Particle Filter SLAM

Huafeng mai
University of Sandiego
La Jolla, CA 92122
Email: humai@ucsd.edu

***Abstract*—For robotics, such as cleaning robots, autonomous driving, and harvesting robots, it is necessary to calculate the current position in real time and avoid encountering obstacles. Therefore, the robot needs to calculate the current position and surrounding maps at the same time. So this paper try to solve this problem based on pricinple of SLAM and implemention through particle filter**

I. Introduction

Robots are involved in many applications, cleaning robots, autonomous driving, and harvesting robots. To complete these tasks, the robot needs to obtain surrounding information through sensors to determine the correct orientation. The SLAM problem is about updating the mapping and its own positioning parameters in an unknown environment [1]. Positioning is to determine the position and orientation of the robot, and at the same time mapping is mapping environment. Because for acquiring a map, a robot requires a good estimate of its location and for localization and needs a consistent map. Correct estimation of localization or trajectory of a robot need a accurate map and the robot's environment can only be obtained given accurate positioning information [2]. Therefore, this problem becomes difficult to solve. So in order to solve this problem, this paper is based on the method of particle filter to achieve this goal [3], Each particle has an associated weight, based on its current position, that describes its confidence in its current estimate based on observations. Use the weighted sum of these particles to approximate the posterior probability density.

The paper structure is as follows. Give the detailed mathematics formulations of SLAM problem in Section II. Technical approaches are introduced in Section III. And at last setup the algorithm, results are presented in Section IV.

II. PROBLEM FORMULATION

Think of slam problem as Markov Chain, According to the map m and the trajectory x1:t = x1, . . . , xt of the robot. This estimation is performed given the observations o1:t = o1, . . . , ot and the robot inputs u1:t1 = u1, . . . , ut1 obtained by the mobile robot. n stand for the motion noise.

$$\mathbf{x}_t = f\left(\mathbf{x}_{t-1}, \mathbf{n}_{t-1}, \mathbf{u}_{t-1}\right) \tag{1}$$

For mapping the surroundings environment. Denote the observation at time t as o_t

$$\mathbf{o}_t = g\left(\mathbf{x}_t, \mathbf{m}, \mathbf{v}_t\right) \tag{2}$$

The probabilistic function can be written in the following.

$$p\left(\mathbf{m}, \mathbf{x}_t \mid \mathbf{o}_{1:t}, \mathbf{u}_{1:t-1}\right) \tag{3}$$

Use o and u to estimate the distribution of x and m. posterior decomposition of probability function according to Markov principle.

$$= p\left(x_{1:t} \mid o_{1:t}, u_{1:t}\right) p\left(m \mid x_{1:t}, o_{1:t}\right) \tag{4}$$

$$\approx p\left(\boldsymbol{x}_t \mid o_{1:t}, u_{1:t}\right) p\left(\boldsymbol{m}_t \mid \hat{\boldsymbol{x}}_t, o_{1:t}, u_{1:t}\right) \tag{5}$$

find the optimal $x_{1:t}$ and m in order to determine the environment and the pose of the robot. estimate $p\left(\boldsymbol{x}_t \mid o_{1:t}, u_{1:t}\right)$using the localization step, while $p\left(\boldsymbol{m}_t \mid \hat{\boldsymbol{x}}_t, o_{1:t}, u_{1:t}\right)$ is estimated via log-odds mapping. The posterior of $p\left(m \mid x_{1:t}, o_{1:t}\right)$ can be computed analytically with the known pose But to estimate the posterior $p\left(x_{1:t} \mid o_{1:t}, u_{1:t-1}\right)$, a particle filter can be applied. Each particle initialized represents the individual potential trajectory of the robot. Furthermore, a separate map is associated with each sample. think as a different robot, map is constructed from observations and trajectories represented by corresponding particle being published.

$$\hat{\boldsymbol{x}}_t = \operatorname{argmax}_{\boldsymbol{x}_t} p\left(\boldsymbol{x}_t \mid o_{1:t}, u_{1:t}\right) \tag{6}$$

Need to maximize the posterior pdf of the parameters given the data. The mainly data reveive from below source
(1) **Encoders** helps in determining the **current robot position**.
(2) **Lidar** help to collect and build the **map information**.
(3) **FOG** to encoder timestamps each FOG corresponding to each encoder reading

III. TECHNICAL APPROACH

The SLAM problem can be solved by using Particle Filter. n particles are initialized with weights 1/N.

$$\delta = \begin{cases} 1 & x = \mu^{(i)} \\ 1, & \text{other case} \end{cases} \tag{7}$$

function gives the positions of the particles. initially for each particles is 1/n. for k from 1 to n

A. Transformation

Need to transform lidar data from polar coordinates to cartesian coordinates and from lidar frame to vehicle frame, then transform it to world frame. The relationship between body-frame coordinates s_B and world-frame coordinates s_W

$$\begin{bmatrix} \mathbf{s}_W \\ 1 \end{bmatrix} = \begin{bmatrix} R & \mathbf{p} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{s}_B \\ 1 \end{bmatrix}$$

The rigid body represent matrix, where p_{wb} is the position of target object, and R_{wb} stand for the transformation from body frame to the world frame.

$$T_{wb} = \begin{bmatrix} R_{wb} & p_{wb} \\ 0 & 1 \end{bmatrix}$$

B. Particle Filter

Particle filter methods uses particles to express stochastic process posterior distribution within partial observations or noisy. initialized with weights

$$\gamma = \begin{cases} 1 & x = \mu^i \\ 1, & \text{else} \end{cases} \tag{8}$$

This equation gives the positions of particles, the distribution funciton is below.

$$x_t \mid o_{0:t}, u_{o:t-1} \sim p_{t|t}\left(x_t\right) = \sum_{k=1}^{n_{t|t}} \alpha_{t|t}^i \gamma\left(x_t; \mu_{t|t}^i\right) \tag{9}$$

o: t-1 mean the time from 0 to t-1, t is time, x is state, o is obseration, u is control input.

C. Prediction method

compute the predicted density over x, $p_{t|t}$ is based on the prior density.

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

D. Update method

$$p_{t+1|t+1}(\mathbf{x}) = \frac{p_h\left(\mathbf{o}_{t+1} \mid \mathbf{x}\right) p_{t+1|t}(\mathbf{x})}{\int p_h\left(\mathbf{o}_{t+1} \mid \mathbf{s}\right) p_{t+1|t}(\mathbf{s}) d\mathbf{s}}$$

ph is the observation model obtain the posterior and contain the measurement information

E. Algorithm step of prediction and update

1. Initial a number of particles around robot position, current is $(x, y, yaw) + \mathrm{rand}(x, y, yaw)$ at time = 0
2. Get motion model:moveforward =(leftwheel+rightwheel)/2
x=x+moveforward*cos(yaw),y=y+moveforward*sin(yaw)
yaw= yaw + d(yaw),d(yaw) is from fog data.
3. For each particle, make a prediction using motion model
4. For each particle, use mapCorrelation in pr2-util.py to update its weight and position.(I am still confusing about how to use the result from mapCorrelation). It's worth to mention that, all information in mapping are used to update particles.
5. Particle weight normalization and set the robot pose as the weighted sum of the particles
6. Remove the particles which weight is smaller than the threshold and initialize new particles near the robot pose.
7. New loop.

F. Resampling

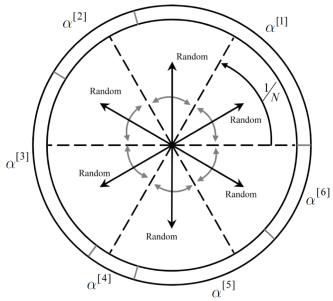
When resampling, the weight vector is set to $\frac{1}{n}$. The should be less. If it's too large, resampling is done really frequently and at any given time, there will be a few set of particles in the cluster Resampling method is following the below equation, effective particles is show by

$$N_{effective} = \frac{1}{\sum_{i=1}^{N} \left(\alpha^i\right)^2} \tag{10}$$

At time t, applied the resampling, less than it threshold is the effective number of particle. particles is less than a threshold, if less than a threshold, then create a new set. below is stratified resampling pseudo code.

Stratified (low variance) resampling

1: **Input**: particle set $\left\{\boldsymbol{\mu}^{(k)}, \alpha^{(k)}\right\}_{k=1}^{N}$
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 $\left(\boldsymbol{\mu}^{(j)}, \frac{1}{N}\right)$ to the new set



G. Process lidar function

1.Transform lidar points from polar coordinates to cartesian coordinate.
2.Transform lidar points from lidar frame to vehicle frame, then transform it to world frame.
3.Use bresenham2D in pr2-util.py to label the empty and occupied grids in map. Notice that this is a sum of lambda in a grid, we need to transform probability to lambda(log4 in picture), then sum up the lambda in the grid, then transform into probability.

H. Mapping

The mapping function requires four inputs: **1.**Map resolution **2.** Robot resolution **3.** old map **4.** Scan data in the world frame W, and finally filter out the data that is too close by 0.1m or too far by 80m. Since too close will make points that hit the car body and too far will introduce noise into the map without improving the quality. Grid map is initialized with ones.

I. Occupancy Grid Map

Occupancy grid mapping is to represent the around environment by dividing it into many cells. it have the purpose to record the state of each map cell are occupied or not, use as 1 and -1 to represent and use the log odd map to represent and since the environment is unpredictable so we need to use

probabilities to estimate each grid cell. So each grid cell value have a pmf conditioned with the sensor observations $o_{0:t}$ and robot's state $x_{0:t}$, since each cell pmf is independent, so the tracked process is individually for each cell.

$$p\left(\mathbf{m} \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}\right) = \prod_{i=1}^{n} p\left(m_i \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}\right) \tag{11}$$

Given the measurement $o_{0:t}$, the distribution of m_i is

$$m_i \mid 0_{0:t} = \begin{cases} \text{Occupied } (1) \\ \text{not Occupied } (0) \end{cases} \tag{12}$$

Occupied with probability $p\left(\mathbf{m} = \mathbf{1} \mid \mathbf{o}_{0:t}, \mathbf{x}_{0:t}\right)$ and not Occupied with probability 1-$p\left(\mathbf{m} = \mathbf{1} \mid \mathbf{o}_{0:t}, \mathbf{x}_{0:t}\right)$, suppose $\delta_{i,t} = p\left(m_i = 1 \mid o_{0:t}, x_{0:t}\right)$, inorder to update $\delta_{i,t}$, Using Bayes rule.

$$\begin{aligned} &= \frac{1}{\eta_t} p_h\left(o_t \mid x_t, m_i = 1\right) p\left(m_i = 1 \mid o_{0:t-1}, x_{0:t-1}\right) \\ &= \frac{1}{\eta_t} p_h\left(o_t \mid x_t, m_i = 1\right) \delta_{i,t-1} \\ &\left(1 - \delta_{i,t}\right) = p\left(m_i = 0 \mid o_{0:t}, x_{0:t}\right) \\ &= \frac{1}{\eta_t} p_h\left(o_t \mid x_t, m_i = 0\right) \left(1 - \delta_{i,t-1}\right) \end{aligned} \tag{13}$$

Since the algorithm follow the Bayes rule and the purpose of the Grid mapping is to keep track a vector of occupancy probabilities δ_t, we can update the map through log-odds ratio. So need to define odds ratio of a binary random variable m_i

$$\begin{aligned} o\left(m_i \mid o_{0:t}, x_{0:t}\right) &:= \frac{\delta_{i,t}}{1 - \delta_{i,t}} \\ &= \frac{p_h\left(o_t \mid m_i = 1, x_t\right)}{p_h\left(o_t \mid m_i = 0, x_t\right)} \frac{\delta_{i,t-1}}{1 - \delta_{i,t-1}} \\ &= o\left(m_i \mid o_{0:t-1}, x_{0:t-1}\right) g_h\left(o_t \mid m_i, x_t\right) equation \end{aligned} \tag{14}$$

we can take the log for both side of the

$$\begin{aligned} \lambda_{i,t} &= \log \frac{p\left(\mathbf{m}_i = 1 \mid \mathbf{o}_{0:t}, \mathbf{x}_{0:t}\right)}{p\left(\mathbf{m}_i = -1 \mid \mathbf{o}_{0:t}, \mathbf{x}_{0:t}\right)} \\ &= \frac{\delta_{i,t-1}}{1 - \delta_{i,t-1}} \log \frac{p\left(\mathbf{o}_t \mid \mathbf{m}_i = 1, \mathbf{x}_{0:t}\right)}{p\left(\mathbf{o}_t \mid \mathbf{m}_i = -1, \mathbf{x}_{0:t}\right)} \\ &= \log g_h\left(\mathbf{o}_t \mid \mathbf{m}_i, \mathbf{x}_t\right) + \lambda\left(\mathbf{m}_i \mid \mathbf{o}_{0:t-1}, \mathbf{x}_{0:t-1}\right) \\ &= \sum_{s=0}^{t} \log g_h\left(_i \mid \mathbf{m}_i, \mathbf{x}_i\right) + \lambda\left(\mathbf{m}_i\right) \end{aligned} \tag{15}$$

this the update method for log-odds and the grid map use thresholding can be derived from log-odds, where $g_h\left(\mathbf{o}_s \mid \mathbf{m}_i, \mathbf{x}_s\right)$ is reliability of the observation

J. Texture Mapping

For texture mapping, need to create and update the texture map after we choose the best particle and using this the position of particles and the disparity images. the rgb values of image can be projected onto an occupancy grid map as 3 dimensional vector since the normal 3d space and the axes of a image are defined differently. so, need to convert the optical coordinated to a regular frame. calculate disparity from particle's pose have largest weight. The particles are represented by below equation.

IV. Results

Unable to display final texture map result due to the time constraint.

References

[1] Beril Sirmaccek, Nicolò Botteghi, and Mustafa Khaled, "Reinforcement learning and slam based approach for mobile robot navigation in unknown environments," in ISPRS Workshop Indoor 3D 2019, 2019.

[2] Michael Montemerlo. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, July 2003.

[3] Törnqvist, David, Thomas B. Schön, Rickard Karlsson, and Fredrik Gustafsson. "Particle filter SLAM with high dimensional vehicle model." Journal of Intelligent and Robotic Systems 55, no. 4 (2009): 249-266.