Orientation Tracking and Panorama Generation

1st Pengxi Zeng
La Jolla, CA
p2zeng@ucsd.edu

Abstract—This is a report regarding orientation tracking and corresponding panorama generation. It mainly discussed how to define the motion model and observation model and utilize the IMU measurements to optimize the rotation quaternions. Gradient descent is adopted as the main method for optimization and a combianation of projection-based and normalization- based optimization metrics are used to constrain the quaternions to space $\mathbb{H}_*$. The result turns out good for most datasets. Finally, the panorama is generated using sphere-cylinder projection by first transforming the spherical coordinates of each pixel of the picture from camera frame to world frame. The final outcome shows great consistency in space, also indicating good effects of the optimization for the quaternions.

Index Terms—orientation tracking, panorama, gradient descent

I. Introduction

SLAM, also known as simultaneous localization and mapping, is a technique utilized by autonomous vehicles to simultaneously create a map and determine the location of the vehicle within that map. By utilizing SLAM algorithms, the vehicle can map out unfamiliar environments, allowing engineers to perform tasks like avoiding obstacles and planning path using the mapped data. In the SLAM process, the robot repeatedly acquires data from sensors, such as cameras or lidars, and uses this data to estimate its own position and to create a map of the environment. The specific implementation of the SLAM algorithm can vary depending on the sensors and hardware used by the robot, as well as the specific application and environment in which it operates. Usually due to the noise of the data collected, we will adopt methods relying on probabilistic reasoning to optimize the process.

II. Problem Formulation

A. Particle-filter SLAM

Denote t as time, $\boldsymbol{x}_t$ as the robot state at time t, $\boldsymbol{u}_t$ as the control input at time t, $\boldsymbol{z}_t$ as the observation at time t and $\boldsymbol{m}_t$ as the map state at time t. According to Markov assumption, we assume that state $\boldsymbol{x}_{t+1}$ only depends on the previous state $\boldsymbol{x}_t$ and the previous input $\boldsymbol{u}_t$ and the observation $\boldsymbol{z}_{t+1}$ only depends on the state $\boldsymbol{x}_t$. Every time, we need to estimate the state of the robot using the motion model and observation model. The general steps are dividede into two:

- Predict: to use noised motion model to estimate the probability of the robot state $\boldsymbol{x}_{t+1}$.
- Update: to use the map data and observation to update the probability of the robot state $\boldsymbol{x}_{t+1}$.

The process of the predicting and updating is called Bayes filter and could be parametrized in a iterative form:

$$p_{t+1|t}(\boldsymbol{x}) = \int p_f(\boldsymbol{x}|\boldsymbol{s},\boldsymbol{u}_t)p_{t|t}(\boldsymbol{s})d\boldsymbol{s}, \tag{1}$$

$$p_{t+1|t+1}(\boldsymbol{x}) = \frac{p_h(\boldsymbol{z}_{t+1}|\boldsymbol{x})p_{t+1|t}(\boldsymbol{x})}{\int p_h(\boldsymbol{z}_{t+1}|\boldsymbol{s})p_{t+1|t}(\boldsymbol{s})d\boldsymbol{s}}. \tag{2}$$

Considering that the probability space is infinite, it is impossible to implement the continuously represented function. Thus, we adopt an approximation of the equations: particle filter which descretize the calculation. We could choose N particles to represent a finite number of the possible states. Thus, we could rewrite the predicting and updating steps as following:

$$\begin{aligned} &p_{t+1|t}\left(\boldsymbol{x}_{t+1}\right) \\ &= \int p_f\left(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{u}_t\right) \sum_{k=1}^{N} \alpha_{t|t}[k] \delta\left(\boldsymbol{x}_t - \boldsymbol{\mu}_{t|t}[k]\right) d\boldsymbol{x}_t \\ &= \sum_{k=1}^{N} \alpha_{t|t}[k] p_f\left(\boldsymbol{x}_{t+1} \mid \boldsymbol{\mu}_{t|t}[k], \boldsymbol{u}_t\right), \end{aligned} \tag{3}$$

$$\begin{aligned} &p_{t+1|t+1}\left(\boldsymbol{x}_{t+1}\right) \\ &= \sum_{k=1}^{N}\left[\frac{\alpha_{t+1|t}[k] p_h\left(\boldsymbol{z}_{t+1} \mid \boldsymbol{\mu}_{t+1|t}[k]\right)}{\sum_{j=1}^{N} \alpha_{t+1|t}[j] p_h\left(\boldsymbol{z}_{t+1} \mid \boldsymbol{\mu}_{t+1|t}[j]\right)}\right] \\ &\delta\left(\boldsymbol{x}_{t+1} - \boldsymbol{\mu}_{t+1|t}[k]\right), \end{aligned} \tag{4}$$

where $\boldsymbol{\mu}[k]$ denotes the state of the k-th robot at time t and $\alpha[k]$ the probability. In the predicting step, the weights of the particles remain unchanged and in the updating step, the states of the particles remain unchanged.

B. Texture Map

III. Technical Approach

A. Predict

Since during the predicting, the weight of the particle remains still, this step simply changes to use motion model $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{w}_t)$ to estimate the state of next timestamp, where $\boldsymbol{w}_t$ denotes the input noise. Considering that the IMU data and encoder data are seprately collected, we also need to compute them seprately and combine them by aligning the timestamps. Knowing that the IMU collects data more frequently, we choose the encoder's timestamps as the base timeline. To estimate the pose along encoder timestamps, I adopted a metric to choose the best-match time interval of IMU, that is: for adjacent two encoder data, we denote their time stamps as t_i^e and t_{i+1}^e, we iterate through the IMU time stamps to find the timestamp t^{imu}'s index j and k, which satisfy that $t_j^{imu} \leq t_i^e \cap t_{j+1}^{imu} > t_i^e$, $t_k^{imu} \leq t_{i+1}^e \cap t_{k+1}^{imu} > t_{i+1}^e$ and $k \geq j$. Then we could split the time interval $t_i^e \sim t_{i+1}^e$ to multiple time spans and sum over the position translation using the same velocity while with correpsonding poses.

B. Update

After the predicting step, we could update each particle's weight using map correlation, which is simply counting the number of matching obstacles. An alternative metric is that we not only measure the given location's correlation but a range around given location, and find the maximum point and renew both the particle's location and weight [1]. For each particle, we multiply its weight with the correlation. After that, we normalize the particles' weights, and check the number of effective particles $N_{eff} = 1/(\sum_i \alpha_i^2)$. If N_{eff} is lower than a set threshold, we need to resample the particle before continuing the process again.

C. Renew the Occupancy Map

The policy is quite straightforward. After updating, we could pick a particle with the largest likelihood to renew the log odds of the occupancy map *odds*. We first transform the lidar scan from lidar frame to the world frame, and then use ray-tracing algorithm bresenham to find the free cells and occupied cells. For each cell covered by the rays described by bresenham, we count the times that it is denoted as free cells and the times that it is denoted as occupied cells, from which we could obtain two matrixes $count_{free}$ and $count_{occ}$ both of the same size as the occupancy map. Then the log odds of the occupancy map could be renewed as $odds = odds + 2 \times count_{occ} \times \log(4) - count_{free} \times \log(4)$[2].

D. Panorama Generation

IV. Results

A. Orientation Tracking

From the results, we can find out that for most of the datasets, angles of Roll and Pitch match with real data realtively well, but for Yaw, the shape of optimized results match with that of the real data, while the is a gap between them. One guess is that the accumulation of the error in Yaw could not be well eliminated.

[1]This step is equivalent to adding the number of the particles. However considering that the every time we will calculate a quite large area, say 5×5, so to achieve a similar effect, we may need to scale the number of the particles by 25 or even more (squarely related), and that is a great cost of time.

[2]In the equation, I adopted different weights for occupied cells and free cells, because I think we should be more certain about the obstacles while less certain about free cells from the lidar scan.

B. Panorama

From the results we can tell that the pictures have the consistency in space, e.g. the connection of the handrail.