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.

III. Technical Approach

A. Orientation Tracking

Since it is nearly not possible to find the closed solution to the problem, we adopt gradient descent to optimize the quaternions step by step.

Optimization: For any quaternion $\boldsymbol{q}_t, \forall t \in [1, T]$, the optimization from step k to $k+1$ could be formulated as:

$$\boldsymbol{q}_t^{k+1} = \boldsymbol{q}_t^k - \alpha \frac{\partial c(\boldsymbol{q}_{1:T})}{\partial \boldsymbol{q}_t^k}, \forall t \in [1, T] \tag{5}$$

However, if adopting this approach alone would lead to invalid quaternions ($\boldsymbol{q}_t \notin \mathbb{H}_*$). Thus, we need validation step to modify the optimization step, in order to satisfy the constraint.

Validation There are two ways to achieve optimization validation. One is to directly adopt normalization, which is:

$$\boldsymbol{q}_t^{k+1} = \frac{\boldsymbol{q}_t^k - \alpha \frac{\partial c(\boldsymbol{q}_{1:T})}{\partial \boldsymbol{q}_t^k}}{||\boldsymbol{q}_t^k - \alpha \frac{\partial c(\boldsymbol{q}_{1:T})}{\partial \boldsymbol{q}_t^k}||_2}. \tag{6}$$

This approach mainly has two disadvantages, one is that $||\boldsymbol{q}_t^k - \alpha \frac{\partial c(\boldsymbol{q}_{1:T})}{\partial \boldsymbol{q}_t^k}||_2$ may equal to 0, leading to sigularity. The other is that the process could only cover part of the solution space, losing possible solutions. Therefore, we have another approach, which is to project the gradient calculated in every optimization step to tangent space and normalize the result. And then use the projection and the previous value to renew the $\boldsymbol{q}_t$:

$$\boldsymbol{g}_t^k = \frac{\partial c(\boldsymbol{q}_{1:T})}{\partial \boldsymbol{q}_t^k}, \tag{7}$$

$$\boldsymbol{h}_t^k = \boldsymbol{g}_t^k - (\boldsymbol{g}_t^k \cdot \boldsymbol{q}_t^k)\boldsymbol{q}_t^k, \tag{8}$$

$$\boldsymbol{n}_t^k = \boldsymbol{h}_t^k / ||\boldsymbol{h}_t^k||_2, \tag{9}$$

$$\boldsymbol{q}_t^{k+1} = \boldsymbol{q}_t^k \cos\phi_t^k + \boldsymbol{n}_t^k \sin\phi_t^k, \tag{10}$$

where we need to optimize ϕ_t^k by adopting linear gradient search. In practice, we combine two methods to optimize $\boldsymbol{q}_t$, which is to first use projection-based optimization in first few epochs and adopt normalization-based optimization in the last few epochs.

B. Panorama Generation

To generate the panorama, the first step is to get the spherical coordinates in the camera frame for each pixel. The assumption is that the pixels of the picture lies on a unit sphere. Given that the camera has a horizontal and vertical field of view with angles $\frac{\pi}{3}$ and $\frac{\pi}{4}$ respectively, the azimuth angle ϕ and inclination θ of each pixel lies in the range $-\frac{\pi}{6} \leq \phi \leq \frac{\pi}{6}$ and $\frac{3\pi}{8} \leq \theta \leq \frac{5\pi}{8}$. Then for a specific pixel at i-th row and j-th column, the spherical coordinate should be defined as:

$$r = 1 \tag{11}$$

$$\phi = \frac{\pi}{6} - \frac{j}{\#cols}\frac{\pi}{3} \tag{12}$$

$$\theta = \frac{3\pi}{8} + \frac{i}{\#rows}\frac{\pi}{4} \tag{13}$$

Then we convert the spherical coordinate to Cartesian coordinate by using the following formulae:

$$x_{i,j} = \cos(\phi_{i,j})\sin(\theta_{i,j}) \tag{14}$$

$$y_{i,j} = \sin(\phi_{i,j})\sin(\theta_{i,j}) \tag{15}$$

$$z_{i,j} = \cos(\theta_{i,j}) \tag{16}$$

Notice that for every picture, the coordinate of a pixel at position (i, j) is the same. We could use a tensor to represent the coordinates of the picture take at time t in the camera frame as: $P_{t,c} \in \mathbb{R}^{r\times c\times 3}$. Thus, given the rotation matrix R_t and translation vector $\boldsymbol{p}_t$, we could convert the coordinates to world frame as:

$$P_{t,w}^T = R_t \times P_{t,c}^T + \boldsymbol{p}_t \tag{17}$$

As long as the Cartesian coordinates of the pixels have been obtained, we could then obtain the spherical coordinates in the world frame using:

$$\phi_w = \arctan\frac{y_w}{x_w} \tag{18}$$

$$\theta_w = \arccos z_w \tag{19}$$

Then we project the spherical coordinates to cylindrical coordinates by assigning the spherical azimuth to cylinder height and inclination along the circumference. Thus for one pixel with the spherical coordinates as $(1, \theta, \phi)$, its position in the panorama (interpreted as (i_p, j_p)-indexed) could be calculated as:

$$i_p = \frac{\theta_w}{\pi}\#rows_p, \tag{20}$$

$$j_p = \frac{\phi_w + \pi}{2\pi}\#cols_p, \tag{21}$$

in which we interpret the pixels with spherical coordinates $(1, 0, \phi)$ as the center line.

C. Some More Details

The initialization of $\boldsymbol{q}_T$: considering the fact that $\boldsymbol{q}_T$ is initialized using motion model, the loss related to motion model is very small at first, and the gradient could become NAN. Thus, when initializing $\boldsymbol{q}_T$, we added a small noise to it in order to make it bias from original value, so that NAN would not appear at the start and could better train the model.

Training: when training the model, though we added a noise to $\boldsymbol{q}_T$ in initialization, the loss of the motion model is still very small. To balance the two loss a little bit, we multiplied the motion loss by 10.

Panorama: when generating panorama, we changed the shape of the coordinates to $[3 \times (r \times c)]$ for acceleration of the calculation.

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.

B. Panorama

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