Project 3: Visual-Intertial SLAM

Joseph Chang
Department of Electrical Computer Engineering
University of California, San Diego
jdchang@ucsd.edu

I. Introduction

Simultaneous Localization and Mapping (SLAM) is a Computer Vision algorithm used to track a car as it moves in the world. It creates a map of its surroundings and continually locates and updates the car's location within it. This technique is extremely useful for many applications dealing with unknown environments such as navigating disaster zones, underwater terrain, or acquiring intelligence in enemy territory. It is also used in consumer electronics such as cleaning robots and in medical devices for internal body mapping.

At first glance, SLAM appears to solve a chicken-and-egg problem since we want to build a map of the environment given the car's state trajectory, but also need to localize the car and estimate its trajectory given a map of the environment. There are numerous variations of SLAM developed to deal with this problem including the particle filter, Kalman filter, Factor Graphs SLAM, Fast SLAM, and Kinect Fusion.

II. Problem Formulation

In our experiment, a car navigates various roads. We are given timestamps *t* in UNIX standard seconds since epoch January 1, 1970 of each data collection. The car's linear velocity $v_t \in R^3$ and angular velocity $\omega_t \in R^3$ are measured in the body frame of the IMU. Stereo camera images are captured with pixel coordinates $z_t \in R^{4\times M}$ of detected visual features with precomputed correspondences between the left and right camera frames as shown below.

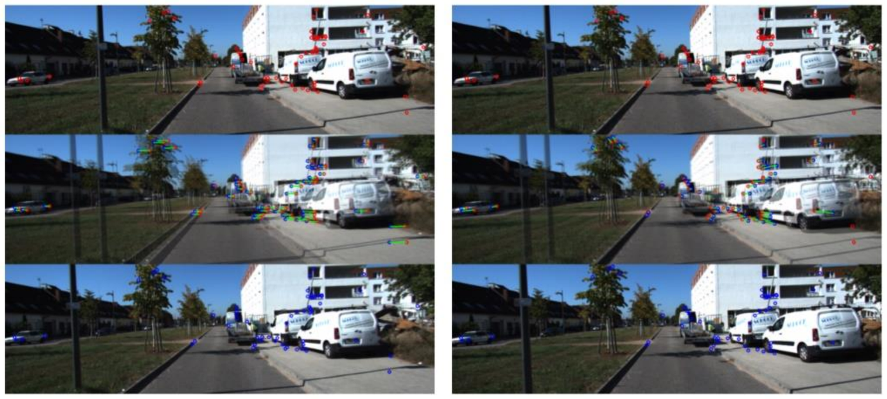

Figure 1: Visual features matched across left-right camera frames across time

The landmarks *i* that were not observable at time *t* have a measurement of $z_{t,i} = [-1\ -1\ -1\ -1]^T$. There are many features in each dataset so truncation methods are used to keep computational complexity manageable. The single camera intrinsic calibration is given in the following form where f_x and f_y are the x and y-component of the focal length in pixels while c_x and c_y are the pixels to the image center also known as the principal point.

$$K = \begin{bmatrix} \mathrm{f_x} & 0 & \mathrm{c_x} \\ 0 & \mathrm{f_y} & \mathrm{c_y} \\ 0 & 0 & 1 \end{bmatrix}$$

Since we are also given the stereo baseline b which is the distance between the stereo cameras in meters, we can compute the stereo camera calibration matrix as follows. The variables s_u and s_v refer to left and right stereo cameras respectively.

$$M = \begin{bmatrix} \mathrm{fs}_u & 0 & \mathrm{c}_u & 0 \\ 0 & \mathrm{fs}_v & \mathrm{c}_v & 0 \\ \mathrm{fs}_u & 0 & \mathrm{c}_u & -\mathrm{fs}_u b \\ 0 & \mathrm{fs}_v & \mathrm{c}_v & 0 \end{bmatrix}$$

The extrinsic calibration matrix from the IMU to the left camera optical frame in SE(3) is approximately as defined below where t1, t2, and t3 are translation in meters.

$${}_{\mathrm{cam}}T_{imu} = \begin{bmatrix} 0 & -1 & 0 & \mathrm{t1} \\ 0 & 0 & -1 & \mathrm{t2} \\ 0 & 0 & 0 & \mathrm{t3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The goal is to implement SLAM using the car’s IMU velocity data, calibration matrices, and features using the Extended Kalman Filter (EKF) for prediction and update steps. We generally follow the formulation shown below, though it is specific to our experiment. Through this, we can estimate the pose $\mathrm{T}_t \in \mathrm{SE}(3)$ of the IMU over time *t*.

$$\textbf{Prior}:\ \mathbf{x}_t \mid \mathbf{z}_{0:t} \sim \mathcal{N}(\boldsymbol{\mu}_{t|t}, \Sigma_{t|t})$$

$$\textbf{Prediction}:\ \begin{aligned} \boldsymbol{\mu}_{t+1|t} &= f(\boldsymbol{\mu}_{t|t}) = \begin{bmatrix} \mu^{\theta}_{t|t} + \tau\mu^{\omega}_{t|t} \\ \mu^{\omega}_{t|t} - \tau\frac{g}{L}\sin\mu^{\theta}_{t|t} \end{bmatrix} \\ \Sigma_{t+1|t} &= F(\mu_{t|t})\Sigma_{t|t}F(\mu_{t|t})^{\top} + W \end{aligned}$$

$$\textbf{Kalman gain}:\ K_{t+1|t} = \Sigma_{t+1|t}H(\mu_{t+1|t})^{\top}S^{-1}_{t+1|t}$$

$$\textbf{Update}:\ \begin{aligned} \boldsymbol{\mu}_{t+1|t+1} &= \boldsymbol{\mu}_{t+1|t} + K_{t+1|t}\nu_{t+1|t} \\ \Sigma_{t+1|t+1} &= \Sigma_{t+1|t} - K_{t+1|t}H(\mu_{t+1|t})\Sigma_{t+1|t} \end{aligned}$$

III. TECHNICAL APPROACH

We begin by implementing IMU localization using EKF prediction. First, we create the necessary variables from the IMU measurements as shown below. The linear velocity v_t and angular velocity ω_t are concatenated to create u_t. In u_t^, ω_t^ is the skew-symmetric version of ω_t and 0^T is 1×3 vector of zeros. A matrix is skew-symmetric matrix if its transpose equals its negative. The third matrix u_t with the curly hat above is called u_t adjoint and contains the skew symmetric version of v_t.

$$\mathrm{v}_t\hat{} = \begin{bmatrix} 0 & -\mathrm{v}(2) & v(1) \\ \mathrm{v}(2) & 0 & -\mathrm{v}(0) \\ -\mathrm{v}(1) & \mathrm{v}(0) & 0 \end{bmatrix}$$

$$\mathbf{u}_t := \begin{bmatrix} \mathbf{v}_t \\ \omega_t \end{bmatrix} \in \mathbb{R}^6 \quad \hat{\mathbf{u}}_t := \begin{bmatrix} \hat{\omega}_t & \mathbf{v}_t \\ \mathbf{0}^\top & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4} \quad \overset{\curlywedge}{\mathbf{u}}_t := \begin{bmatrix} \hat{\omega}_t & \hat{\mathbf{v}}_t \\ 0 & \hat{\omega}_t \end{bmatrix} \in \mathbb{R}^{6\times 6}$$

Next, we predict our IMU mean and covariance at the next timestamp given the current data using the equations below. Before the first iteration, $\mu_{t+1|t}$ and $\Sigma_{t+1|t}$ are initialized as 4×4 identity and 6×6 identity matrices respectively. Time discretization τ is the difference between two consecutive timestamps and W is the motion model noise which we define as a normal distribution centered at 0 with standard deviation 1 and size 6. Then, we estimate the pose of $\mathrm{T}_t \in \mathrm{SE}(3)$ of the IMU by taking the inverse of $\mu_{t+1|t}$ and saving it to our car's path at time *t* which is plotted later.

$$\mathrm{W} = \tau^2 * N\sim(0,1,6)$$

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

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

$$\mathrm{T}_t \in \mathrm{SE}(3) = (\mu_{t+1|t})^{-1}$$

The second step is to implement landmark mapping using EKF Update. We initialize the landmark mean $\mu_{t+1|t+1}$ as a 4×M matrix filled with -1 and the landmark covariance $\Sigma_{t+1|t+1}$ as a M×3M identity matrix multiplied by *V*. The scalar *V* is used to add noise and set to 10^{-4}. First, find all observable features at this sample by checking that $\mathrm{z}_{\mathrm{t,i}}$ does not contain $[-1\ -1\ -1\ -1]^T$ which is the default for non-observable features. If it is the first time the feature has been seen at that index, add it to the $\mu_{t+1|t+1}$ array. Otherwise, append it to an array of new observations.

For all new observations $\mathrm{z}_{t+1} \in R^{4xN_t}$, retrieve the prior mean $\mu_{t+1|t} \in \mathrm{SE}(3)$ and calculate the Jacobian *H* where ${}_{O}T_I$ is the transform from IMU to camera. Section 1 shown below is the projection function and its derivative. Section 2 is described by the equation below.

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

$$\begin{bmatrix} \mathbf{s} \\ 1 \end{bmatrix}^{\odot} := \begin{bmatrix} I & -\hat{\mathbf{s}} \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times 6}$$

$$H_{i,t+1|t} = \underbrace{M\frac{d\pi}{d\mathbf{q}}\left({}_OT_I\boldsymbol{\mu}_{t+1|t}\mathbf{m}_j\right)}_{1}\,{}_OT_I\,\underbrace{\left(\boldsymbol{\mu}_{t+1|t}\mathbf{m}_j\right)^{\odot}}_{2} \in \mathbb{R}^{4\times 6}$$

We are now ready to perform to the EKF update. The Kalman Gain *K*, mean update $\mu_{t+1|t+1}$, and covariance update $\Sigma_{t+1|t+1}$ are calculated as follows where z-z^ is known as the innovation.

$$K_{t+1|t} = \Sigma_{t+1|t}H_{t+1|t}^{\top}\left(H_{t+1|t}\Sigma_{t+1|t}H_{t+1|t}^{\top} + I \otimes V\right)^{-1}$$
$$\boldsymbol{\mu}_{t+1|t+1} = \exp\left(\left(K_{t+1|t}(\mathbf{z}_{t+1} - \tilde{\mathbf{z}}_{t+1})\right)^{\wedge}\right)\boldsymbol{\mu}_{t+1|t}$$
$$\Sigma_{t+1|t+1} = (I - K_{t+1|t}H_{t+1|t})\Sigma_{t+1|t}$$

$$H_{t+1|t} = \begin{bmatrix} H_{1,t+1|t} \\ \vdots \\ H_{N_{t+1},t+1|t} \end{bmatrix}$$

Finally, we combine the IMU prediction step and landmark update step with an IMU update step based on the stereo camera observation model to obtain a complete Visual-Inertial SLAM algorithm. Updating the IMU and landmarks covariance simultaneously is more accurate than updating separately because it acknowledges the non-zero correlation between the map state and the IMU inverse pose. We attempted to complete this portion of the programming assignment, but were unsuccessful so it is commented out. Nonetheless, here is the conceptual basis behind it.

We are doing localization and mapping jointly using one combined filter state $\mathrm{U}_{t,m}$ consisting of the IMU inverse pose and landmark positions. The joint covariance has dimension (3M+6)×(3M+6) and an associated mean. It can be updated in one shot while also computing the (3M+6) perturbation which is K*innovation. Adding the perturbation to the previous mean can be done separately because we have already captured the correlation when multiplying by K. Separate equations are used to calculate the Jacobian *H*, for $\mathrm{z}_{t|\mathrm{i}}$ and $\mathrm{z}_{t+1|\mathrm{i}}$ as below.

$$\tilde{\mathbf{z}}_{t,i} := M\pi\left({}_OT_IU_t\underline{\boldsymbol{\mu}}_{t,j}\right) \in \mathbb{R}^4 \qquad \text{for } i = 1,\ldots,N_t$$

$$H_{t,i,j} = \begin{cases} M\frac{d\pi}{d\mathbf{q}}\left({}_OT_IU_t\underline{\boldsymbol{\mu}}_{t,j}\right){}_OT_IU_tP^{\top} \\ \\ \mathbf{0} \in \mathbb{R}^{4\times 3} \end{cases}$$

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

$$H_{i,t+1|t} = M\frac{d\pi}{d\mathbf{q}}\left({}_OT_I\boldsymbol{\mu}_{t+1|t}\mathbf{m}_j\right){}_OT_I\left(\boldsymbol{\mu}_{t+1|t}\mathbf{m}_j\right)^{\odot} \in \mathbb{R}^{4\times 6}$$

The Kalman Gain *K* is also calculated and results in a 3M+6×4N$_t$ matrix. The Jacobians and Kalman Gain are used to update the covariance matrix at each sample along with the innovation.

IV. Results

Our Visual-Inertial SLAM implementation is run on three datasets stored in .npz files. Each dataset contains timestamps, features, linear velocity, angular velocity, an intrinsic camera matrix, baseline, and an extrinsic matrix transforming the IMU frame to camera frame. We downsample the features to only keep every tenth sample. When there are too many features, IDLE encounters memory errors when initializing the landmark covariance $\Sigma_{t+1|t+1}$ which depends on the number of features. After IMU prediction and landmark updates at all samples, the car path is displayed using the visualize_trajectory_2d function provided in utils.py. The red path is the car's path, blue dots are detected landmarks, blue square is start location, and orange circle is end location.

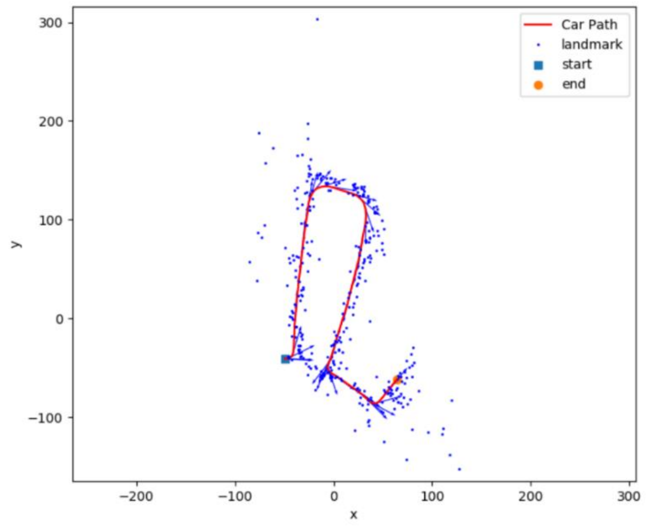

Figure 3: Data 0022

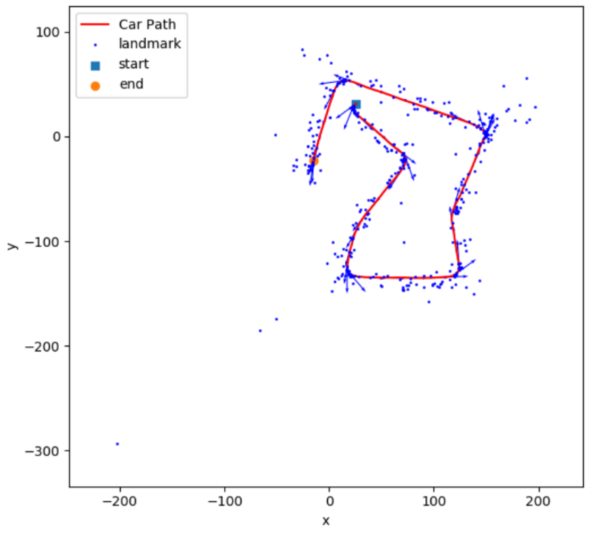

Figure 4: Data 0027

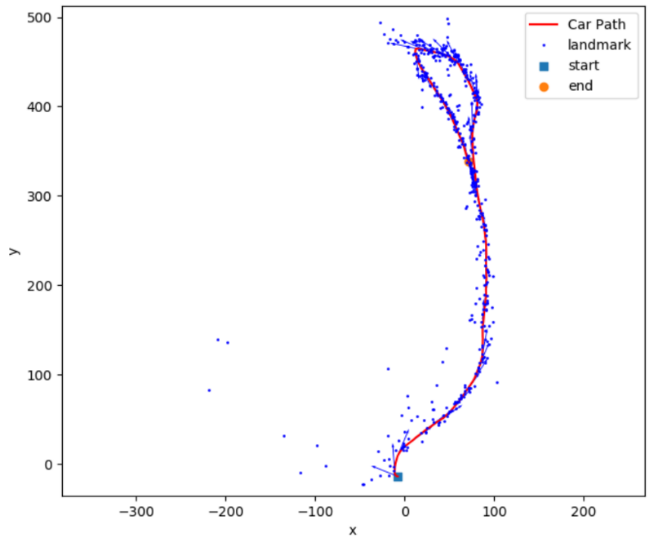

Figure 5: Data 0034

Comparing the resulting car path to the video of the car driving with feature tracking over time, they are identical so the algorithm is working. Turns are happening and landmarks are showing up at the correct locations. If IMU updates were implemented, accuracy would be even better, but without it, the results match the car's actual trajectory fairly accurately.

V. Conclusion

This project successfully implements Visual-Inertial SLAM for a car equipped with stereo cameras, and an IMU. It is an effective technique for mapping an environment and tracking the car's path. Although this implementation reads pre-collected data, the algorithm could easily be tweaked to receive live data as the car moves around so it can see obstacles and react. We were not able to complete part c of the project which adds IMU updates to improve accuracy, but it does not change the fact mapping with IMU prediction and landmark updates produces respectable results. Future works would be implementing IMU updates.