Abstract—This is the report for Hao’s Project 3 Visual Inertial SLAM (VI-SLAM). In this report, Hao reviewed the key component of VI-SLAM and the techniques Hao has tried to improve the performance. In this project Visual-inertial SLAM utilize IMU data and camera data to joint estimate the inverse pose of robot and the 3D positions of landmarks.

Index Terms—SLAM, EKF, Visual-Inertial

I. INTRODUCTION

SLAM aims to build a map from an unknown environment while keeping the trajectory of the agents. Leonard, John etc. [2] first introduced the concept of SLAM. Since then, SLAM has gradually become a key problem in the mobile robotics and has developed several approaches including Graph-SLAM [4], EKF-SLAM [1], Fast-SLAM [3] etc. The difficulty of the SLAM problem comes from the chicken-or-egg situation, i.e. the environment and the location both are unknown while estimating one needs the information of the other unknown one.

SLAM is very useful in many domain like outdoor/indoor robot navigation, building local map/global map as well as terrian mapping in the space. And with the development of deep learning and semantic segmentation, there is also new research on semantic slam.

EKF-based SLAM algorithm uses moment matching idea by leveraging linear approximation techniques to make the calculation feasible. Due to the Gaussian assumption, EKF can estimate the positions in continuous 3D space while histogram filter based SLAM can only approximate the position in finite space. Another approach similar to EKF is Unscented Kalman Filter, it utilize the numerical approximation method to estimate the joint probability.

In this project, Hao implemented EKF-SLAM for the task of Visual-Inertial SLAM by utilizing the stereo camera and IMU data. To capture the correlations between landmarks and inverse poses, Hao utilized joint update. Hao tried several techniques like feature selection, z-axis fixation etc.

II. PROBLEM FORMULATION

Visual Inertial SLAM: Given the IMU data \( u_{1:T} \), sampling time interval \( \delta t \) and visual feature observation \( z_{1:T} \) which is detected by stereo cameras, estimate the inverse IMU pose \( U_{1:T} \) and landmark position \( m \), s.t. \( U_t, m = \arg\max_{U_t, m} p(U_{1:T}, m | u_{1:T}, z_{1:T}) \).

Here the subcript \( t \) indicate the data for current time. \( u_t = \begin{bmatrix} v_t \\ \omega_t \end{bmatrix} \) is the observed IMU data including linear velocity \( v_t \in \mathbb{R}^3 \) and rotational velocity \( \omega_t \in \mathbb{R}^3 \). \( \delta t \) is the difference between time stamps \( t \) and time stamps \( t - 1 \). Landmark positions \( m \) is the 3D position of all the landmarks.

In this project, Hao assume the number of total landmarks are knwon in advance, thus \( m \in \mathbb{R}^{3 \times M} \), where \( M \) is the total number of features or landmarks we care about. \( U_t = T_{IMU,t}^{-1} \in SE(3) \) is the inverse IMU pose for the time stamp \( t \). Also in this project, Hao assume the data association \( \pi_t \) is also known. Visual feature observations are the pixel coordinates of the feature points in the left and right camera frame, thus \( z_t \in \mathbb{R}^{4 \times N_t} \) where \( N_t \) is the number of features observed at time frame \( t \). Besides, Hao also assumed the map is static thus no motion model for this project’s mapping.

This problem can be subdevided to Localization and Mapping problem.

Localization: Assume the world frame landmark coordinates \( m \) are known, given the IMU data \( u_{1:t} \) and the feature observations \( z_{1:t} \), Hao want to estimate the inverse IMU pose \( U_t \) s.t. \( U_t = \arg\max_{U_t} p(U_t | u_{1:t}, z_{1:t}, m) \).

Mapping: Assume the inverse pose \( U_{1:t} \) is known, given the visual feature observation \( z_{1:t} \), Hao estimate \( m \) s.t. 
\[
\begin{align*}
    m &= \arg\max_m p(m | z_{1:t}, U_{1:t}) \\
    U_{1:t} &= \exp (\delta \tau_t (u_t + w_t)^\top) U_t
\end{align*}
\]

III. TECHNICAL APPROACH

A. Localization EKF Prediction Step

For the prediction step, Hao assume the prior \( U_t \) satisfies
\[
U_t | z_{1:t}, u_{0:t-1} \sim \mathcal{N} (\mu_{t|t}, \Sigma_{t|t})
\]
And then Hao have the motion model as equation.2 ,where \( \delta t \) is the time discretization.
\[
U_{t+1} = \exp (\delta \tau_t (u_t + w_t)^\top) U_t
\]

Thus for the mean \( \mu_{i=1:t} \in SE(3) \) and covariance \( \Sigma_{i+1:t} \in \mathbb{R}^{6 \times 6} \) their update rules are shown as equation.3
\[
\begin{align*}
    \mu_{i+1|t} &= \exp (\delta \tau_t (u_t)^\top) \mu_{i|t} \\
    \Sigma_{i+1|t} &= \exp (\delta \tau_t (u_t)^\top) \Sigma_{i|t} \exp (\delta \tau_t (u_t)^\top)^\top + W
\end{align*}
\]

where \( W \) is the covariance of noise for motion model. After experimenting with several values for the noise \( W \) ranging from 10^{-6} – 10^{6}, Hao choose the noise as equation.4
\[
W = \begin{bmatrix}
    10^{-2} & 0 & 0 & 0 & 0 & 0 \\
    0 & 10^{-2} & 0 & 0 & 0 & 0 \\
    0 & 0 & 10^{-2} & 0 & 0 & 0 \\
    0 & 0 & 0 & 10^{-1} & 0 & 0 \\
    0 & 0 & 0 & 0 & 10^{-1} & 0 \\
    0 & 0 & 0 & 0 & 0 & 10^{-1}
\end{bmatrix}
\]
Since our IMU is better at estimating the linear velocity, thus I gave the corresponding correlation slightly smaller value while giving the one for rotational velocity higher value, meaning higher unconfidence. Also the hat ` and curly hat  operations are defined as equation.5

\[
\hat{u}_t := \begin{bmatrix} \hat{\omega}_t \\ v_t \\ 0 \end{bmatrix} \in \mathbb{R}^{4 \times 4} \quad \land \quad \mu_t := \begin{bmatrix} \hat{\omega}_t \\ \hat{v}_t \\ 0 \end{bmatrix} \in \mathbb{R}^{6 \times 6}
\] (5)

B. Mapping EKF update step

I first initialize the position of the new landmarks which means it is their first time being observed. And then update the EKF mean and covariance for all the previously observed landmarks. To be more specific, for initialization, I transform the points from pixel coordinate to optical frame and then transform to IMU frame via the transformations. After that I will transform the points from IMU frame to world frame via the pose of the robot e.i. \( U_t^{-1} \).

\[
P_o = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \frac{Z(u_k-c_w)}{f_x} \\ \frac{Z(u_k-c_w)}{f_y} \\ \frac{u_k-a_l}{a_l-a_R} \end{bmatrix}
\] (6)
\[
P_w = U_t^{-1} o T_i^{-1} P_o
\] (7)

For the EKF update step, assume the prior \( \mathbb{m} \) satisfies:

\[
\mathbb{m} z_{1:t} \sim N(\mu_t, \Sigma_t)
\] (8)

Also we have the observation model as equation.9

\[
z_{t,i} = h(U_t, \mathbb{m}_j) + v_{t,i} := M \pi (o T_i U_t \mathbb{m}_j) + v_{t,i}
\] (9)

Thus the update rule for the mean and covariance are shown in equation.10

\[
K_t = \Sigma_t H_t^T (H_t \Sigma_t H_t^T + I \otimes V)^{-1}
\]
\[
\mu_{t+1} = \mu_t + K_t (z_t - M \pi (o T_i U_t \mu_j))
\]
\[
\Sigma_{t+1} = (I - K_t H_t) \Sigma_t
\] (10)

Here \( \pi(q) \) is the projection function as \( \pi(q) = \frac{1}{q_3} q \). And the \( \mu_t \) is the homogeneous coordinates of \( \mu_t \). Besides the \( \frac{d \pi}{dq} \) is

\[
\frac{d \pi}{dq}(q) = \frac{1}{q_3} \begin{bmatrix} 1 & 0 & -q_2 & 0 \\ 0 & 1 & -q_2 & 0 \\ 0 & 0 & q_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 4}
\]

And \( H_t \in \mathbb{R}^{4N_t \times 3M} \) is shown in equation.11

\[
H_{t,i,j} = \begin{cases} M \frac{d \pi}{dq}(o T_i U_t \mu_{j}) o T_i U_t P^T & \text{If correspondence} \\ 0 & \text{otherwise} \end{cases}
\] (11)

Here I assume \( V \) is a diagonal matrix with the value of 1, thus \( I \otimes V \) will also be a diagonal matrix in \( \mathbb{R}^{4N_t \times 4N_t} \) with element of 1.

C. EKF joint update for Mapping and Localization

For part c, I calculate a joint \( H_t \) for both mapping and localization and then I calculate the corresponding \( K_t \). Also we have the observation model as equation.9. And use the first 3\( M \) row to update mapping’s mean/variance and the last 6 rows to update localization’s mean/variance. Also I use \( K_t (z_t - M \pi (o T_i U_t \mu_j)) \) to update the joint variance. I will detail the process in the following paragraphs.

Define the means for mapping and localization are \( \mu_{map,t} \) and \( \mu_{loc,t} \) respectively. Define the joint covariance as \( \Sigma_t \in \mathbb{R}^{(3(M+6)) \times (3(M+6))} \). Define the joint \( H_t \in \mathbb{R}^{4N_t \times (3M+6)} \) where \( H_{map,t} \in R^{4N_t \times 3M} \) is calculated by using equation.11 like before. And for \( H_{loc,t,i} \in \mathbb{R}^{4N_t \times 6} \), I use equation.12.

\[
H_{loc,t,i} = M \frac{d \pi}{dq}(o T_i \mu_{loc,t,i}) o T_i (\mu_{loc,t,i})^T
\] (12)

where \( \circ \) is defined as equation.13

\[
\frac{\partial}{\partial q} = \begin{bmatrix} I & -\frac{\partial}{\partial q} \end{bmatrix} \in \mathbb{R}^{4 \times 6}
\] (13)

Then I use the joint \( H_t \) to calculate \( K_t \) by using the equation.14

\[
K_t = \Sigma_t H_t^T (H_t \Sigma_t H_t^T + I \otimes V)^{-1}
\]

And I would calculate the \( K_t \) times innovation as equation.15

\[
\delta Kz = K_t (z_t - M \pi (o T_i U_t \mu_{map,t}))
\] (15)

And then I will use the equation.16 to update the mean of mapping and use equation.17 to update the joint covariance. Also I use equation.18 to update the mean for the localization.

\[
\mu_{map,t+1} = \mu_{map,t} + \delta Kz
\]
\[
\Sigma_{t+1} = (I - K_t H_t) \Sigma_t
\]
\[
\mu_{loc,t+1} = \exp \left( (\delta Kz [3M : 3M + 6, :]) \right) \mu_{loc,t}
\] (16)

Also for the prediction step of the localization, I will use the same process but this time only update part of the joint covariance. In other words, I will update \( \Sigma_t [3M : 3M + 6, 3M : 3M + 6] \in \mathbb{R}^{6 \times 6} \). This process can be shown in the following equation.

\[
\Sigma_{loc,t+1}[3M : 3M + 6, 3M : 3M + 6] =
\exp \left( -\delta \tau u_t^\lambda \right) \Sigma_{loc,t}[3M : 3M + 6, 3M : 3M + 6] \exp \left( -\delta \tau u_t^\lambda \right)^T + W
\]

D. Techniques

1) Fix z-axis: As stated in the problem description, since sensor’s movement along the axis is very small, thus we can assume the axis are the same over time. To model this, I tried two approaches

a) Set z-axis of \( \mu_{map,t} \) to 0 after the update step of the mapping. But during experiment, I find that this approach would sometimes lead to singular matrix error. I think the
reason for this may due to the fact that we are forcing 
\( z = 0 \), thus the \( y \)-axis in the optical frame is also zero. 
According to the stereo camera model as equation 19, 
we know that \( v_L = v_R = c_v \). Thus it would introduce 
the error when calculating the innovation term since the 
observations don’t have fixed \( v_L \) or \( v_R \).

\[
\begin{bmatrix}
u_L \\
v_L \\
v_R \\
v_R \\
\end{bmatrix} = \begin{bmatrix}
fs u & 0 & ca & 0 \\
0 & fs v & cv & 0 \\
0 & fs u & 0 & -fs a b \\
0 & fs v & cv & 0 \\
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
1 \\
\end{bmatrix} \tag{19}
\]

To solve this problem, I force the \( v_L \) and \( v_R \) of observations to be \( c_v \). And the experiment results showed promising performance as figure 1.

\[
\begin{bmatrix}
\begin{bmatrix}
ul \\
v_l \\
ul \\
v_l \\
\end{bmatrix} \\
\begin{bmatrix}
ul \\
v_l \\
ul \\
v_l \\
\end{bmatrix} \\
\begin{bmatrix}
ul \\
v_l \\
ul \\
v_l \\
\end{bmatrix} \\
\begin{bmatrix}
ul \\
v_l \\
ul \\
v_l \\
\end{bmatrix} \\
\end{bmatrix} \\
\begin{bmatrix}
s_f u & 0 & ca & 0 \\
0 & s_f v & cv & 0 \\
0 & s_f u & 0 & -s_f a b \\
0 & s_f v & cv & 0 \\
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
1 \\
\end{bmatrix} = \begin{bmatrix}
x \\
y \\
z \\
1 \\
\end{bmatrix}
\]

Fig. 1: Results for fix z-axis to 0 on dataset 27.

*) Set the z-axis to be the same as the first observed z-axis value after update step. To be more specific, after calculating the update step for mapping, I choose to set the z-axis value to be the one before the update step so that during the EKF process the estimated z-axis remains fixed. This approach also demonstrate promising results as shown in figure 2.

\[
\begin{bmatrix}
\begin{bmatrix}
ul \\
v_l \\
ul \\
v_l \\
\end{bmatrix} \\
\begin{bmatrix}
ul \\
v_l \\
ul \\
v_l \\
\end{bmatrix} \\
\begin{bmatrix}
ul \\
v_l \\
ul \\
v_l \\
\end{bmatrix} \\
\begin{bmatrix}
ul \\
v_l \\
ul \\
v_l \\
\end{bmatrix} \\
\end{bmatrix} \\
\begin{bmatrix}
s_f u & 0 & ca & 0 \\
0 & s_f v & cv & 0 \\
0 & s_f u & 0 & -s_f a b \\
0 & s_f v & cv & 0 \\
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
1 \\
\end{bmatrix} = \begin{bmatrix}
x \\
y \\
z \\
1 \\
\end{bmatrix}
\]

Fig. 2: Results for fix z-axis to the same observed value on dataset 27.

IV. EXPERIMENT RESULTS

After some experiments, I choose to use 1000 features and the experiments results demonstrate that using all the feature points may not increase the performance while needing long time.

A. IMU-based Localization via EKF Prediction

The results for part (a) are shown in figure 3. In part (a), I only do the EKF prediction step for the localization. As we can see as the robot observes the inertial input, the robot would update its mean and covariance. And the trajectory is inverse of the estimated mean. Since there is no update step, thus the robot could not correct its pose. This corresponds to the accumulated error in the image like (c) in figure 3.

\[
\begin{bmatrix}
\begin{bmatrix}
ul \\
v_l \\
ul \\
v_l \\
\end{bmatrix} \\
\begin{bmatrix}
ul \\
v_l \\
ul \\
v_l \\
\end{bmatrix} \\
\begin{bmatrix}
ul \\
v_l \\
ul \\
v_l \\
\end{bmatrix} \\
\begin{bmatrix}
ul \\
v_l \\
ul \\
v_l \\
\end{bmatrix} \\
\end{bmatrix} \\
\begin{bmatrix}
s_f u & 0 & ca & 0 \\
0 & s_f v & cv & 0 \\
0 & s_f u & 0 & -s_f a b \\
0 & s_f v & cv & 0 \\
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
1 \\
\end{bmatrix} = \begin{bmatrix}
x \\
y \\
z \\
1 \\
\end{bmatrix}
\]

Fig. 3: Part (a) results for dataset 22, 27, 34. Localization EKF prediction step only.

B. Landmark Mapping via EKF Update

The results for part (b) are shown in figure 4.

\[
\begin{bmatrix}
\begin{bmatrix}
ul \\
v_l \\
ul \\
v_l \\
\end{bmatrix} \\
\begin{bmatrix}
ul \\
v_l \\
ul \\
v_l \\
\end{bmatrix} \\
\begin{bmatrix}
ul \\
v_l \\
ul \\
v_l \\
\end{bmatrix} \\
\begin{bmatrix}
ul \\
v_l \\
ul \\
v_l \\
\end{bmatrix} \\
\end{bmatrix} \\
\begin{bmatrix}
s_f u & 0 & ca & 0 \\
0 & s_f v & cv & 0 \\
0 & s_f u & 0 & -s_f a b \\
0 & s_f v & cv & 0 \\
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
1 \\
\end{bmatrix} = \begin{bmatrix}
x \\
y \\
z \\
1 \\
\end{bmatrix}
\]

Fig. 4: Part (b) results for dataset 22, 27, 34. The results are for landmark mapping via EKF update step.
In this part, I implemented both localization prediction step and mapping update step but no joint update. As shown in the figure, the landmark prediction could adjust its position according to the information of the observation model. But since there is no update step for the localization, thus the trajectory is the same as part (a) with accumulated error.

C. Visual Inertial SLAM

The results for part (c) are shown in the figure. In this part, I implemented the full visual inertial slam including localization prediction step and joint update step. Also I have attached the experiment results as a gif figure in the zip in the code part. For readers’ convenience, I also plot the process for SLAM is shown in the feature. (They are attached at the end of the report.)

![Dataset 22](image1.png)
(a) dataset 22

![Dataset 27](image2.png)
(b) dataset 27

![Dataset 34](image3.png)
(c) dataset 34

Fig. 5: Part (c) results for dataset 22, 27, 34. The results are for Visual Inertial SLAM including the joint EKF update step and localization prediction step.

As we can see the results are pretty good! For dataset 27, the start point and end point is very close to each other while in part (a) and part (b), the start point and end point are far away from each other. For dataset 22, the line started from forth turning point is parallel to the start line while in part (a) and part (b) the parallel relationship is not satisfied. For dataset 34, the right turn is almost 90 degree while in the part (a) and part(b) the degree is near 180, which is very different from the video. Also the landmark positions become more accurate! Compared with part (a) and (b), the results have far less outliers which are far away from the trajectory.

The improvement is largely due to the joint update step. Joint update step could capture the correlations between landmarks and inverse pose, making the estimation more accurate. And during prediction step, robot estimate its inverse pose via motion model while during update step, robot adjust its estimated inverse pose according to the motion model. The landmark estimation also become more accurate during update step.

D. Comparison between two different methods for fixing z-axis

As I mentioned in the technical approach section, I tried two methods to fix z-axis. The first approach is by setting z-axis of mean to 0 and forcing the observed features’ \(v_L\) and \(v_R\) to \(c_v\). The second approach is forcing the z-axis of mean to the first observed value so that z-axis remains the same over time.

![Comparison between 2 different methods of fixing z-axis](image4.png)
(a) Set z to 0
(b) Set z to first observed value

Fig. 6: Comparison between 2 different methods of fixing z-axis.

As shown in the figure 6, both approach could reach satisfying results. But there is a slightly rotation relationship between two methods. This is because if we use approach 2, even setting the z-axis to the previous value, we would still estimate a small innovation error along this axis, so the model would also try to fit the z-axis a little bit.

E. Data Association and data processing

Since the data association is stored, thus I use a numpy boolen array to store the index of the landmark. If the element is true then it is observed before. Otherwise it is new. And I use a boolen mask to store the observed features at current time stamp. If the feature is not [-1,-1,-1], then it is observed at current time stamp. And its correpsponding value would be True.

F. Tensor Speed Up

I use \texttt{np.tensordot} to perform the tensor multiplication instead of a for loop. So that the joint update could speed up! And when use 1000 features, it takes me less than 2 minutes to run the whole dataset.

V. Conclusion

In this project, I implemented Visual Inertial SLAM and achive satisfying results for all three dataset. For better performance, I utilize joint update techniques. And I experiment with 2 fix-z axis techniques. And analysis the methodology behind them.
Fig. 7: Process for building SLAM on dataset 34

Fig. 8: Process for building SLAM on dataset 22.

REFERENCES


Fig. 9: Process for building SLAM on dataset 27.