_{t+1}|x_{t+1}) \int p_f(x_{t+1}|x_t, u_t) p(x_t|z_{0:t}, u_{0:t-1}) dx_t \\ &= \frac{1}{\eta_{t+1}} p_h(z_{t+1}|x_{t+1}) \int p_f(x_{t+1}|x_t, u_t) p_{t|t}(x_t) dx_t \end{aligned}$$

, where $\eta_{t+1} := p(z_{t+1}|z_{0:t}, u_{0:t})$ is normalization constant.
We can separate the approach into two steps. The **prediction step** is compute the predicted density using our motion model p_f:

$$p_{t+1|t}(x) = \int p_f(x|s, u_t) p_{t|t}(s) ds \tag{1}$$

Given the predicted density, the **update step** obtain the posterior density using the observation model p_h:

$$p_{t+1|t+1}(x) = \frac{p_h(z_{t+1}|x) p_{t+1|t}(x)}{\int p_h(z_{t+1}|s) p_{t+1|t}(s) ds} \tag{2}$$

For particle filter, we use a mixture of delta functions, particles

$$\delta(x; \mu^{(k)}) := \begin{cases} 1 & \cancel{x}= \mu^{(k)} \\ 0 & else \quad \text{for k=1,...,N} \end{cases}$$

with weights $\alpha^{(k)}$ to represent the pdfs $p_{t|t}$ and $p_{t+1|t}$. Then we can rewrite the Eq.1,2 to approximate as delta-mixture pdf

$$\begin{aligned} p_{t+1|t}(x) &= \int p_f(x|s, u_t) \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} \delta(s; \mu_{t|t}^{(k)}) ds \\ &= \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} p_f(x|\mu_{t|t}^{(k)}, u_t) \\ &\approx \sum_{k=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(k)} \delta(x; \mu_{t+1|t}^{(k)}) \end{aligned} \tag{3}$$

$$\begin{aligned} &p_{t+1|t+1}(x) \\ &= \frac{p_h(z_{t+1}|x) \sum_{k=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(k)} \delta(x; \mu_{t+1|t}^{(k)})}{\int p_h(z_{t+1}|s) \sum_{j=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(j)} \delta(s; \mu_{t+1|t}^{(j)}) ds} \\ &= \sum_{k=1}^{N_{t+1|t}} [\frac{\alpha_{t+1|t}^{(k)} p_h(z_{t+1}|\mu_{t+1|t}^{(k)})}{\sum_{j=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(j)} p_h(z_{t+1}|\mu_{t+1|t}^{(j)})}] \delta(x; \mu_{t+1|t}^{(k)}) \\ &= \sum_{k=1}^{N_{t+1|t+1}} \alpha_{t+1|t+1}^{(k)} \delta(x; \mu_{t+1|t+1}^{(k)}) \end{aligned} \tag{4}$$

In each time stamp, we update our particle weights iteratively. In order to avoid particle depletion, a situation in which most of the updated particle weights become close to zero, we applied particle resampling if the effective number of particles $N_{eff} := \frac{1}{\sum_{k=1}^{N_{t|t}} (\alpha_{t|t}^{(k)})^2}$ is less than a threshold.

III. Technical Approaches

Given a differential-drive robot, the training sets contain odometry, intertial, 2-D laser range, and RGBD measurements to generate the prediction of localization position which is represent as $x_t = (x_{rob}, y_{rob}, \theta_{rob}) \in \mathbb{R}^3$. The measurement of 2-D laser readings is $z_t \in \mathbb{R}^{1081}$ with 1081 measurements through -135 to 135 degree rotation at time t. The measurement of odometry encoder readings is $O_t = (FR, FL, RR, RL) \in \mathbb{R}^4$ corresponding to the four wheels and the wheel travels $0.0022m$ per tic. Our world map define as a 1600×1600 grid map $m_t \in \mathbb{R}^{1600 \times 1600}$ with resolution $0.05m$.

A. Mapping

We use Hokuyo UTM-30LX Lidar to obtain our observation measurements. The scan $z_t \in \mathbb{R}^{1081}$ contains 1081 measured ranges in each time stamp t. The range of this model only can scan 30m, so we need to remove scan points that are too close or too far which can be consider as noise or reflection from other laser bins. The horizontal field of view is from -135 to 135 degree of the lidar. Therefore, the rotation of each bin is 0.25 degree. Now, we can transfer lidar coordinate to sensor frame as

$$x_s = r \times \cos\theta$$
$$y_s = r \times \sin\theta$$

where r is the distance from lidar model to the obstacle, and θ is the rotation. To transfer the coordinates from sensor frame to world frame, we multiply with the transformation matrix

$$\begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix} =_w T_{bb} T_l \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix}$$

where

$${}_bT_l = \begin{bmatrix} {}_bR_l & {}_bp_l \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0.146 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$${}_wT_b = \begin{bmatrix} {}_wR_b & {}_wp_b \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_{rob} & -\sin\theta_{rob} & x_{rob} \\ \sin\theta_{rob} & \cos\theta_{rob} & y_{rob} \\ 0 & 0 & 1 \end{bmatrix}$$

We use bresenham2D function to obtain the cell locations $y_{t_{occ}}$ that are occupied and $y_{t_{fre}}$ that are free along the laser ray. Than, we can update our log-odds map

$$m_i = \begin{cases} m_i + 0.9 & \text{i}\in \text{free cell} \\ m_i - 0.7 & \text{i}\in \text{occupied cell} \end{cases}$$

B. Localization

To imply the particle filter, we first initialize $N = 100$ particles $\mu_{0|0} \sim N(x_{0|0}, \sigma^2)$ with weights $\alpha_{0|0}^{(i)} = \frac{1}{N}$, for all $i = 1, ..., N$. And the robot state is $x_{0|0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$.

1) Predict Step: In prediction step, we use the motion model with input from the encoder measurements and the IMU yaw rate and additive noise to compute an updated pose $\mu_{t+1|t}^{(i)}$ for all i particles. For each time stamp, the change of time is τ. We obtain the rotation velocity ω_t from IMU, and use the measurements obtained from encoder $O_t = (FR, FL, RR, RL)$ to compute the velocity v_t of the robot.

$$v_L = \frac{(FL + RL)/2 * 0.0022}{\tau}$$
$$v_R = \frac{(FR + RR)/2 * 0.0022}{\tau}$$
$$v_t = \frac{v_L + v_R}{2}$$

According to the motion model, we can all update particle pose as

$$\mu_{t+1} = \mu_t + \tau \begin{bmatrix} v_t \text{sinc}(\omega_t\tau/2)\cos(\theta_{rob} + \omega_t\tau/2) \\ v_t \text{sinc}(\omega_t\tau/2)\sin(\theta_{rob} + \omega_t\tau/2) \\ \omega_t \end{bmatrix}$$

2) Update Step: In update step, we update the weight α_t of each particle. For each particle, we transform the laser data z_t from the particle pose $\mu_{t+1|t}^{(i)}$ to world frame, and obtain z_t^{world}. Then, compute the correlation between z_t^{world} and current map. We can update the weights as

$$\beta_{t|t}^{(i)} = log(\alpha_{t|t}^{(i)})$$
$$\beta_{t+1|t+1}^{(i)} = \beta_{t+1|t}^{(i)} + correlation(z_{t+1}, map)$$

To keep $\sum_{i=1}^{N} \alpha_{t+1|t+1}^{(i)} = 1$, we need to normalize the weights as

$$\beta_{t+1|t+1}^{(i)} = \beta_{t+1|t+1}^{(i)} - max_j \beta_{t+1|t+1}^{(j)} - log \sum_{k=1}^{N} exp(\beta_{t+1|t+1}^{(k)} - max_j \beta_{t+1|t+1}^{(j)})$$
$$\alpha_{t+1|t+1}^{(i)} = exp(\beta_{t+1|t+1}^{(i))}$$

If $N_{eff} < N_{th}$, we need to do resampling, where $N_{eff} = \frac{1}{\sum_{i=1}^{N} (\alpha_{t+1|t+1}^{(i)})^2}$ and $N_{th} = N * 0.05$. Since some of particles performance are poor, we want to focus on those particles which show better result.

C. Texture Mapping

We try to project color our grid map via the information we obtain from RGBD sensor. First, we need to use a transformation to match the color to the depth. Given the values d at location (i, j) of the disparity image, you can use the following mapping to obtain the depth and associated color at location (u, v):

$$dd = (0.00304 \cdot d + 3.31)$$
$$depth = \frac{1.03}{dd}$$
$$u = \frac{i \cdot 526.37 + dd \cdot (4.5 \cdot 1750.46) + 19276.0}{585.051}$$
$$v = \frac{j \cdot 526.37 + 16662.0}{585.051}$$

Now, we get RGBD frame by setting RGBD mesh grid according the RGBD output matrix size, and inverse the camera matrix to normalized coordinate vector.

$$\begin{bmatrix} X_o \\ Y_o \\ Z_o \end{bmatrix} = Z_o \cdot K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

, where

$$K = \begin{bmatrix} f_{s_u} & f_{s_\theta} & c_u \\ 0 & f_{s_v} & c_v \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 585.05108211 & 0 & 242.94140713 \\ 0 & 585.05108211 & 315.83800193 \\ 0 & 0 & 1 \end{bmatrix}$$

We know that our camera is located at $(0.18, 0.005, 0.36)m$ with respect to the robot center and has rotation $roll = 0rad$, $pitch = 0.36rad$, and $yaw = 0.021rad$. Now, we project the RGBD coordinate from sensor frame to the world frame by multiplying the transformation matrix ${}_wT_o = ({}_oT_w)^{-1}$

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = ({}_oT_w)^{-1} \begin{bmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{bmatrix}$$

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

, where

$$R_{oc} = \begin{bmatrix} cos(0.021) & -sin(0.021) & 0 \\ sin(0.021) & cos(0.021) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times$$

$$\begin{bmatrix} cos(0.36) & 0 & sin(0.36) \\ 0 & 1 & 0 \\ -sin(0.36) & 0 & cos(0.36) \end{bmatrix}$$

$$R_{wc} = \begin{bmatrix} cos(\theta_{rob}) & -sin(\theta_{rob}) & 0 \\ sin(\theta_{rob}) & cos(\theta_{rob}) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$p_{wc} = \begin{bmatrix} x_{rob} + 0.18 \\ y_{rob} + 0.05 \\ 0.36 \end{bmatrix}$$

After getting the transfer position, we assign the original RGB data in (i, j) to the world frame position (X_w, Y_w, Z_w). We obtain the color of ground plane by thresholding the height $Z_w < 5$, and color the cells in our texture map.

IV. Results

We first use the motion model to predict the robot pose with a single particle. It is to make sure our motion model is accurate. The result is shown in Fig.1. We can see that there are some miss-predicted path in the right-below part of the environment.

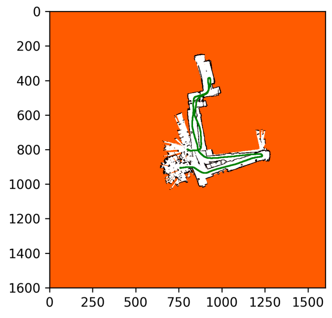

Fig. 1. The map of motion update

We go through each time stamps to predict and update the robot location and the map. The result of the map is shown in Fig.2. The miss-predicted path has been improved, and the end point go back to the start point which performs well. The texture mapping result is shown in Fig.3.

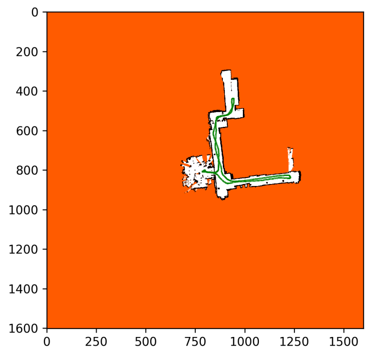

Fig. 2. The final result of dataset20 with SLAM.

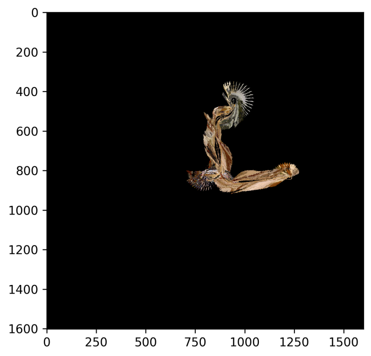

Fig. 3. The texture map of dataset20 with SLAM.

The results of another datasets are shown in Fig.4-6. The performance is worse than our first dataset, and we can not match the start point and end point. We may need to adjust the parameters to get a better performance.

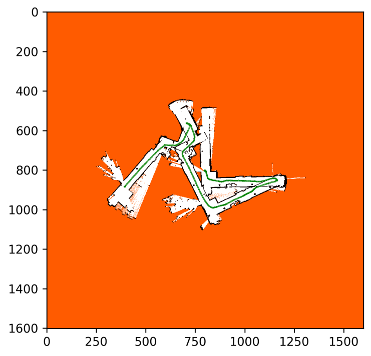

Fig. 4. The final result of dataset21 with SLAM.

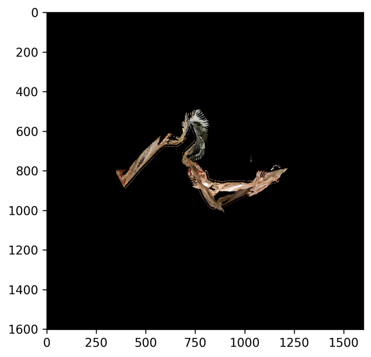

Fig. 5. The texture map of dataset21 with SLAM.

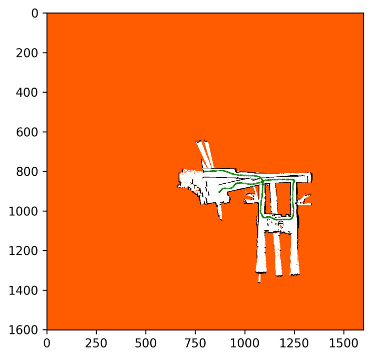

Fig. 6. The final result of test dataset with SLAM.