VISUAL INERTIAL SLAM

Joseph Chang

Department of Electrical and Computer Engineering, University of California, San Diego

ABSTRACT

There are many scenarios in the real-world that require calculating the position of an object relative to its environment. This is used in cleaning robots autonomous vehicles. It's also crucial for the robot to build a map of the environment so it can avoid crashing into objects or walls it has already detected. This is solved by simultaneous localization and mapping aided by many sensor data collected from a moving vehicle.

Index Terms— computer vision, SLAM, Extended Kalman Filter

1. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) is an important topic in computer vision. It is used to gain a 2D understanding of the world from inertial measurement unit (IMU) data and a stereo camera as well as intrinsic camera calibration and the extrinsic calibration between the sensors. The algorithm consists of two primary steps in mapping and localization.For mapping, we build a map of the environment given a vehicle's trajectory and surrounding landmarks. In localization, we estimate the vehicle's pose given IMU data and so the cycle continues. There are other popular sparse SLAM algorithms including the Particle Filter and Factor Graphs SLAM. However, we choose to use the Extended Kalman Filter (EKF) method which uses prediction and update steps to track a vehicle over time and estimate landmark positions. It can accurately track a robot's position faster than Particle Filter SLAM. In this paper, we implement EKF SLAM to create a map of a vehicle's environment as it drives and track it's position.

2. PROBLEM FORMULATION

Our problem is to localize the car's pose over time to build a map with its trajectory and surrounding landmarks.

Dataset: We are given synchronized IMU measurements from a car which contains the linear velocity $v_t \in R^3$ and angular velocity $\omega_t \in R^3$ measured in the frame of the IMU. All data is synchronized to timestamps τ_t given in UNIX standard seconds-since-the-epoch January 1, 1970. We are also given pre-computed features $z_{t,i}$ which are left and right image pixels matched across the stereo camera frames across time. If there are no features observed at a specific timestamp, the returned data is denoted by $[-1,-1,-1,-1]^T$. As the number of features is large, we downsample it by selecting every 5, 10, 50, or 100 for optimization. The intrinsic camera calibration contains stereo baseline b which the distance between cameras and intrinsic calibration matrix K.

$$\text{K} = \begin{bmatrix} fs_u & 0 & c_u \\ 0 & fs_v & c_v \\ 0 & 0 & 1 \end{bmatrix}$$

From this, we can formulate the complete stereo camera calibration matrix M where f is focal length, su, sv is pixel scaling [pixels/m], and cu, cv is principal point [pixels].

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

The given extrinsic calibration ${}_IT_C \in SE(3)$ specifies the transformation from the left camera to IMU frame. Inverting it, we obtain:

$${}_CT_I = {}_IT_C{}^{-1}$$

Visual Mapping: Visual mapping is used to represent represent the environment by locating a car's pose in the world given noisy sensor observations. Once the car is localized and its pose x_t is known, observations z_t sensed by the robot at a particular timestamp can be translated into landmarks m_i in the world frame. As the car continues moving, more landmarks will appear on the map as dots. Areas with dense landmarks are objects in the environment. The goal of visual mapping is to estimate landmarks as follows where z_t are observation features, x_t is the car's pose, m are landmarks, and n_t are landmark indices at z_i, t for observations $i = 1, ..., K$.

$$p(z_t|x_t, m, n_t)$$

Localization: Given a map m, sequence of control inputs $u_{0:t-1}$, and sequence of sensor measurements $z_{0:t}$, the car state trajectory $x_{0:t}$ can be inferred. We maintain a pmf of the car state over time. A prediction step is performed using the motion model p_f to obtain the predicted pmf.

$$p_{t+1|t}(x_{t+1})$$

An update step is performed using the observation model p_h to obtain the updated pmf. The particle poses remain unchanged, but the weights are scaled by the observation model.

$$p_{t+1|t+1}(x_{t+1})$$

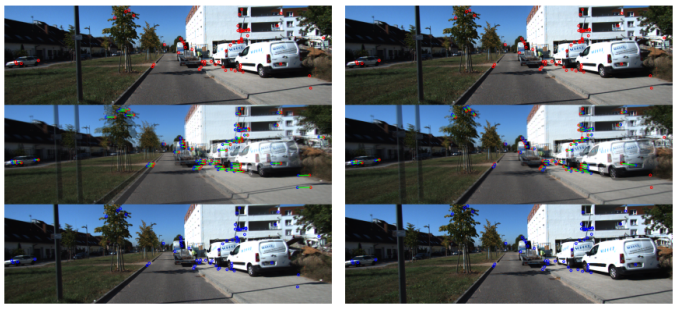

Fig. 1: Visual features matched across the left-right camera frames (left) and across time (right).

3. TECHNICAL APPROACH

The goal of this project is to use an Extended Kalman Filter (EKF) to predict and update the car's pose and trajectory over time. The prediction step is based on SE(3) kinematics with IMU measurements and the EKF update step based on the stereo camera observation model with feature observations to perform localization and mapping.

IMU Localization: We implement the EKF prediction step based on the SE(3) kinematics and the linear and angular velocity measurements to estimate the pose $T_t \, \epsilon \, SE(3)$ of the IMU over time t. Given the IMU's current pose $\mu_{t|t}$ and $\Sigma_{t|t}$ which are initialized as $I \, \epsilon \, R^{4x4}$ and $I \, \epsilon \, R^{6x6}$ respectively, the pose prior is as follows. $\Sigma_{t|t}$ is 6x6 because only the 6 degrees of freedom of T_t are changing.

$$T_t | z_{0:t}, u_{0:t-1} \; N(\mu_{t|t}, \sigma_{t|t})$$

This means that

$$T_t = \mu_{t|t} exp(\hat{\delta}\mu_{t|t})$$

where

$$\delta\mu_{t|t} \sim N(0, \Sigma_{t|t})$$

Then, we rewrite the motion model using the nominal kinematics of $\delta\mu_{t|t}$ with time discretization τ:

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

With the motion model defined above, we derive the following EKF Prediction Step where $w_t \sim N(0, W)$. In this scenario, v_t are the linear velocities and w_t are the angular velocities obtained from the IMU. τ is the difference between the current and previous timestamps and W is the variance of Gaussian noise w_t with mean 0. u_{hat} is a 4x4 matrix and comes from the Lie algebra of the pose space after taking the 6D vector and transforming it into a 4x4. $u_{curlyhat}$ is a 6x6 matrix and comes from the Lie algebra of the adjoint space since it's needed to update the 6x6 covariance.

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

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

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

$$\mu_{t+1|t} = \mu_{t|t} exp(\tau \hat{u}_t)$$

$$\Sigma_{t+1|t} = E[\delta\mu_{t+1|t}\delta\mu_{t+1|t}^T] = exp(-\tau \overset{\curlywedge}{u_t})\Sigma_{t|t} exp(-\tau \overset{\curlywedge}{u_t})^T + W$$

Then, transforms between the camera and world frames are updated.

$$_C T_W = {}_C T_I \cdot \mu_{t+1|t}$$
$$_W T_C = {}_C T_W{}^{-1}$$

Feature Matching: Given a left and right stereo image, the goal is to find corresponding features between them. For $G(z)$ to be invertible, the region $W(z)$ must have nontrivial gradients along independent directions resmbling a corner. A corner is the pixel z such that the smallest eigenvalue of $G(z)$ is larger than some threshold ρ. The Harris corner detector is s a variation of the regular corner detector that thresholds as follows where $k \, \epsilon \, [0.04, 0.06]$ is a small scalar and λ_1, λ_2 are eigenvalues of G. Since k is small, both eigenvalues of G need to be large enough to pass the threshold.

$$\lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 = det(G) - k \, tr^2(G) \geq \rho$$

Now that we have corners features detected on the left image, optical flow can track the features on the right image. Temporal feature tracking. Finally, we perform temporal feature tracking using the features in the left image to the left image at the next timestamp.

Landmark Mapping: Once the IMU trajectory is predicted correctly, we focus on estimating the landmark positions. The landmark poses μ_t and Σ_t are initialized as unobserved $[-1, -1, -1, -1]^T \epsilon R^{3M}$ and $I \cdot V \, \epsilon \, R^{3M\times3M} + V$ respectively where V is the variance for Gaussian noise. We implement an EKF by first checking if there are newly observed features at a specific timestamp. The unknown landmark positions $m \, \epsilon \, R^3$ as a state.

Given a new observation z_{t+1}, the predicted observations based on μ_t and known correspondences δ_{t+1} are as follows:

$$\tilde{z}_{t+1,i} = M\pi({}_O T_I T_{t+1}^{-1} \mu_{t,j}) \, \epsilon \, R^4 \quad for \;\; i = 1, ..., N_{t+1}$$

Considering a perturbation $\delta\mu_{t,j} \epsilon R^3$ for the position of landmark j, a landmark is described as:

$$m_j = \mu_{t,j} + \delta\mu_{t,j}$$

The projection matrix then is shown below such that $m_j = P\underline{m}_j$.

$$P = [I \ 0] \epsilon R^{3x4}$$

The projection function $\pi(q)$ and its derivative $\frac{d\pi}{dq}(q)$ are calculated as follows:

$$\pi(q) = \frac{1}{q_3} \epsilon R^4$$

$$\frac{d\pi}{dq}(q) = \frac{1}{q_3} = \begin{bmatrix} 1 & 0 & -q1/q3 & 0 \\ 0 & 1 & -q_2/q_3 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -q_4/q_3 & 1 \end{bmatrix} \epsilon R^{4\times4}$$

Then, the we solve the Jacobian $H_{t+1,i,j}$ of $\tilde{z}_{t+1,i}$ with respect to landmark m_j evaluated at the car's current position $\mu_{t,j}$. The second equation below is equivalent.

$$H_{t+1,ij} = \begin{cases} M\frac{d\pi}{d\mathbf{q}}\left({}_OT_I\, T_{t+1}^{-1}\underline{\boldsymbol{\mu}}_{tj}\right){}_OT_I\, T_{t+1}^{-1}P^\top & \text{if } \Delta_t(j) = i \\ \mathbf{0}, \in \mathbb{R}^{4\times3} & \text{otherwise} \end{cases}$$

$$H_{t+1,i,j} = M\frac{d\pi}{dq} \cdot {}_CT_W \cdot P^T$$

We now perform the EKF update step after every visual observation z_t. This keeps track of the mean and covariance of landmarks m. The updated Kalman Gain K_{t+1} is used to update the mean and covariance.

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

Note that we do not need to implement a prediction step because the landmarks are assumed static. We also assume the z-coorinates for all landmarks are 0 and focus on estimating xy-coordinates because the sensor moves minimally along the z-axis.

Visual-Inertial SLAM: Combining the IMU prediction step with the landmark update step, we implement an IMU update step based on the stereo camera observation model to obtain a complete visual-inertial SLAM algorithm. Using EKF prediction, we localize the new IMU pose and append it to the car's trajectory. Now, we iteratively plot the car's trajectory and new landmarks observed.

4. RESULTS

We run our SLAM algorithm on synchronized IMU and stereo camera feature data. The initial run is dead reckoning where we calculate the car's pose based on the previously determined position, linear velocity, and angular velocity over time. Here, we only perform IMU-based localization and ignore updating landmarks. The result is in Fig. 2. and shows the car's raw trajectory.

Then, we plot our world-frame map after downsampling our landmark features by factors of 5, 10, 50, and 100. The total number of iterations is the number of downsampled features. After the algorithm has finished running, the map displays the car trajectory and environment landmarks.

The resulting plots show good results in Fig. 3-5. The car starts from the top right of the map and begins driving down to the bottom left on winding path. The car trajectory is the red curve and landmarks are shown clearly on both sides of the car. The map is a smooth curve and looks like a road. It matches the stereo image data from the front cameras of the car. As the number of features is downsampled, the landmarks become more sparse as expected, but the algorithm runs much faster.

The thin, blue quivers show the car's yaw or orientation at each timestamp. They show the car turning at bends in the road, but aren't very visible when the car is moving relatively straight because they are covered by the red trajectory curve.

5. CONCLUSION

In this paper, we implemented a Visual-Inertial SLAM algorithm to localize a driving car and its surrounding environment with IMU detections onto a world-frame map. We tracked the car's pose with IMU-based localization using EKF prediction by its linear velocity, angular velocity, and added noise. The landmark coordinates were successfully estimated from the observable feature correspondences detected by the stereo camera. Further research would be increasing the accuracy of the algorithm by optimizing parameters and vectorizing some loops in the code to increase computational efficiency. A method could also be created to quantify the algorithm's performance.

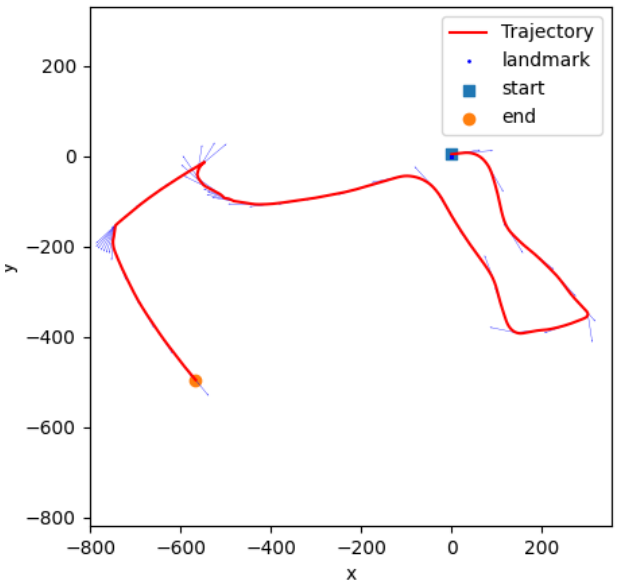

Fig. 2: Dead Reckoning Map

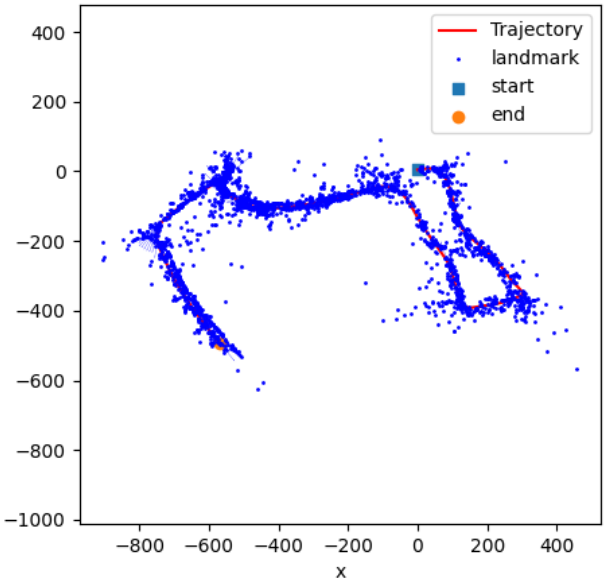

Fig. 3: VI-SLAM Map, Landmarks downsampled by 5

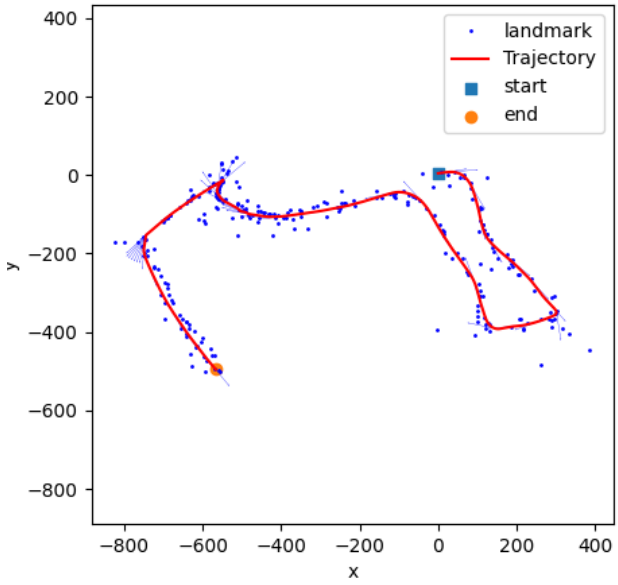

Fig. 5: VI-SLAM Map, Landmarks downsampled by 50

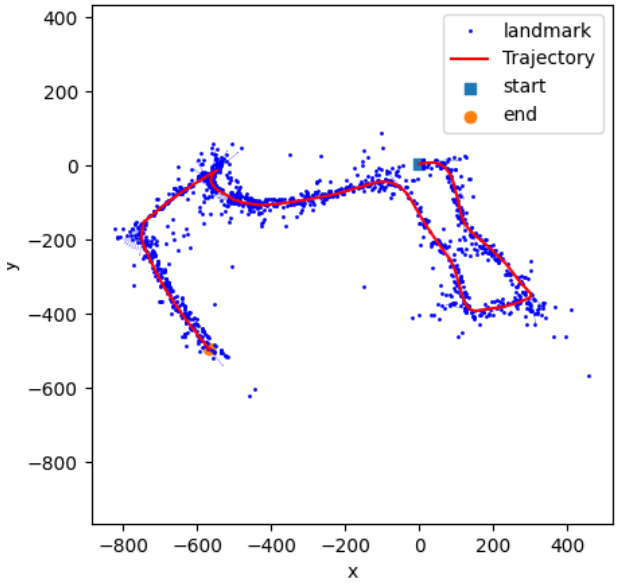

Fig. 4: VI-SLAM Map, Landmarks downsampled by 10

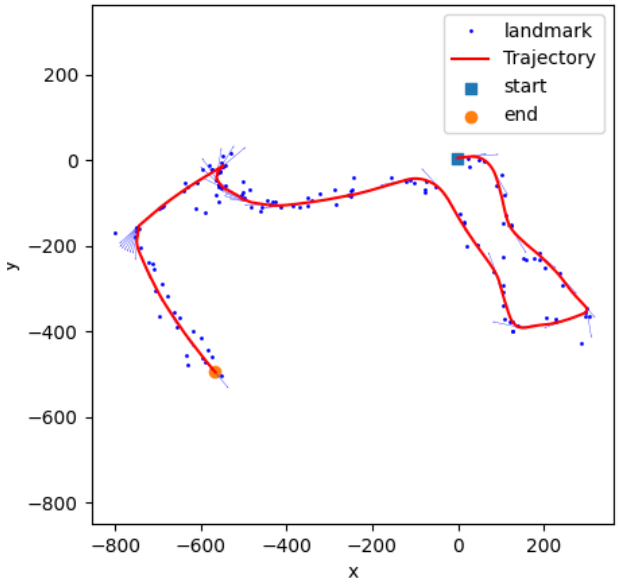

Fig. 6: VI-SLAM Map, Landmarks downsampled by 100