ECE 276A: Sensing & Estimation in Robotics
Project 1: Visual-Inertial SLAM

Name: Raghul Shreeram Sivakumaran

03/22/2024

1 Introduction

The Visual-Inertial Simultaneous Localization and Mapping (VI-SLAM) problem addresses the challenge of estimating the trajectory of a moving agent in an unknown environment while simultaneously constructing a map of that environment using sensory data from both cameras and inertial sensors. This project focuses on implementing a VI-SLAM algorithm using an extended Kalman filter (EKF) to fuse information from both visual and inertial measurements. Accurate localization and mapping are critical for various applications such as robotics, autonomous navigation, augmented reality, and virtual reality. VI-SLAM plays a vital role in enabling robots and other autonomous systems to navigate and interact with their surroundings effectively, especially in dynamic or GPS-denied environments where relying solely on GPS signals is insufficient. By combining data from visual cameras and inertial sensors, VI-SLAM systems can achieve robust localization and mapping capabilities, even in challenging conditions such as low-light environments, featureless areas, or rapid movements.

2 Problem Formulation

Camera Parameters

The camera parameters provided for the visual-inertial SLAM problem include:

1. **Intrinsic Calibration (Camera Matrix K):**

$$K = \begin{bmatrix} f_{su} & 0 & c_u \\ 0 & f_{sv} & c_v \\ 0 & 0 & 1 \end{bmatrix}$$

   where f_{su} and f_{sv} represent the focal lengths along the x and y axes, and c_u and c_v denote the optical center coordinates.

2. **Extrinsic Calibration (Transformation Matrix $T_{\text{IMU-Camera}}$):**

$$T_{\text{IMU-Camera}} = \text{Transformation matrix}$$

   This matrix specifies the translation and rotation components necessary to align the camera frame with the IMU frame.

Parameters in Kalman Filter

- $\mathbf{x}_t$: State vector at time t.
- F: State transition matrix.
- G: Control-input matrix.
- $\mathbf{u}_t$: Control input vector at time t.
- $\mathbf{w}_t$: Process noise vector at time t.

- W: Process noise covariance matrix.
- $\mu_{t|t}$: Prior mean estimate of the state at time t.
- $\Sigma_{t|t}$: Prior covariance matrix of the state at time t.
- $\mu_{t+1|t}$: Predicted mean estimate of the state at time $t+1$ given information up to time t.
- $\Sigma_{t+1|t}$: Predicted covariance matrix of the state at time $t+1$ given information up to time t.
- $\mathbf{z}_t$: Observation or measurement vector at time t.
- H: Observation matrix.
- $\mathbf{v}_t$: Measurement noise vector at time t.
- V: Measurement noise covariance matrix.
- $K_{t+1|t}$: Kalman gain matrix at time $t+1$ given information up to time t.

IMU Prediction using EKF

Motion model:

$$\mathbf{x}_{t+1} = F\mathbf{x}_t + G\mathbf{u}_t + \mathbf{w}_t, \quad \mathbf{w}_t \sim \mathcal{N}(0, W)$$

Prediction:

$$\mu_{t+1|t} = F\mu_{t|t} + G\mathbf{u}_t$$
$$\Sigma_{t+1|t} = F\Sigma_{t|t}F^\top + W$$

Landmark Mapping Prediction and IMU Update using EKF

The prediction step estimates the positions of new landmarks observed in the camera images based on triangulation and camera parameters. The update step integrates new visual feature observations to refine the estimated landmark positions. Observation model:

$$\mathbf{z}_t = H\mathbf{x}_t + \mathbf{v}_t, \quad \mathbf{v}_t \sim \mathcal{N}(0, V)$$

Update:

$$K_{t+1|t} = \Sigma_{t+1|t}H^\top(H\Sigma_{t+1|t}H^\top + V)^{-1}$$
$$\mu_{t+1|t+1} = \mu_{t+1|t} + K_{t+1|t}(\mathbf{z}_{t+1} - H\mu_{t+1|t})$$
$$\Sigma_{t+1|t+1} = (I - K_{t+1|t}H)\Sigma_{t+1|t}$$

The resulted VI SLAM system integrates IMU predictions and camera observations using the Kalman Filter framework to estimate the state (pose and landmarks) of the system over time. This approach allows for accurate localization and mapping in environments where both visual and inertial sensors are available.

3 Technical Approach

3.1 Extended Kalman Filter (EKF) for SLAM

The Extended Kalman Filter (EKF) process in SLAM can be divided into two main steps: the prediction step and the update step. These steps are formulated as follows:

EKF Prediction

The prediction step estimates the robot's state at the next time step $(t+1)$ based on the current state, control inputs, and the process noise.

- Let $F_t := \frac{\partial f}{\partial x}(\mu_{t|t}, u_t, 0)$ and $Q_t := \frac{\partial f}{\partial w}(\mu_{t|t}, u_t, 0)$, where f represents the motion model. This allows us to approximate the motion model as:

$$f(x_t, u_t, w_t) \approx f(\mu_{t|t}, u_t, 0) + F_t(x_t - \mu_{t|t}) + Q_t w_t$$

- The predicted mean ($\mu_{t+1|t}$) is computed as:

$$\mu_{t+1|t} = f(\mu_{t|t}, u_t, 0)$$

- The predicted covariance ($\Sigma_{t+1|t}$) is given by:

$$\Sigma_{t+1|t} = F_t \Sigma_{t|t} F_t^\top + Q_t W Q_t^\top$$

EKF Update

The update step refines the state estimate by incorporating new measurements obtained from the environment.

- Let $H_{t+1} := \frac{\partial h}{\partial x}(\mu_{t+1|t}, 0)$ and $R_{t+1} := \frac{\partial h}{\partial v}(\mu_{t+1|t}, 0)$, where h represents the observation model. The observation model can be linearly approximated around the predicted state:

$$h(x_{t+1}, v_{t+1}) \approx h(\mu_{t+1|t}, 0) + H_{t+1}(x_{t+1} - \mu_{t+1|t}) + R_{t+1} v_{t+1}$$

- The conditional Gaussian distribution of $x_{t+1}|z_{t+1}$ is calculated as:

$$\mu_{t+1|t+1} = \mu_{t+1|t} + K_{t+1|t}(z_{t+1} - h(\mu_{t+1|t}, 0))$$

$$\Sigma_{t+1|t+1} = \Sigma_{t+1|t} - K_{t+1|t} S_{t+1|t} K_{t+1|t}^\top$$

  where $K_{t+1|t} = \Sigma_{t+1|t} H_{t+1}^\top S_{t+1|t}^{-1}$ and $S_{t+1|t} = H_{t+1} \Sigma_{t+1|t} H_{t+1}^\top + R_{t+1} V R_{t+1}^\top$.

These steps describe the EKF process in SLAM, highlighting how the prediction and update mechanisms work together to estimate the robot's trajectory and the environment's map.

3.2 Approach

Implementing an Extended Kalman Filter (EKF) prediction step for IMU localization within the context of SE(3) kinematics involves estimating the pose of an Inertial Measurement Unit (IMU) over time, taking into account its linear and angular velocities. This process is fundamental for applications like robotics and autonomous vehicles, where precise tracking of an object's position and orientation in space is crucial. Let's break down the components and steps involved in technical detail.

State Representation in SE(3)

The pose of the IMU, $T_t \in \mathrm{SE}(3)$, represents its position and orientation in 3D space at time t. The space SE(3) refers to the Special Euclidean group, consisting of rotations and translations in 3D space. An element T_t of SE(3) is typically represented as a 4x4 homogeneous transformation matrix:

$$T_t = \begin{bmatrix} R_t & \mathbf{p}_t \\ \mathbf{0}^\top & 1 \end{bmatrix}$$

where $R_t \in \mathrm{SO}(3)$ is a rotation matrix representing the orientation of the IMU, and $\mathbf{p}_t \in \mathbb{R}^3$ is a vector representing its position in space. The bottom row $\mathbf{0}^\top$ and 1 make T_t a homogeneous matrix, facilitating the representation of rotation and translation in a unified framework.

IMU Kinematics in SE(3)

The motion of the IMU can be characterized by its linear velocity $\mathbf{v}_t \in \mathbb{R}^3$ and angular velocity $\boldsymbol{\omega}_t \in \mathbb{R}^3$. These velocities are typically expressed in the body frame of the IMU. The kinematic equation describing the evolution of T_t over time is:

$$\dot{T}_t = T_t \cdot \hat{\xi}_t$$

where $\hat{\xi}_t$ is the twist vector associated with the IMU's velocity, represented in matrix form as:

$$\hat{\xi}_t = \begin{bmatrix} \hat{\boldsymbol{\omega}}_t & \mathbf{v}_t \\ \mathbf{0}^\top & 0 \end{bmatrix}$$

and $\hat{\boldsymbol{\omega}}_t$ is the skew-symmetric matrix form of the angular velocity vector $\boldsymbol{\omega}_t$, which facilitates the representation of cross products as matrix multiplications.

EKF Prediction Step

The EKF prediction step involves updating the estimated state $T_{t|t-1}$ based on the model of the system's dynamics, in this case, the IMU kinematics in SE(3). Given the current state estimate $T_{t-1|t-1}$ and the measured IMU velocities, the prediction for the next state is:

$$T_{t|t-1} = T_{t-1|t-1} \exp(\hat{\xi}_t \Delta t)$$

where $\exp(\cdot)$ denotes the matrix exponential, transforming the twist vector into an element of SE(3), and Δt is the time step between $t-1$ and t. The covariance of the state is also updated to reflect the uncertainty in the prediction.

This formulation allows the EKF to propagate the state estimate forward in time, accounting for the nonlinearities inherent in the SE(3) space through the use of the Lie group formalism and the matrix exponential.

EKF Update for Landmark Mapping

In our project, we aim to refine the process of estimating the positions of static landmarks using synchronized measurements from an inertial measurement unit (IMU) and a stereo camera. By implementing an Extended Kalman Filter (EKF) update step and judiciously selecting observations from the dataset, we seek to achieve accurate 3D landmark positioning while maintaining computational efficiency.

Data Preparation and Utility Functions

The foundation of our approach is laid by loading essential data, such as IMU linear and angular velocities, stereo camera observations of landmarks, and camera calibration parameters. An integral part of our methodology is the conversion of vectors into skew-symmetric matrices through the `skew_symmetric` function. This is critical for representing angular velocities within the SE(3) motion model as follows:

$$[\mathbf{v}]_\times = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}, \tag{1}$$

where $\mathbf{v} = [v_1, v_2, v_3]^T$ is a vector in $\mathbb{R}^3$.

The `projection_` and `derivative` functions are employed for projecting 3D points into the camera frame and computing the Jacobian of this projection, respectively, which are vital for mapping the world coordinates of landmarks into 2D image coordinates.

Landmark Initialization

We have designed a function for initializing newly observed landmarks by transforming 2D points in image coordinates to 3D world coordinates using camera calibration parameters and the IMU's pose, facilitating the start of the EKF update cycle with an initial estimate of landmark positions.

EKF Update for Landmark Mapping

Identifying Valid Observations

Firstly, we identify landmarks visible in the current frame to concentrate the update step solely on these observations:

$$\text{valid\_observations} = \{\mathbf{z}_t \mid \mathbf{z}_t \neq -1\}. \tag{2}$$

Computation of the H Matrix

The Jacobian matrix H of the observation model is calculated for each landmark position. This matrix relates changes in the observed feature positions in image coordinates to small variations in the 3D positions of the landmarks:

$$H = \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}, \tag{3}$$

where $\mathbf{h}(\mathbf{x})$ represents the observation model mapping the state space to the measurement space.

Updating Landmark Positions and Covariance

Using the EKF equations, the estimated positions of the observed landmarks and their covariance matrix are updated based on the discrepancies between predicted and observed feature positions:

$$\hat{\mathbf{x}}_{t+1} = \hat{\mathbf{x}}_t + K_t(\mathbf{z}_t - H\hat{\mathbf{x}}_t), \tag{4}$$

$$P_{t+1} = (I - K_t H)P_t, \tag{5}$$

where K_t is the Kalman Gain calculated to balance the uncertainty in the state estimate and the measurement noise.

Storing Updated States and Covariances

Finally, the updated landmark positions and their covariances are stored for future iterations, allowing for iterative refinement of landmark positions with each new observation.

By systematically integrating these mathematical formulations and computational steps, our code performs an EKF update for landmark mapping. This approach underpins our visual-inertial SLAM methodology, enabling accurate mapping of the environment and robust tracking of the IMU's pose over time.

3.3 Visual-Inertial SLAM

Mathematical Expressions in Visual-Inertial SLAM Algorithm

Kalman Gain:

The Kalman gain is computed as follows:

$$K = P_{\text{prev}} H^T (H P_{\text{prev}} H^T + R)^{-1}$$

where:

- K is the Kalman gain matrix,
- P_{prev} is the covariance matrix of the state estimate at the previous time step,
- H is the Jacobian matrix of the observation model,
- R is the covariance matrix of the observation noise.

Update Step:

The update step for the IMU pose T_t is given by:

$$T_{\text{upd}} = T_{\text{next}} \exp(\hat{\delta\xi})$$

where:

- T_{upd} is the updated IMU pose,
- T_{next} is the predicted IMU pose for the next time step,
- $\hat{\delta\xi}$ is the correction term obtained from the Kalman gain.

H Matrix:

The H matrix is constructed based on the observation model and the Jacobian matrices:

$$H = \begin{bmatrix} \frac{\partial h}{\partial \xi_1} & \frac{\partial h}{\partial \xi_2} & \cdots & \frac{\partial h}{\partial \xi_n} \\ \frac{\partial h}{\partial \xi_{n+1}} & \frac{\partial h}{\partial \xi_{n+2}} & \cdots & \frac{\partial h}{\partial \xi_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h}{\partial \xi_{(m-1)n+1}} & \frac{\partial h}{\partial \xi_{(m-1)n+2}} & \cdots & \frac{\partial h}{\partial \xi_{mn}} \end{bmatrix}$$

where:

- h is the observation model,

- ξ_i are the elements of the state vector ξ,
- m is the number of observations,
- n is the dimensionality of the state space.

Prediction Step:
The prediction step for the IMU pose T_t involves using the SE(3) kinematics equations:

$$T_{\text{next}} = \exp(\hat{\omega}\Delta t)T_{\text{prev}}$$

where:

- T_{next} is the predicted IMU pose for the next time step,
- $\hat{\omega}$ is the skew-symmetric matrix of the angular velocity,
- Δt is the time step,
- T_{prev} is the IMU pose at the previous time step.

To implement a complete visual-inertial SLAM algorithm, we need to combine the IMU prediction step with the landmark update step and incorporate a stereo-camera observation model to update the IMU pose. Let's break down each component in detail:

IMU Prediction Step:

- The IMU prediction step utilizes the data from the Inertial Measurement Unit (IMU), which typically provides measurements of linear acceleration and angular velocity.
- Using the SE(3) kinematics equations, we can predict the next pose T_t (a rigid body transformation representing the robot's pose) based on the current pose and the IMU measurements.
- The SE(3) kinematics equations describe the motion of the robot in 3D space, incorporating both translational and rotational motion. They are used to propagate the robot's pose forward in time.

Landmark Update Step:

- In the landmark update step, observations from the stereo-camera system are used to update the positions of landmarks in the environment.
- When new features are observed by the camera, the corresponding landmarks are initialized using triangulation, leveraging the camera projection model and known camera parameters.
- Jacobian matrices are computed to linearize the observation model, allowing for the incorporation of camera measurements into the SLAM algorithm.
- Kalman filtering techniques, such as the Extended Kalman Filter (EKF), are used to fuse IMU and camera measurements, updating both the robot's pose and the landmark positions simultaneously.

Update Step for IMU Pose:

- To obtain a complete visual-inertial SLAM algorithm, we need to incorporate an update step for the IMU pose based on the stereo-camera observation model.
- This involves using the camera observations to refine the estimate of the IMU pose T_t.
- The stereo-camera observation model relates the observed features in the camera images to the 3D positions of landmarks in the environment.
- By comparing the observed features with the predicted features based on the current estimate of the robot's pose and the known positions of landmarks, we can compute an update to refine the IMU pose estimate.
- Jacobian matrices are again utilized to linearize the observation model and compute the Kalman gain for fusing camera measurements with the IMU pose estimate.

Fusion of IMU and Camera Measurements:

- The IMU predictions and camera measurements are fused together using a Kalman filter-based approach.
- Kalman filtering techniques enable the integration of noisy sensor measurements over time, providing a robust estimate of the robot's trajectory and the positions of landmarks in the environment.
- Covariance matrices are used to represent the uncertainties associated with both IMU and camera measurements, allowing the Kalman filter to weight the contributions of each sensor appropriately.

By combining these components, we create a complete visual-inertial SLAM algorithm that leverages both IMU and camera measurements to estimate the robot's trajectory and map its environment accurately. This integrated approach improves localization and mapping performance, especially in challenging environments where one sensor modality alone may be insufficient.

Reduction of Computation Complexity:

- **Jacobian Approximation**: Approximating the measurement Jacobian reduces computational complexity compared to exact analytical computation, especially for nonlinear systems.
- **Sparse Updates**: Updating only relevant parts of the state and covariance matrices based on observed landmarks reduces computational overhead.
- **Parallelization and Vectorization**: Leveraging parallel computing and vectorized operations optimizes matrix computations for efficient execution.
- **Optimized Implementation**: Careful implementation and optimization of algorithms ensure computational resources are used judiciously, balancing accuracy and efficiency.

Hyperparameters

- Landmark Covariance: 5×10^{-3}

$$\text{landmark\_covariance} = \begin{bmatrix} 5 \times 10^{-3} & 0 & 0 \\ 0 & 5 \times 10^{-3} & 0 \\ 0 & 0 & 5 \times 10^{-3} \end{bmatrix}$$

- Position Covariance: 1×10^{-3}

$$\text{position\_covariance} = \begin{bmatrix} 1 \times 10^{-3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 \times 10^{-3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 \times 10^{-3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \times 10^{-3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \times 10^{-3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \times 10^{-3} \end{bmatrix}$$

- Observation Noise Covariance: 100

$$\text{observation\_noise\_covariance} = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix}$$

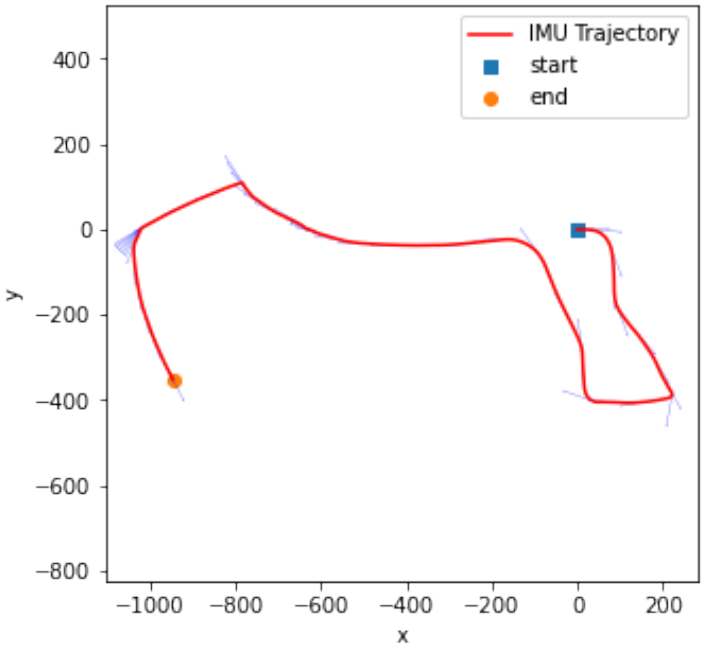
- Process Noise Covariance:

$$\text{process\_noise\_covariance} = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \times 10^{-5} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \times 10^{-5} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \times 10^{-4} \end{bmatrix}$$

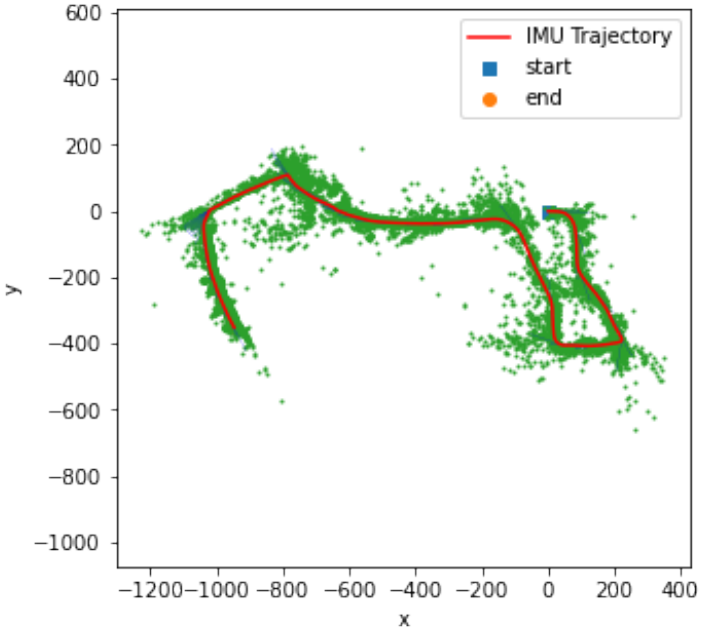
4 Results

4.1 Dataset - 10

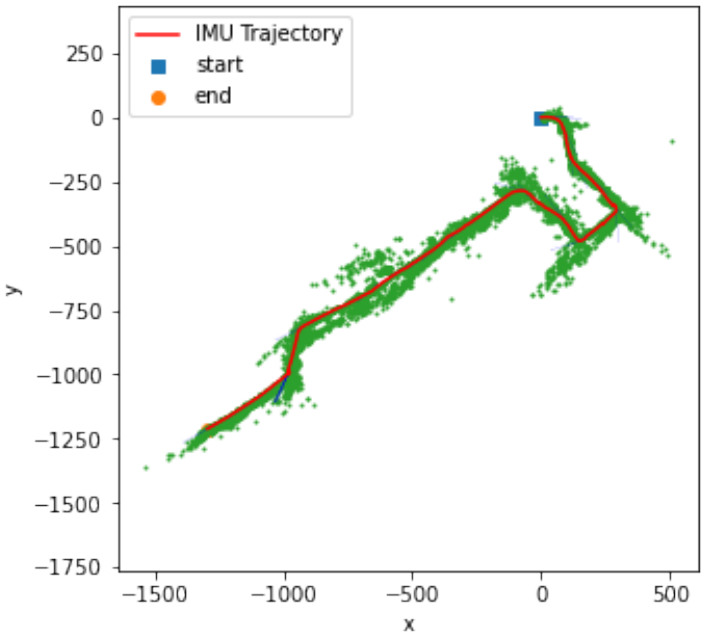
4.1.1 IMU Prediction



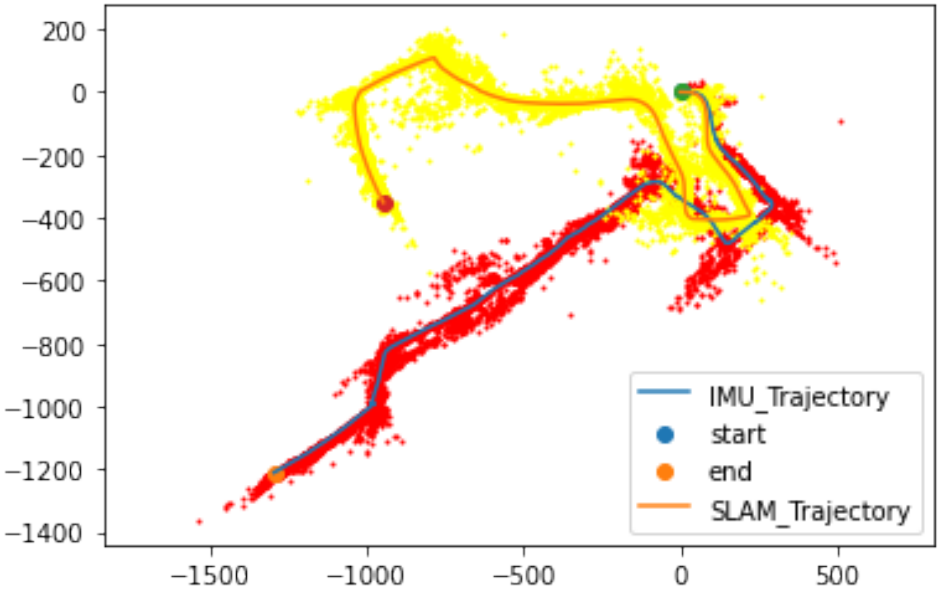
4.1.2 Landmark Mapping



4.1.3 VI SLAM



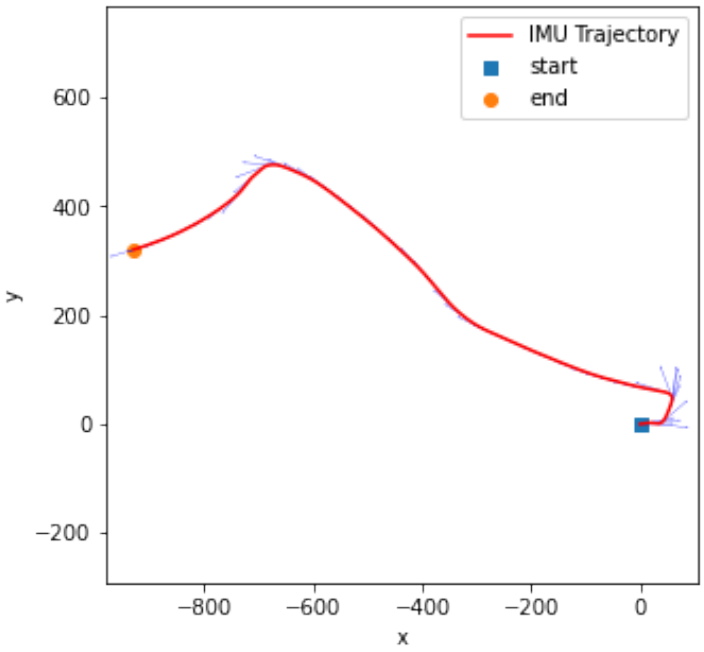
4.1.4 Comparison with predicted Trajectory



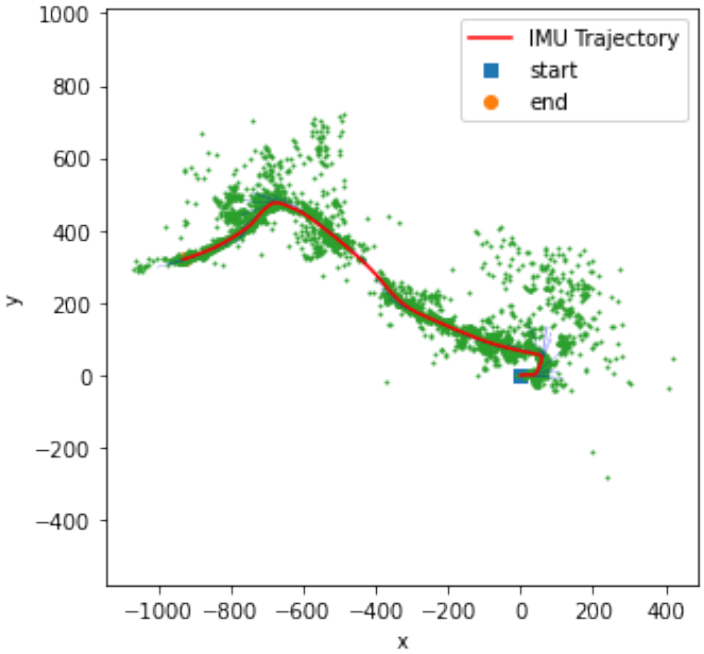
VI SLAM Video Link: **VI SLAM Dataset 10**

4.2 Dataset - 03

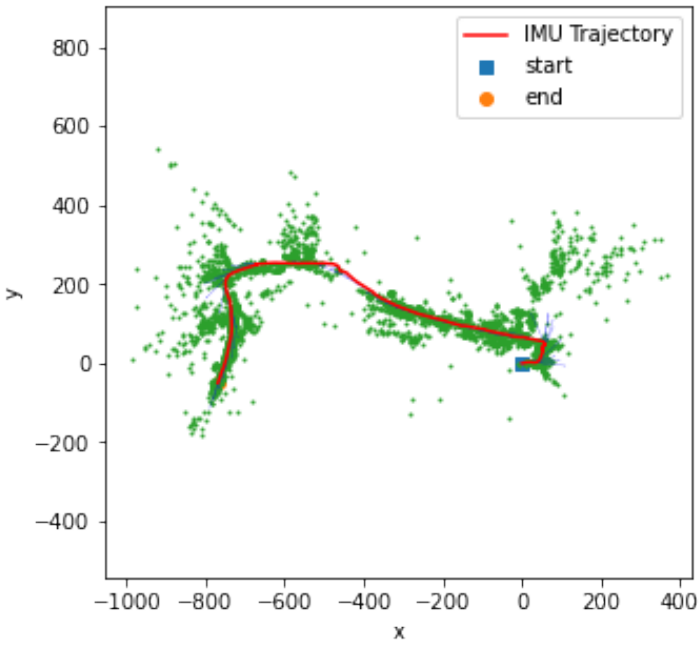
4.2.1 IMU Prediction



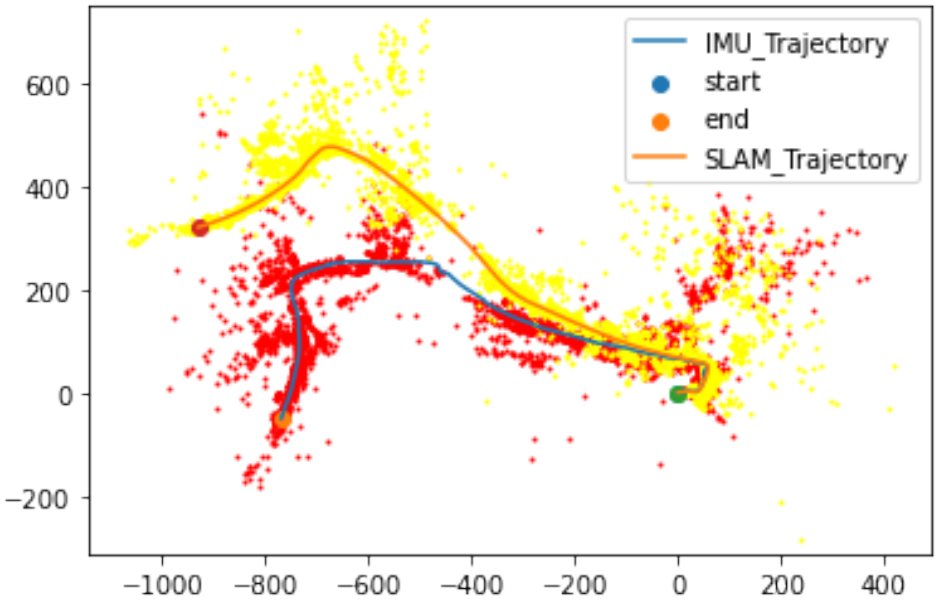
4.2.2 Landmark Mapping



4.2.3 VI SLAM



4.2.4 Comparison with predicted Trajectory



VI SLAM Video Link: **VI SLAM Dataset 3**

Effects of Covariance Parameters

`lc` - Landmark Covariance

This parameter affects the uncertainty associated with the positions of landmarks in the environment.

- A higher value of `lc` indicates higher uncertainty in the landmark positions, while a lower value indicates lower uncertainty.
- Increasing `lc` makes the algorithm less reliant on landmark observations, potentially resulting in smoother trajectories but less accurate landmark position estimates.
- Conversely, decreasing `lc` increases the reliance on landmark observations, potentially resulting in more accurate but noisier trajectory estimates.

`obn` - Observation Noise Covariance

This parameter represents the uncertainty associated with sensor observations.

- A higher value of `obn` indicates higher sensor noise, leading to more uncertainty in observed features (landmarks).
- Increasing `obn` makes the algorithm more tolerant to noisy observations, resulting in smoother trajectories but potentially reduced accuracy.
- Conversely, decreasing `obn` decreases tolerance to noisy observations, potentially leading to more accurate but jagged trajectories.

`pc` - Position Covariance

This parameter represents the uncertainty associated with the robot's pose estimation.

- A higher value of `pc` indicates higher uncertainty in the robot's pose estimation.
- Increasing `pc` makes the algorithm less reliant on IMU predictions, potentially resulting in smoother trajectories but less accurate pose estimates.
- Conversely, decreasing `pc` increases reliance on IMU predictions, potentially resulting in more accurate but noisier trajectory estimates.

`process_noise_cov`

This parameter represents uncertainty associated with the process model used for predicting the robot's motion.

- It affects how much trust the algorithm places in IMU predictions versus observed landmarks.
- A higher value of `process_noise_cov` indicates higher uncertainty in IMU predictions, leading to more reliance on observed landmarks.
- Increasing `process_noise_cov` makes the algorithm more sensitive to observed landmarks, potentially resulting in smoother trajectories but reduced accuracy in pose estimation.
- Conversely, decreasing `process_noise_cov` increases reliance on IMU predictions, potentially resulting in more accurate but jagged trajectories.