Project 3: Visual-Inertial SLAM

Prathamesh Saraf (A59015739)
Department of Electrical and Computer Engineering
University of California, San Diego
psaraf@ucsd.edu

***Abstract*—The project focuses on the implementation of a visual-inertial simultaneous localization and mapping (SLAM) algorithm using an Extended Kalman filter (EKF). The algorithm utilizes synchronized measurements from an inertial measurement unit (IMU) and a stereo camera, with provided intrinsic and extrinsic calibration parameters. The algorithm consists of three parts - an EKF prediction step based on SE(3) kinematics equations and IMU measurements to estimate the pose of the IMU over time, an EKF update step using visual feature observations to estimate the positions of observed landmarks, and combining the IMU prediction step with the landmark update step to implement an update step for the IMU pose based on the stereo-camera observation model, thus obtaining a complete visual-inertial SLAM algorithm.**

***Index Terms*—Visual-inertial SLAM, Extended Kalman Filter, Stereo Camera, Pose Estimation**

I. Introduction

Simultaneous Localization and Mapping (SLAM) is an essential problem in robotics that has been extensively studied in recent years. In this project, we implement a complete SLAM algorithm by combining an Extended Kalman Filter (EKF) prediction step based on SE(3) kinematics with IMU measurements, and an EKF update step based on stereo-camera observation model with visual feature observations. The algorithm allows us to estimate the pose of a robot and the locations of the landmarks observed by it in an unknown environment.

First, we perform IMU localization via EKF prediction. We implement an EKF prediction step based on SE(3) kinematics equations and the linear and angular velocity measurements from the IMU to estimate the pose of the IMU over time. By using the EKF algorithm, we can keep track of the mean and covariance of the pose estimate, which allows us to update it based on new measurements over time. Next, we estimate the landmark positions observed by the robot using visual feature measurements. Assuming that the predicted IMU trajectory from the first step is correct, we will focus on estimating the landmark positions observed in the images. We will implement an EKF with the unknown landmark positions as a state and perform EKF update steps using the visual observations to keep track of the mean and covariance of the landmarks. Assuming that the static landmarks are static, we estimate the x and y coordinate using the EKF algorithm. Finally, we combine the IMU prediction step with the landmark update step to obtain a complete visual-inertial SLAM algorithm. We will implement an update step for the IMU pose based on the stereo-camera observation model to obtain accurate estimates of the pose and the landmark positions. This enables us to perform SLAM in an unknown environment accurately.

The following sections describe the mathematical problem formulation and the technical approach to implementing the algorithm along with the results obtained based on the given dataset.

II. Problem Formulation

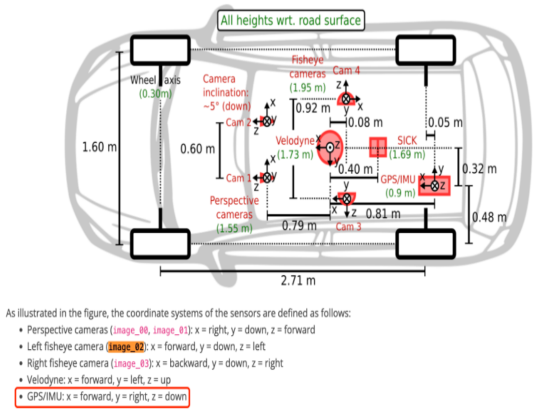
A. Extended Kalman Filter - Prediction Step

The Extended Kalman Filter (EKF) is a variant of the Kalman Filter that can handle nonlinear systems and is used to estimate the states of a system based on a series of measurements. The EKF linearizes the system around an operating point, propagates the state and covariance estimates using the linearized dynamics, and updates them based on nonlinear measurements.

$$\mathbf{x}_{t+1} = f\left(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t\right) \sim p_f\left(\cdot \mid \mathbf{x}_t, \mathbf{u}_t\right) \tag{1}$$

With mean and covariance:

$$\begin{aligned}\boldsymbol{\mu}_{t+1|t} &= \boldsymbol{\mu}_{t|t} \exp\left(\tau_t \hat{\mathbf{u}}_t\right) \\ \Sigma_{t+1|t} &= \mathbb{E}\left[\delta\boldsymbol{\mu}_{t+1|t}\delta\boldsymbol{\mu}_{t+1|t}^\top\right] = \exp\left(-\tau\hat{\mathbf{u}}_t\right)\Sigma_{t|t}\exp\left(-\tau\hat{\mathbf{u}}_t\right)^\top + W\end{aligned} \tag{2}$$



As illustrated in the figure, the coordinate systems of the sensors are defined as follows:

- Perspective cameras (image_00, image_01): x = right, y = down, z = forward
- Left fisheye camera (image_02): x = forward, y = down, z = left
- Right fisheye camera (image_03): x = backward, y = down, z = right
- Velodyne: x = forward, y = left, z = up
- GPS/IMU: x = forward, y = right, z = down

Fig. 1. The car configuration model

B. Extended Kalman Filter - Update Step

The second objective is to estimate landmark positions using an EKF with previously predicted IMU trajectory as input. Landmark positions are the state variable represented by a 3xM vector, and the EKF update step is performed after every visual observation to maintain the mean and covariance of landmark positions. As landmarks are assumed to be stationary. This method is frequently used in robotics to determine a mobile robot's position and orientation using fixed landmark locations. EKF provides a precise and resilient solution for landmark estimation. Thus we need to estimate the homogeneous coordinates (m) in the world frame based on visual observations. The observation model:

$$\mathbf{z}_t = h\left(\mathbf{x}_t, \mathbf{v}_t\right) \sim p_h\left(\cdot \mid \mathbf{x}_t\right) \tag{3}$$

The EKF update equations used for this step are:

$$H_{t+1,i,j} = \begin{cases} K_s \frac{d\pi}{dq}\left(oT_l T_{t+1}^{-1} \underline{\mu}_{t,j}\right) oT_l T_{t+1}^{-1} P^\top, & \text{if } \Delta_t(j) = i, \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{4}$$

$$K_{t+1} = \Sigma_t H_{t+1}^\top \left(H_{t+1} \Sigma_t H_{t+1}^\top + I \otimes V\right)^{-1} \tag{5}$$

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + K_{t+1}\left(\mathbf{z}_{t+1} - \tilde{\mathbf{z}}_{t+1}\right) \tag{6}$$

$$\Sigma_{t+1} = \left(I - K_{t+1} H_{t+1}\right) \Sigma_t \tag{7}$$

H is the Jacobian matrix which requires to be updated for each step. The jacobian matrix changes its shape dynamically throughout the update step since it depends on the number of visible and visited features. Except this, the other parameters have constant shapes and are only updated at every step.

Also, as seen landmarks align closely with the car's path, implying that the mapping was successful. To speed up computation, the features were downsampled. Specifically, every 10th feature was used for each of the datasets. ekf prediction update off map

$$\begin{aligned} z_0 &= \frac{f s_u b}{u_L - u_R} \\ x_0 &= \frac{z_0\left(u_L - c_u\right)}{f s_u} \\ y_0 &= \frac{z_0\left(v_L - c_V\right)}{f s_V} \end{aligned} \tag{8}$$

C. *Visual Inertial SLAM*

The third task involves merging the IMU prediction step from part (A), the landmark update step from part (B), and an IMU update step based on the stereo camera observation model to obtain a comprehensive visual-inertial SLAM algorithm. The new feature in this step is updating the IMU pose based on the stereo camera observation model, similar to part (B). However, the first-order Taylor series approximation is expanded at an inverse IMU pose, using a pose perturbation t+1—t+1, instead of the landmark perturbation t,j used in part (B). The objective is to update the IMU pose Ut, given visual feature observations z0:T and inverse IMU pose Ut, using the observation model and sensor value. This step assumes the same set of assumptions as in the previous tasks.

III. Technical Approach

This section provides the technical approach and mathematical equations used to solve the problem formulations and highlights the specific techniques employed in the code to achieve the desired result. The EKF approach presents a robust and efficient solution to the visual-inertial SLAM problem by estimating the robot's pose and landmark locations. The Technical Approach section serves as a valuable resource for researchers and practitioners looking to implement EKF for visual-inertial SLAM.

A. *Prediction*

Once all time steps have been iterated, the estimated means are used to extract the full trajectory of the IMU pose. The equation reveals that when the update step is not implemented, the means for all time steps are not influenced by the covariances. This can be viewed as a simplification of the pose estimation process, however, it cannot be considered as an implementation of the EKF since the Kalman gain is not computed. It's important to note that the Kalman gain is a crucial factor in the EKF, as it adjusts the balance between the predicted state and the observed measurements, resulting in more accurate state estimates. Therefore, the omission of the Kalman gain would compromise the effectiveness and accuracy of the EKF approach for pose estimation. The prediction equations used:

$$T_k := T\left(t_k\right), \tau_k := t_{k+1} - t_k \tag{9}$$

$$\left[\begin{array}{c} \hat{\mathbf{v}} \\ \boldsymbol{\omega} \end{array}\right] := \left[\begin{array}{cc} \hat{\omega} & \mathbf{v} \\ \mathbf{0} & 0 \end{array}\right] \tag{10}$$

$$T_{k+1} = T_k \exp\left(\tau_k \hat{\zeta}_k\right) \tag{11}$$

where,

$$\begin{aligned} \boldsymbol{\mu}_{t+1 \mid t} &= \boldsymbol{\mu}_{t \mid t} \exp\left(\tau_t \hat{\mathbf{u}}_t\right) \\ \Sigma_{t+1 \mid t} &= \exp\left(-\tau \hat{\mathbf{u}}_t\right) \Sigma_{t \mid t} \exp\left(-\tau \hat{\mathbf{u}}_t\right)^\top + W \end{aligned} \tag{12}$$

$$\begin{gathered} u_t = \left[\begin{array}{c} v_t \\ \omega_t \end{array}\right] \\ \hat{u_t} = \left[\begin{array}{cc} \hat{\omega}_t & v_t \\ 0 & 0 \end{array}\right] \in \mathbf{R}^{\mathbf{4} \times \mathbf{4}} \\ \widehat{u_t} = \left[\begin{array}{cc} \hat{\omega}_t & v_t \\ 0 & \hat{\omega}_t \end{array}\right] \mathbf{R}^{\mathbf{6} \times \mathbf{6}} \end{gathered} \tag{13}$$

B. *Update*

To implement landmark mapping, predicted observations are computed using the robot pose and intrinsic camera matrix. The map is updated by finding visible features, classifying them as first-time or revisited, and using the inverse camera model to obtain world coordinates for first-time features. For revisited features, the zTilda and Jacobian matrix are found to update the mean and covariance matrix. To speed up debugging and computation, features were down-sampled, with every 10th feature used to update the map and a separate array created for these features.

We first extract all the visible features from the camera data. The invisible features are the ones for which the observation values are [-1, -1, -1, -1]. After extracting all the visible features from the dataset at each timestep, we run the update step and compute/update the mean and covariance matrices. These are then used by the observation model to compute the new observations. Using the new observations, we update the Kalman gain matrix, then use it to compute new step mean and covariance matrices. The equations for the update step are given below:

$$\tilde{\mathbf{z}}_{t+1,i} = K_s \pi\left(O, T_{t+1}^{-1} \underline{\mu}_{t,j}\right) \in \mathbb{R}^4 \quad \text{for } i = 1, \ldots, N_{t+1} \tag{14}$$

$$\begin{bmatrix} u_L \\ v_L \\ u_R \\ v_R \end{bmatrix} = \underbrace{\begin{bmatrix} f_{s_u} & 0 & c_u & 0 \\ 0 & fs_v & c_v & 0 \\ f_{s_u} & 0 & c_u & -fs_u b \\ 0 & f_{s_v} & c_v & 0 \end{bmatrix}}_{M} \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{15}$$

C. Pesudo Code for the EKF SLAM

This pseudocode outlines the process of importing data, resampling features, and initializing the IMU pose mean and covariance matrix, the covariance matrix, and the revisited feature tracker. Then, for each timestamp, the algorithm finds visible features and applies the motion model to predict the pose mean and covariance. For each visible feature, the algorithm checks whether it is visible for the first time or not. If it is visible for the first time, the feature is projected from IMU coordinates to the world frame, and the revisited feature tracker is updated. Otherwise, the algorithm removes disparity features, applies the EKF mapping equations, calculates the predicted observations zTilda, calculates the Jacobian for features and body, calculates the Kalman gain, updates the body pose mean, updates the feature mean, calculates the covariance, and replaces the previous covariance with new values.

Algorithm 1: EKF SLAM Algorithm

```
Input: Data Import
Output: Updated Pose Mean and Covariance Matrix
Feature Resampling - Lowering the number of
  features; Initialization of IMU pose mean and
  covariance matrix; Initialization of covariance matrix;
  Initialization of Revisited Feature Tracker; for $t$ in
  timestamps do
    Finding visible features at a given timestamp;
      Prediction step - Applying the Motion Model to
      predict the Pose mean and covariance.; for visible
      $j$ in visible feature do
        if visible for the first time then
            Project feature from IMU coordinates to
              World frame; Update revisited feature
              tracker;
        else
            Remove disparity features; Apply EKF
              Mapping equations; Calculate predicted
              Observations $zTilda$; Calculate Jacobian
              for features and body; Calculate Kalman
              Gain; Update Body Pose Mean; Update
              Feature Mean; Calculate Covariance;
              Replace the previous covariance with new
              values;
```

D. Visual SLAM

This section employs a prediction step that closely follows the EKF prediction step for the robot's motion model. The primary deviation, however, lies in the requirement to update the covariance between the robot and landmarks, as well as within themselves. As a result, the covariance has a shape of R3M+63M+6. The equations given below exaplin the vectorised implementation for fast computing. This way all the updates are computed in one go instead of indiviual update which increases the efficiency and reduces the computation time.

$$\begin{aligned} \Sigma_t &= \begin{bmatrix} \Sigma_{LL} & \Sigma_{LR} \\ \Sigma_{RL} & \Sigma_{RR} \end{bmatrix} \\ \Sigma_{LL} &\in \mathbf{R}^{\mathbf{3M+6\times 3M+6}} \\ \Sigma_{RR} &\in \mathbf{R}^{6\times 6} \end{aligned} \tag{16}$$

$$\begin{aligned} \boldsymbol{\mu}_{t+1|t} &= \boldsymbol{\mu}_{t|t} \exp\left(\tau_t \hat{\mathbf{u}}_t\right) \\ \Sigma_{RR_{t+1}} &= \exp\left(-\tau \hat{\mathbf{u}}_t\right) \Sigma_{RR_t} \exp\left(-\tau \widehat{\mathbf{u}}_t\right)^\top + W \end{aligned} \tag{17}$$

$$\begin{aligned} H_{\text{land}} &= K_s \frac{d\pi}{dq}\left({}_oT_I \mu_R^{-}1 \mu_L\right)_o T_I \mu_R^{-}1 P^T \\ H_{\text{land}} &= K_s \frac{d\pi}{dq}\left({}_oT_I \mu_R^{-}1 \mu_L\right)_o T_I \left(\mu_R \mu_L\right) \end{aligned} \tag{18}$$

$$K_{t+1} = \Sigma_{t+1|t} H_{t+1}^T \left(H_{t+1} \Sigma_{t+1|t} H_{t+1}^T + I \otimes V\right)^{-} 1 \tag{19}$$

$$\mu_{t+1|t+1} = \mu_{t+1|t} \exp\left(\left(K_{t+1}\left(z_{t+1} - z_{t+1}\right)\right)\right) \tag{20}$$

$$\Sigma_{t+1|t+1} = \left(I - K_{t+1}H_{t+1}\right)\Sigma_{t+1|t} \tag{21}$$

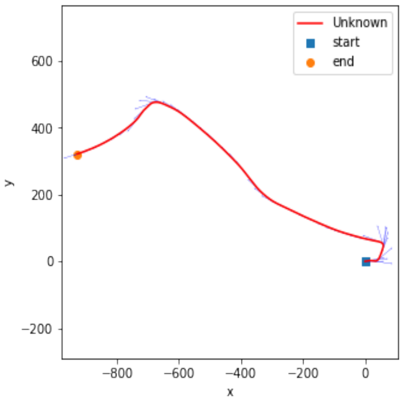
IV. Results

This section presents the plots for:

1) Prediction and localization step trajectory for datasets 03 and 10
2) Map (EKF update) plot for datasets 03 and 10
3) Visual SLAM output for datasets 03 and 10

A. *IMU Localization - EKF Prediction*

The dead reckoning plots are shown below for dataset 03 and dataset 10. The twist model as mentioned in the technical approach was implemented and the plots were obtained. There were no errors or special techniques encountered in this method and the implementation was straightforward based on the pr3utils file.



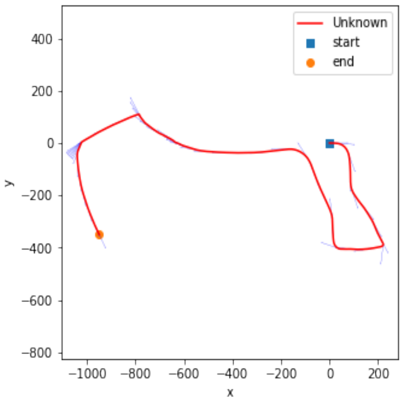


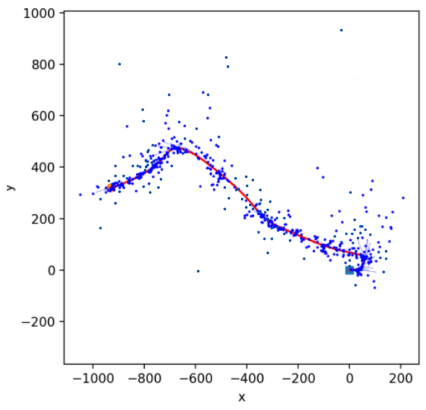
Fig. 2. Dead reckoning for the trajectory of dataset 03 and dataset 10

B. *Mapping - EKF Update*

The process of landmark mapping involves projecting and updating landmark features onto the world frame using dead reckoning. To achieve accurate results, the noise factor V in the Kalman gain equation was tuned for each dataset, with a value of 1 for dataset 03 and 0.5 for dataset 10. Additionally, it was crucial to ensure that all equations had carefully matched matrix dimensions to ensure the successful execution of the code.

To handle any errors that may arise during the computation of the Kalman gain, an error-handling approach was introduced. The code was programmed to ignore cases where a singular matrix error was encountered during the Kalman gain or Ztilda calculation and proceed to the next iteration. It is worth noting that singular matrix errors can arise when a matrix is not invertible, which can occur during Kalman gain or Ztilda calculation when the inverse of a matrix is being computed.

Although fixing noise factors is the preferred method for tackling singular matrix errors, it can be time-consuming to iterate over multiple noise factors. As a result, the error-handling approach was found to be an effective solution. Despite the preference for fixing noise factors, the error-handling approach was used due to its ability to handle errors more efficiently. Overall, the combination of tuning noise factors, matching matrix dimensions, and error handling was crucial in obtaining accurate results during the landmark mapping process.



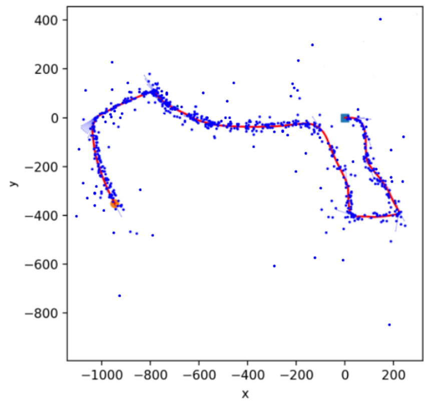


Fig. 3. Update step of dataset 03 and dataset 10

C. *Visual SLAM*

The visual SLAM is implemented as discussed in the pseudocode with appropriate noise settings. the plots are shown below for dataset 03 and dataset 10. Incorrect estimation of noise and covariance matrices is the root cause of the problem with the Kalman filter. The filter's performance heavily relies on the chosen noise values, and even slight variations in these matrices can lead to significant errors in the results. Changes to the W or V matrices, for example, can cause the robot's position estimate to explode or deviate from the intended path, respectively. The W and V matrices used for EKF are given below:

$$W = \begin{bmatrix} 0.01 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.001 \end{bmatrix}$$

$$V = 5 * \text{np.eye } (N_t) \quad (22)$$

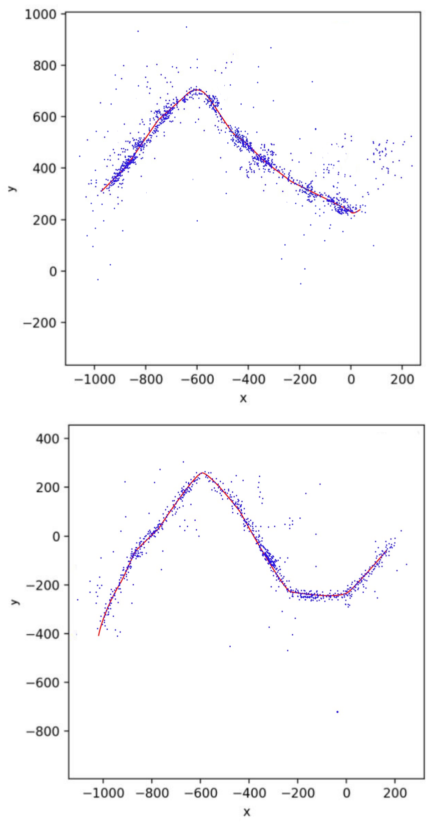


Fig. 4. Visual SLAM dataset 03 and dataset 10

V. Acknowledgement

I collaborated with Mihir Kulkarni (A59018127) and Yeshwant Matey (A59015107) for the assignment where we helped in debugging each other's code.

VI. References

[1] https://natanaso.github.io/