Visual Inertial SLAM

Benjamin Chang
Department of Electrical and Computer Engineering
University of California, San Diego
bmc011@ucsd.edu

***Abstract*—In recent years, large efforts have been made toward enabling autonomous vehicles to better understand their surroundings. This usually involves a variety of sensors such as IMU, lidar, stereo cameras, and radar. One such common method which uses a combination of stereo camera and IMU is Visual-Inertial SLAM (Simultaneous Localization and Mapping). This project aims to use data captured from a variety of sensors attached to a vehicle to perform SLAM, thus localizing the vehicle and mapping its surrounding environment.**

I. Introduction

The ability for autonomous vehicles to locate themselves in their environment is extremely important for safe navigation. Mapping a vehicle's environment is also highly valuable as that enables it to better understand of its surroundings and thus to traverse more safely. Environment mapping also has applications not related to autonomous vehicles as seen in the Google Earth project. Without an efficient way to do environment mapping, the task of mapping large areas would otherwise be tedious and require cars with human drivers.

In this project, several steps are implemented to localize a vehicle and to map its surroundings using two types of captured sensor data: stereo camera and IMU. These enable the vehicle to understand its surroundings and inertial movements (linear and angular velocity). For this project, we use a specific dataset which includes landmark features calculated from the stereo camera images using an external algorithm.

This project is split into three steps. The first is "IMU-based Localization via EKF Prediction" which predicts where the vehicle will be at a future time. The second is "Landmark Mapping via EKF Update" which estimates and updates where landmarks are located on a map in the world frame. The third is "Visual-Inertial SLAM" which combines the first two steps and adds an IMU update step to obtain a complete visual-inertial SLAM algorithm which plots the vehicle trajectory and estimated landmark positions on a map.

II. Problem Formulation

For Visual-Inertial Localization and Mapping, we have 3 inputs: IMU linear acceleration $\alpha_t \in \mathcal{R}^3$, IMU angular acceleration $\omega_t \in \mathcal{R}^3$, and features detected from stereo camera images $z_{t,i} \in \mathcal{R}^3$ which include pixel feature coordinates in both left and right stereo images for $i = 1, ..., N_t$.
Some parameters assumed to be known are the IMU to camera optical frame transformation ${}_oT_I \in \mathrm{SE}(3)$ and the stereo camera calibration matrix M which has the following equation.

$$M := \begin{bmatrix} fs_u & 0 & c_u & 0 \\ 0 & fs_v & c_v & 0 \\ fs_u & 0 & c_u & -fs_u b \\ 0 & fs_v & c_v & 0 \end{bmatrix}$$

f = focal length $[m]$
s_u, s_v = pixel scaling $[pixels/m]$
c_u, c_v = principal point $[pixels]$
b = stereo baseline $[m]$

With the above given inputs and assumed parameters, we want to solve for the world frame IMU pose ${}_WT_I \in \mathrm{SE}(3)$ over time and world-frame coordinates $m_j \in \mathcal{R}^3$ of the $j = 1, ..., M$ point landmarks that generated the visual features $z_{t,i} \in \mathcal{R}^4$.
Considering the landmark mapping problem first, We assume for our project that IMU pose $T_t = {}_WT_{I,t} \in \mathrm{SE}(3)$ is always known. Our objective then is to estimate the coordinates $m = [m_1^T ... m_M^T] \in \mathcal{R}^{3M}$ of the landmarks that generated the given input observations $z_t = [z_{t,1}^T ... z_{t,N_t}^T] \in \mathcal{R}^{4N_t}$ for $t = 0, ..., T$. In doing this, we assume the data association $\Delta_t : \{1, ..., M\} -> \{1, ..., N_t\}$ such that landmark j corresponds to observation $z_{t,i} \in \mathcal{R}^4$ with $i = \Delta_t(j)$ at time t is known or provided by an external algorithm. The landmarks m_i are assumed to be static so that it is not necessary to consider a motion model or prediction step.

The observation model is given to be as follows.

$$\mathbf{z}_{t,i} = h(T_t, \mathbf{m}_j) + \mathbf{v}_{t,i} := M\pi\left({}_oT_I T_t^{-1} \underline{\mathbf{m}}_j\right) + \mathbf{v}_{t,i} \qquad \mathbf{v}_{t,i} \sim \mathcal{N}(0, V)$$

where $v_{t,i}$ is random Gaussian noise with standard deviation V, and M is the the calibration matrix (intrinsic parameters) of the stereo camera.

We now consider the visual-inertial odometry (localization) problem which needs to predict and update the location of the the vehicle IMU. For this, we assume linear velocity and linear acceleration measurements are available along with known world frame landmark coordinates $m \in \mathcal{R}^{3M}$ and the data association mentioned earlier for the landmark mapping only problem. With these inputs and assumptions, we seek to estimate the pose $T_t = {}_WT_{I,t} \in \mathrm{SE}(3)$ of the IMU over time using the IMU motion model below.

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t \exp\left(\tau \hat{\mathbf{u}}_t\right)$$
$$\delta\boldsymbol{\mu}_{t+1} = \exp\left(-\tau \hat{\mathbf{u}}_t\right) \delta\boldsymbol{\mu}_t + \mathbf{w}_t$$

This motion model is obtained by perturbing the pose kinematics with time discretization τ.

III. Technical Approach

We first import all of our data from the 10.npz dataset which includes timestamps, features, linear velocity, angular velocity, istrinsic calibration matrix, baseline, and IMU to camera transformation, to use in our calculations. This dataset includes features which are already matched across left-right camera frames through use of an external algorithm. The feature matching this algorithm does is shown below in Fig. 1.

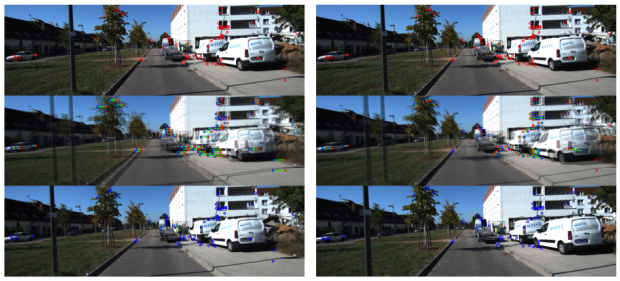

Fig. 1. Matched visual features between left camera frame (left) and right camera frame (right)

We also choose to use only every fifth feature to enable our code to run faster.

Using our data, we calculate the IMU pose prediction step at a future timestamp $t+1$ given the IMU pose at t with the following equations.

$$\boldsymbol{\mu}_{t+1|t} = \boldsymbol{\mu}_{t|t} \exp\left(\tau \hat{\mathbf{u}}_t\right)$$
$$\Sigma_{t+1|t} = \mathbb{E}[\delta\boldsymbol{\mu}_{t+1|t}\delta\boldsymbol{\mu}_{t+1|t}^{\top}] = \exp\left(-\tau\hat{\mathbf{u}}_t\right)\Sigma_{t|t}\exp\left(-\tau\hat{\mathbf{u}}_t\right)^{\top} + W$$

The above equations predict the pose mean and covariance of the IMU. τ is the time difference between IMU data timestamps and control input u_t consists of linear velocity $\omega_t \in \mathcal{R}^{3x1}$ and angular velocity $v_t \in \mathcal{R}^{3x1}$ collected from the vehicle IMU. The control input is utilized with the below equations.

$$\mathbf{u}_t := \begin{bmatrix}\mathbf{v}_t \\ \boldsymbol{\omega}_t\end{bmatrix} \in \mathbb{R}^6 \quad \hat{\mathbf{u}}_t := \begin{bmatrix}\hat{\boldsymbol{\omega}}_t & \mathbf{v}_t \\ \mathbf{0}^{\top} & 0\end{bmatrix} \in \mathbb{R}^{4\times4} \quad \hat{\hat{\mathbf{u}}}_t := \begin{bmatrix}\hat{\boldsymbol{\omega}}_t & \hat{\mathbf{v}}_t \\ 0 & \hat{\boldsymbol{\omega}}_t\end{bmatrix} \in \mathbb{R}^{6\times6}$$

The *hat* operator denotes a skew-symmetric matrix which is found for a 3x1 vector as follows [0, -w[2], v[1]; v[2], 0, -v[0]; -v[1], v[0], 0]. We test the above described IMU prediction step and update the vehicle position in world frame for dead-reckoning to see if our trajectory seems reasonable before also doing landmark updates.

To update landmark positions, we use the EKF update step to calculate new landmark locations μ_{t+1} in the world frame given new observations $z_{t+1} \in \mathcal{R}^{4N_{t+1}}$ and previous landmark position estimates μ_t . This update step is done using the following equations.

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + K_{t+1}\left(\mathbf{z}_{t+1} - \tilde{\mathbf{z}}_{t+1}\right)$$
$$\Sigma_{t+1} = (I - K_{t+1}H_{t+1})\Sigma_t$$

In the above equations, the new observations $z_{t+1} \in \mathcal{R}^{4N_{t+1}}$ and $K_{t+1|t}$ are modelled with the following observation model.

$$\tilde{\mathbf{z}}_{t+1,i} := M\pi\left({}_oT_I\boldsymbol{\mu}_{t+1|t}^{-1}\underline{\mathbf{m}}_j\right) \qquad \text{for } i = 1,\ldots,N_{t+1}$$

$$K_{t+1} = \Sigma_t H_{t+1}^{\top}\left(H_{t+1}\Sigma_t H_{t+1}^{\top} + I \otimes V\right)^{-1}$$

where V is a tunable parameter. $H_{t+1,i} \in \mathcal{R}^{4x6}$ is the Jacobian of predicted observations $z_{t+1} \in \mathcal{R}^{4N_{t+1}}$ with respect to the IMU pose T_{t+1} evalutaed at the predicted IMU position $\mu_{t+1|t}$. The equation for the Jacobian is shown below.

$$H_{t+1,i} = -M\frac{d\pi}{d\mathbf{q}}\left({}_oT_I\boldsymbol{\mu}_{t+1|t}^{-1}\underline{\mathbf{m}}_j\right){}_oT_I\left(\boldsymbol{\mu}_{t+1|t}^{-1}\underline{\mathbf{m}}_j\right)^{\odot} \in \mathbb{R}^{4\times6}$$

where M is the camera's calibration matrix, m_j is the homogenous coordinates of landmark j, ${}_oT_I \in \mathrm{SE}(3)$ is the IMU to optical (camera) frame transformation, and $\frac{d\pi}{dq}$ is the the derivative of the projection function $\pi(q)$. The equations for these are given below.

$$\pi(\mathbf{q}) := \frac{1}{q_3}\mathbf{q} \in \mathbb{R}^4 \qquad \frac{d\pi}{d\mathbf{q}}(\mathbf{q}) = \frac{1}{q_3}\begin{bmatrix}1 & 0 & -\frac{q_1}{q_3} & 0\\ 0 & 1 & -\frac{q_2}{q_3} & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & -\frac{q_4}{q_3} & 1\end{bmatrix} \in \mathbb{R}^{4\times4}$$

The "IXV" term is given by the equation below.

$$I \otimes V := \begin{bmatrix}V & & \\ & \ddots & \\ & & V\end{bmatrix}$$

Once finished updating the landmark positions based on observations, we combine the IMU prediction and landmark update steps to perform the EKF update to find new landmark positions, each denoted by a mean mu and variance σ value.

We do not need to implement a prediction step for the landmarks since the vehicle's landmark capture sensor doesn't move significantly in the z-axis direction between timestamps. As such, only the x and y coordinates of landmarks are estimated.

IV. Results

We start by running our code to plot the estimated vehicle IMU trajectory with dead reckoning to test if our code runs correctly. For this, the algorithm is run with collected observation features sampled every 5 to speed up processing. The

update steps in our code are also commented out for dead-reckoning so that only the pose prediction step is run. The dead reckoning result is shown below in Fig. 2.

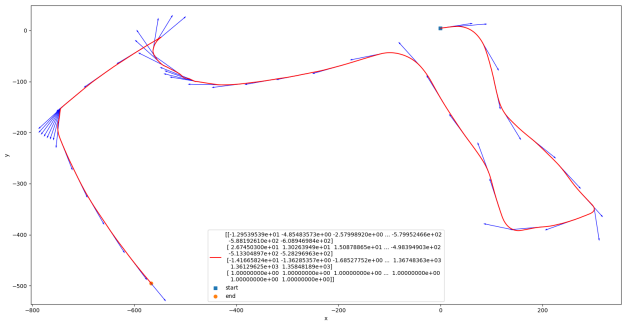

Fig. 2. estimated vehicle trajectory

The above result looks reasonable with the vehicle trajectory being relatively smooth.

We then run the full Visual-Inertial SLAM algorithm with landmark positions plotted as blue dots. For 200, 1000, 2000, and 3026 out of 3026 iterations, the result is shown below in Fig. 3, 4, 5, and 6.

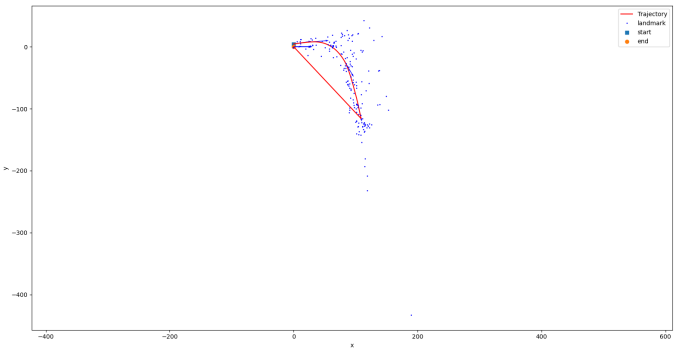

Fig. 3. estimated vehicle trajectory plus estimated landmark positions

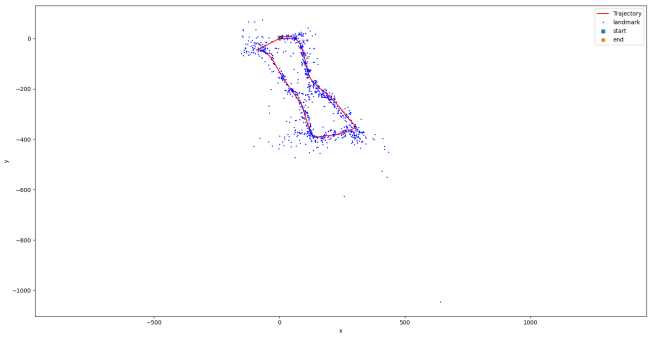

Fig. 4. estimated vehicle trajectory plus estimated landmark positions

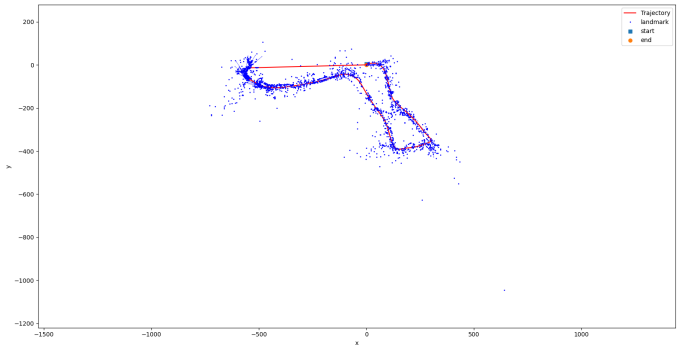

Fig. 5. estimated vehicle trajectory plus estimated landmark positions

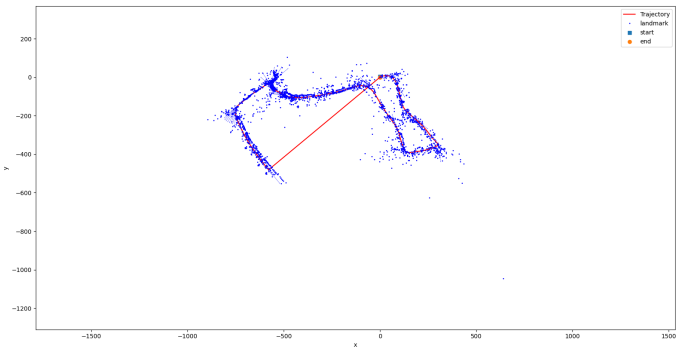

Fig. 6. estimated vehicle trajectory plus estimated landmark positions

The vehicle trajectory in the results looks smooth and the plotted landmark positions seem reasonable as they lie along the vehicle trajectory and are distributed along both sides of the vehicle.

V. Discussion

From the above results, the Visual-Inertial SLAM algorithm seems to do a good job of plotting the vehicle trajectory smoothly, but runs much slower than would be needed for real-time vehicle localization and environment mapping.

Problems that were run into while doing this project include figuring out how to use frame transformations, understanding how to use the feature data, and figuring out how to combine the prediction and update steps together for the EKF update step.

VI. Conclusion

This paper explores Visual-Inertial SLAM localization and mapping method, providing results which show it achieves good performance. There are more complex versions of Visual-Inertial SLAM to do vehicle localization, but our version used with a simple dataset shows Visual-Inertial SLAM performs well. In the future, the algorithm can be used with more captured observation data as well as texture mapping to increase the completeness of our mapping results. In the future, it also would be interesting to do a study on the runtime and accuracy tradeoff of Visual-Inertial SLAM.