An Implementation of Visual-inertial Simultaneous localization and mapping (SLAM) Based on Extended Kalman Filter and Stereo Camera

Wenran, Tian

In this paper, our main object is to build a visual-inertial simultaneous localization and mapping (SLAM) on a moving robot, and based on Extended Kalman Filter (EKF). There is a IMU on the robot to obtain the movement information, and a stereo camera to extract the feature and then obtain the localization information. And after implementing EKF on visual-inertial SLAM, we get a reasonable result.

***Index Terms*—EKF, SLAM, Stereo Camera**

I. Introduction

IN nowadays, self-deriving car is a dramatically hot field. With more and more sophisticated algorithms were implemented, we can make the self-deriving car much more robust than last century. Sensing and detection of the environment is an essential link in self-deriving car control because with an accurate location, we can give the control parameters more specifically and subtly. Our main object is to build a visual-inertial simultaneous localization and mapping (SLAM) on a moving robot, and based on Extended Kalman Filter (EKF). There is a IMU on the robot to obtain the movement information, and a stereo camera to extract the feature and then obtain the localization information.

II. Problem Formulation

A. Kalman Filter

The Kalman Filter is a kind of special case of Bayes filter. Specifically, there are several assumptions in Kalman Filter:

1. The prior pdf $p_{t|t}$ is Gaussian
2. The motion model is linear in the state $\boldsymbol{x}_t$ with Gaussian noise $\boldsymbol{w}_t$
3. The observation model is linear in the state $\boldsymbol{x}_t$ with Gaussian noise $\boldsymbol{v}_t$
4. The motion noise $\boldsymbol{w}_t$ and observation noise $\boldsymbol{v}_t$ are independent of each other, of the state $\boldsymbol{x}_t$, and across time.

However, with those assumptions, it's not easy to give an analytical solution of a Kalman filter model. Because if do that, we need to solve integrals of some equations multiple with Gaussian distribution, which can be quite complicated. On another hand, motion model and observation model are not often linear models. For these reasons, we will choose the Extended Kalman Filter in this paper.

Extended Kalman Filter has the following formats:

$$\boldsymbol{x}_t|\boldsymbol{z}_{0:t}, \boldsymbol{u}_{0:t-1} \sim \mathcal{N}(\boldsymbol{\mu}_{t|t}, \Sigma_{t|t})$$

Motion model:

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{w}_t), \quad \boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{0}, W)$$

$$F_t = \frac{df(\boldsymbol{\mu}_{t|t}, \boldsymbol{u}_t, \boldsymbol{0})}{d\boldsymbol{x}}, \quad Q_t = \frac{df(\boldsymbol{\mu}_{t|t}, \boldsymbol{u}_t, \boldsymbol{0})}{d\boldsymbol{w}}$$

Prediction:

$$\boldsymbol{\mu}_{t+1|t} = f(\boldsymbol{\mu}_{t|t}, \boldsymbol{u}_t, \boldsymbol{0})$$
$$\Sigma_{t+1|t} = F_t \Sigma_{t|t} F_t^T + Q_t W Q_t^T$$

Obs. model:

$$\boldsymbol{z}_t = h(\boldsymbol{x}_t, \boldsymbol{v}_t) \quad \boldsymbol{v}_t \sim \mathcal{N}(0, V)$$

$$H_{t+1} = \frac{dh(\boldsymbol{\mu}_{t+1|t}, \boldsymbol{0})}{d\boldsymbol{x}}, \quad R_{t+1} = \frac{dh(\boldsymbol{\mu}_{t+1|t}, \boldsymbol{0})}{dv}$$

Kalman Gain:

$$K_{t+1|t} = \Sigma_{t+1|t} H_{t+1}^T (H_{t+1} \Sigma_{t+1|t} H_{t+1}^T + R_{t+1} V R_{t+1}^T)^{-1}$$

Update:

$$\boldsymbol{\mu}_{t+1|t+1} = \boldsymbol{\mu}_{t+1|t} + K_{t+1|t}(z_{t+1} - h(\boldsymbol{\mu}_{t+1|t}, \ 0))$$
$$\Sigma_{t+1|t+1} = (I - K_{t+1|t} H_{t+1}) \Sigma_{t+1|t}$$

B. Target Problems

IMU-based Localization via EKF Prediction: Implement the EKF prediction step, based on the SE(3) kinematics and the linear and angular velocity measurements to estimate the pose Tt ∈ SE(3) of the IMU over time t.

Landmark Mapping via EKF Update: assume that the predicted IMU trajectory from above is correct and focus on estimating the landmark positions. In detail, we will implement an EKF with the unknown landmark positions $\boldsymbol{m} \in R^{3\times M}$ as a state and perform EKF update steps after every visual observation $\boldsymbol{z}_t$ in order to keep track of the mean and covariance of $\boldsymbol{m}$. Besides, we are assuming that the landmarks are static so it is not necessary to implement a prediction step. Moreover, since the sensor does not move sufficiently along the

z-axis, the estimation of the z coordinate of the landmarks will not be very good and we will focus only on estimating their xy coordinates.

III. Technical Approach:

A. IMU and Trejectory

The IMU will provide the instantaneous velocity $\boldsymbol{v}_t \in R^3$ and angular velocity $\boldsymbol{\omega}_t \in R^3$ in body frame. Then, we use $\boldsymbol{\zeta}_t$ to denote the generalized velocity $\boldsymbol{\zeta}_t = [\boldsymbol{v}_t \ \boldsymbol{\omega}_t]^T$ .We can calculate the instantaneous transformation to implement the prediction step in Kalman filter. Specifically, we have:

$$T_{t+1} = T_t \exp(\tau \hat{\boldsymbol{\zeta}}_t)$$
$$\hat{\boldsymbol{\zeta}}_t = \begin{bmatrix} \hat{\boldsymbol{\omega}}_t & \boldsymbol{v}_t \\ \boldsymbol{0} & 0 \end{bmatrix}$$

In the EKF prediction step, in time t we have:

$$T_t = \boldsymbol{\mu}_{t|t} \exp\left(\delta \hat{\boldsymbol{\mu}}_{t|t}\right), \ \ \delta \boldsymbol{\mu}_{t|t} \sim \mathcal{N}\left(0, \Sigma_{t|t}\right)$$
$$\boldsymbol{\mu}_{t|t} = \boldsymbol{\mu}_{t|t} \exp(\tau \hat{\boldsymbol{u}}_t)$$
$$\delta \boldsymbol{\mu}_{t+1|t} = \exp\left(-\tau \, \widehat{\boldsymbol{u}}_t\right) \delta \boldsymbol{\mu}_{t|t} + \boldsymbol{w}_t$$
$$\widehat{\boldsymbol{u}}_t = \begin{bmatrix} \hat{\boldsymbol{\omega}}_t & \hat{\boldsymbol{v}}_t \\ \boldsymbol{0} & \hat{\boldsymbol{\omega}}_t \end{bmatrix} \in R^{6\times 6}$$

After implement this prediction step on the given data and tried the dead-reckoning method, we can have the result in Figure 1.

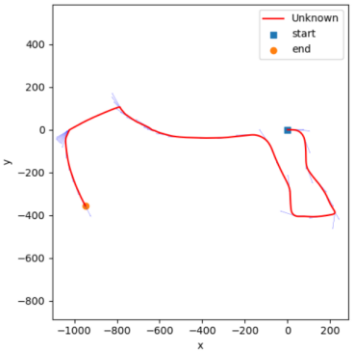

Fig. 1.

B. Visual Mapping

Firstly, we use $\boldsymbol{m}_j$ to denote the j-th feature in world frame, for the relationship between observation $\boldsymbol{z}_{t,i}$ we have:

$$\boldsymbol{z}_{t,i} = h\left(T_t, \boldsymbol{m}_j\right) = K\pi\left(T_{OI} T_t^{-1} \boldsymbol{m}_j\right) + \boldsymbol{v}_{t,i}$$

In the visual mapping step, because we assume the landmarks will not move during sensor collecting data. So, we will ignore the prediction step, and implement our EKF visual mapping algorithm in the following several steps:

a. Initialize the prior parameters $\boldsymbol{\mu}_0 \in R^{3M}$, $\Sigma_0 \in R^{3M\times 3M}$ as all zeros. We have the calibration matrix K, extrinsic $T_{OI} \in SE(3)$.

b. Predicted observations based on $\boldsymbol{\mu}_t$ and known correspondence Δ_{t+1}:

$$\tilde{\boldsymbol{z}}_{t+1,i} = K\pi\left(T_{OI} T_t^{-1} \boldsymbol{\mu}_{t,j}\right) \in R^4, \ \ for\ i = 1,2,\dots,N_{t+1}$$

c. Jacobian of $\tilde{\boldsymbol{z}}_{t+1,i}$ with respect to $\boldsymbol{m}_j$ evaluated at $\boldsymbol{\mu}_{t,j}$:

$$H_{t+1,i,j} = \begin{cases} K \dfrac{d\pi}{d\boldsymbol{q}}\left(T_{OI} T_t^{-1} \boldsymbol{\mu}_{t,j}\right) T_{OI} T_t^{-1} P^T, & if \Delta_t(j) = i \\ \boldsymbol{0}, \in R^{4\times 3}, & otherwise \end{cases}$$

d. EKF update:

$$K_{t+1} = \Sigma_t H_{t+1}^T (H_{t+1} \Sigma_t H_{t+1}^T + I \otimes V)^{-1}$$
$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + K_{t+1|t}(\boldsymbol{z}_{t+1} - \tilde{\boldsymbol{z}}_{t+1})$$
$$\Sigma_{t+1} = (I - K_{t+1} H_{t+1}) \Sigma_t$$

It needs to be emphasized that during the computation, we may not able to update the entire Σ_t matrix each time. If we have 10000 landmarks, the size of Σ_t will be $(3 \times 10000) \times (3 \times 10000) = 9 \times 10^8$. If we use float32 to store Σ_t, the size of cache to store Σ_t will be about 3.35 GiB, which will be quite computational costly if we take it as operand during the iteration. On the other hand, in EKF update step, we even need to compute an inverse of a matrix, which we require even more computational resource. So, to simplify the computation, in our code we will just find those landmarks will be affected in an iteration, and crop the corresponding variance in Σ_t to compute. In addition, the H_t and K_t will also be implemented within a certain range in an iteration.

For the initial state, there is no value of $\mu_0 \in R^{3M}$ because we did not do any update step. Our strategy is, once we get a point from observation that is not initialized in μ, we will transform the point in stereo camera coordinate to world frame, then use the value in world frame to initialize it. So this means we will initial the location of each landmark separately.

And this time, we will add some noise to the IMU data and solve the trajectory of the robot. After we implement the aforementioned steps, we can get the point cloud image in Figure 2:

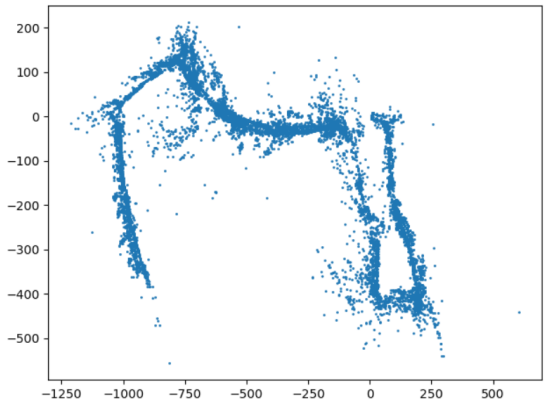

Fig. 2.

C. Visual-Initial SLAM

Here we will combine the methods in two previous section and get a complete visual-initial SLAM. In each time of iteration, the execution steps will be:

1. Predict the localization
2. Update/Initialize the mapping
3. Update the localization

The update processes for landmarks are the same as we described in visual mapping section, so following EKF processes are all about the localization.

1. Prediction step:

$$\mu \in SE(3), \ \ \Sigma \in R^{6\times 6}$$
$$\mu_{t+1|t} = \mu_{t|t} \exp(\tau \widehat{\boldsymbol{u}}_t)$$
$$\delta \mu_{t+1|t} = \exp\left(-\tau \, \widehat{\boldsymbol{u}}_t\right) \delta \mu_{t|t} + \boldsymbol{w}_t, \ \ \boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{0}, W)$$
$$\Sigma_{t+1|t} = E\left(\delta \mu_{t+1|t} \delta \mu_{t+1|t}^T\right)$$
$$= \exp\left(-\tau \, \widehat{\boldsymbol{u}}_t\right) \Sigma_{t|t} \exp\left(-\tau \, \widehat{\boldsymbol{u}}_t\right)^T + W$$

2. Update step:

$$\tilde{\boldsymbol{z}}_{t+1,i} = K\pi\left(T_{OI}\mu_{t+1|t}^{-1}\boldsymbol{m}_j\right) \in R^4, \quad for\ i = 1,2,\dots,N_{t+1}$$

$$H_{t+1,i} = -K\frac{d\pi}{d\boldsymbol{q}}\left(T_{OI}\mu_{t+1|t}^{-1}\boldsymbol{m}_j\right)T_{OI}(\mu_{t+1|t}^{-1}\boldsymbol{m}_j)^{\odot} \in R^{4\times 6}$$

For the "$\odot$" operation:

$$\boldsymbol{p} \in R^3, \quad \begin{bmatrix}\boldsymbol{p}\\ 1\end{bmatrix}^{\odot} = \begin{bmatrix} I & -\hat{\boldsymbol{p}} \\ \boldsymbol{0} & 0\end{bmatrix} \in R^{4\times 6}$$

Then we stack all the $\tilde{\boldsymbol{z}}_{t+1,i}$ to $\boldsymbol{z}_{t+1}$, all the $H_{t+1,i}$ to H. We can have:

$$\boldsymbol{z}_{t+1} \in R^{4N_t}, \quad H_{t+1} \in R^{4N_t\times 6}$$

$$K_{t+1} = \Sigma_{t+1|t}H_{t+1}^T(H_{t+1}\Sigma_t H_{t+1}^T + I \otimes V)^{-1} \in R^{6\times 4N_t}$$

$$\boldsymbol{\zeta}_t = K_{t+1}(\boldsymbol{z}_{t+1} - \tilde{\boldsymbol{z}}_{t+1}) \in R^6$$

$$\boldsymbol{\mu}_{t+1|t+1} = \boldsymbol{\mu}_{t+1|t}exp\ (\hat{\boldsymbol{\zeta}}_t)$$

$$\Sigma_{t+1|t+1} = (I - K_{t+1}H_{t+1})\Sigma_{t+1|t}$$

And again, in our code, we will still use the same tactics as we did in Visual Mapping step. In each iteration, we will compute the part of Σ_t that will be influenced, instead of the entire Σ_t. In addition, we will treat the sigma of position Σ_t^P and sigma of landmarks Σ_t^M separately instead of using a combined Sigma matrix $\Sigma_t \in R^{(3M+6)\times(3M+6)}$.

IV. RESULTS

After we execute the Visual-Initial SLAM, we can have the point cloud of SLAM and trajectory that in Figure 3:

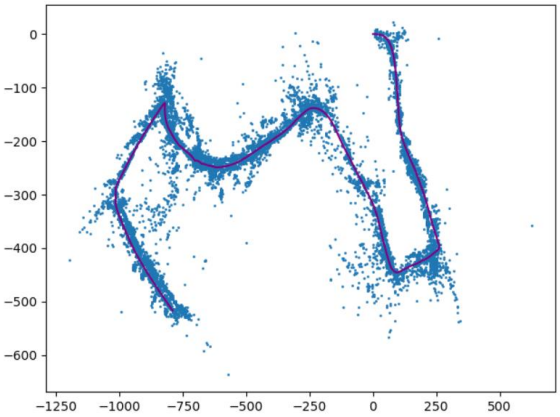

Fig. 3. The purple line stands for trajectory, the blue points stands for features in stereo camera.

Although we get a reasonable result, we find that the program takes about 50mins to run. After analysis, we hold the opinion that is because it contains two steps of finding the inverse matrix in each iteration:

$$K_{t+1} = \Sigma_{t+1|t}H_{t+1}^T(H_{t+1}\Sigma_t H_{t+1}^T + I \otimes V)^{-1}$$

As we know, it is quite computational costly to inverse a large matrix. In the code, even if we crop the matrix to compute in each iteration, we still need to inverse a matrix with size about $(50\times 4,\ 50\times 4)$. If the computational complexity of np.linalg.inv is $O(n^3)$, it will require $O(200^3)$ times of iteration to run. We think it is worth to discussion furtherly about how to replace the inverse item in the computation of K_{t+1} to speed up the algorithm in the future.