ECE 276A Project 3: Visual-Inertial SLAM

Jean Gorby Calicdan
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, United States
jcalicda@ucsd.edu

***Abstract*—This project explores Visual-Inertial Simultaneous Localization and Mapping (VI-SLAM) utilizing an Extended Kalman Filter (EKF) framework. VI-SLAM integrates inertial measurement unit (IMU) data, capturing linear and angular velocities, with stereo-camera observations to accurately estimate the trajectory of a robot and map environmental landmarks in real time. Using measurements collected by a Clearpath Jackal robot navigating MIT's campus, this implementation addresses several critical components: IMU-based localization through SE(3) kinematics, landmark initialization and mapping via stereo triangulation, and feature detection and matching using optical flow techniques. The approach carefully manages computational complexity by dynamically selecting robust features and filtering unreliable observations. Through iterative EKF updates, the system refines both landmark positions and robot poses, effectively reducing localization drift inherent in pure inertial odometry. Experimental results demonstrate precise trajectory estimation and robust landmark mapping, highlighting the effectiveness of integrating visual and inertial sensors for enhanced navigation in robotic applications.**

***Index Terms*—Visual-Inertial SLAM (VI-SLAM) Extended Kalman Filter (EKF) Inertial Measurement Unit (IMU) Stereo-camera SE(3) kinematics**

I. Introduction

Visual-Inertial Simultaneous Localization and Mapping (VI-SLAM) is a critical capability in robotics that enables autonomous navigation by simultaneously determining a robot's location and constructing a map of the surrounding environment. This project investigates the problem of accurately estimating both the robot trajectory and landmark positions through the integration of visual data from stereo cameras and inertial measurements from an IMU sensor. The challenges addressed include localization using inertial data, landmark mapping from visual observations, and combining these modalities for improved accuracy and robustness.

The localization problem involves predicting the robot's trajectory based solely on inertial sensor measurements, specifically linear and angular velocities. The approach for this utilizes a kinematic model within an Extended Kalman Filter (EKF) framework to systematically integrate inertial measurements and continuously predict the robot's pose.

For landmark mapping, the task is to estimate precise landmark positions from visual observations provided by a stereo camera setup. The method employed here is to initialize landmark positions via stereo triangulation, followed by continuous refinement through subsequent visual observations, again using the EKF update mechanism.

Finally, the visual-inertial SLAM challenge is addressed by integrating both the IMU-based localization and visual landmark mapping components. The system leverages stereo-camera observations to correct the drift inherent in purely inertial localization, dynamically selecting and tracking robust visual features through optical flow techniques. Through iterative EKF updates, the combined visual-inertial method significantly enhances trajectory estimation accuracy and landmark mapping reliability. The effectiveness of this integrated approach is demonstrated using experimental data collected by a Clearpath Jackal robot navigating the MIT campus.

II. Problem Formulation

A. IMU Localization via EKF Prediction

I consider the problem of estimating the pose of a mobile robot over time using measurements from an inertial measurement unit (IMU). At each time step t, the IMU provides linear velocity $v_t \in \mathbb{R}^3$ and angular velocity $\omega_t \in \mathbb{R}^3$, expressed in the robot's body frame. My objective is to estimate the robot's pose $T_t \in SE(3)$ using an Extended Kalman Filter (EKF) prediction step.

The general EKF prediction update for a nonlinear system with state x, control input u, and process noise covariance W is given by:

$$\begin{aligned}\mu_{t+1|t} &= f(\mu_{t|t}, u_t),\\ \Sigma_{t+1|t} &= F_t \Sigma_{t|t} F_t^\top + Q_t W Q_t^\top,\end{aligned}$$

where:

- f is the motion model,
- $F_t = \frac{\partial f}{\partial x}$ is the Jacobian of the motion model with respect to the state,
- $Q_t = \frac{\partial f}{\partial w}$ is the Jacobian with respect to noise.

In the context of SE(3), this framework is applied using the exponential map to evolve the pose, and the adjoint operator to propagate uncertainty in the tangent space of the manifold.

B. Feature detection and matching

The problem of feature detection and matching is to identify and track distinctive image locations across stereo pairs and over time in order to support reliable landmark association in SLAM. The inputs to this problem are sequences of stereo grayscale images collected at discrete time steps. The desired output is a measurement matrix $z_t \in \mathbb{R}^{4\times M}$ for each time step t, where M is the total number of feature tracks. Each column of z_t corresponds to a single feature and contains the pixel coordinates in the left and right images:

$$z_t = \begin{bmatrix} l_{x,1} & l_{x,2} & \cdots & l_{x,M} \\ l_{y,1} & l_{y,2} & \cdots & l_{y,M} \\ r_{x,1} & r_{x,2} & \cdots & r_{x,M} \\ r_{y,1} & r_{y,2} & \cdots & r_{y,M} \end{bmatrix},$$

where $(l_{x,j}, l_{y,j})$ and $(r_{x,j}, r_{y,j})$ represent the image coordinates of feature j in the left and right cameras, respectively.

The formulation of this problem relies on selecting features that are well-conditioned for tracking, i.e., points with strong intensity gradients in orthogonal directions. This is typically characterized using the gradient matrix $G(z)$ over a local window $W(z)$:

$$G(z) = \sum_{y \in W(z)} \begin{bmatrix} I_u(y)^2 & I_u(y)I_v(y) \\ I_u(y)I_v(y) & I_v(y)^2 \end{bmatrix},$$

$$\text{valid feature if } \lambda_{\min}(G(z)) > \rho.$$

Here, I_u and I_v are spatial image gradients and ρ is a threshold determining whether the pixel exhibits sufficient structure.

The temporal evolution of each feature is modeled under a local translational motion assumption, based on the brightness constancy constraint. For a pixel location z, I seek the motion vector $\nu \in \mathbb{R}^2$ that minimizes:

$$\sum_{y \in W(z)} \left\| \nabla I(y,t)^\top \nu + I_t(y,t) \right\|^2.$$

This formulation applies to both stereo matching (left-to-right correspondences) and temporal tracking (frame-to-frame motion), and together they produce a measurement matrix z_t for each time step. If a feature is not observed, the corresponding column in z_t is filled with $[-1,-1,-1,-1]^\top$. The resulting set of measurements provides the necessary data association structure to support landmark triangulation and state estimation in the SLAM framework.

C. Landmark mapping via EKF update

The problem of landmark mapping is to estimate the 3D positions of static point landmarks observed by a stereo camera, using known IMU poses and visual feature measurements over time. The state to be estimated is a stacked vector $m \in \mathbb{R}^{3M}$, representing the coordinates of M landmarks in the world frame:

$$m = \begin{bmatrix} m_1^\top & m_2^\top & \cdots & m_M^\top \end{bmatrix}^\top, \quad m_j \in \mathbb{R}^3.$$

The inputs to the problem include:

- Known IMU poses $T_t \in SE(3)$ for each time step t.
- Stereo visual observations $z_t \in \mathbb{R}^{4N_t}$, where each $z_{t,i} \in \mathbb{R}^4$ is the pixel location of a feature in both left and right images.
- The stereo camera calibration matrix $K_s \in \mathbb{R}^{4\times4}$ and the extrinsic transformation ${}_OT_I \in SE(3)$ from IMU to camera frame.
- Known data association function Δ_t mapping observed measurements $z_{t,i}$ to landmark indices.

The observation model maps a world landmark m_j to a 4D stereo pixel measurement via the projection function:

$$z_{t,i} = K_s \pi({}_OT_I T_t^{-1} m_j) + v_{t,i}, \quad v_{t,i} \sim \mathcal{N}(0, V),$$

where $\pi(\cdot)$ denotes the perspective projection function and $v_{t,i}$ is zero-mean Gaussian noise.

My objective is to maintain a Gaussian belief over the full landmark state vector, and update it recursively via an Extended Kalman Filter (EKF).

Given the nonlinear measurement function $h(m,v)$, the EKF update is given by the standard equations:

$$\begin{aligned} H_{t+1} &= \left.\frac{\partial h}{\partial m}\right|_{\mu_t}, \\ R_{t+1} &= \left.\frac{\partial h}{\partial m}\right|_{v_t}, \\ K_{t+1} &= \Sigma_t H_{t+1}^\top (H_{t+1}\Sigma_t H_{t+1}^\top + R)^{-1}, \\ \mu_{t+1} &= \mu_t + K_{t+1}(z_{t+1} - \hat{z}_{t+1}), \\ \Sigma_{t+1} &= (I - K_{t+1}H_{t+1})\Sigma_t, \end{aligned}$$

where μ_t, Σ_t are the current landmark mean and covariance.

This problem formulation assumes static landmarks and known correspondences, and seeks to refine their positions over time as new visual measurements become available.

D. Visual-inertial SLAM

The Visual-Inertial SLAM problem involves jointly estimating the trajectory of a moving robot and the positions of observed landmarks, using measurements from an inertial measurement unit (IMU) and a stereo camera system. The state consists of the IMU pose $T_t \in SE(3)$ at each time step t, and the 3D coordinates of M static landmarks $m \in \mathbb{R}^{3M}$:

$$x_t = \{T_t, m_1, \ldots, m_M\}.$$

The inputs to the system are:

- IMU measurements $u_t = [v_t^\top, \omega_t^\top]^\top \in \mathbb{R}^6$, representing linear and angular velocity in the body frame.
- Stereo visual observations $z_t \in \mathbb{R}^{4N_t}$, where each $z_{t,i} \in \mathbb{R}^4$ contains left and right pixel coordinates of feature i at time t.
- Known camera intrinsics K_s and extrinsics ${}^OT_I \in SE(3)$.

The motion model evolves the pose using SE(3) kinematics:

$$T_{t+1} = T_t \exp(\tau_t \hat{u}_t), \quad \hat{u}_t = \begin{bmatrix} \hat{\omega}_t & v_t \\ 0^\top & 0 \end{bmatrix},$$

where τ_t is the time interval and exp denotes the matrix exponential.

The observation model relates a known landmark m_j to a measurement in the stereo image:

$$z_{t,i} = K_s \pi\left({}_OT_I T_t^{-1} m_j\right) + v_{t,i}, \quad v_{t,i} \sim \mathcal{N}(0, V).$$

The outputs are the estimated IMU trajectory $\{T_t\}_{t=0}^T$ and the landmark positions $\{m_j\}_{j=1}^M$, obtained by recursively applying EKF prediction and update steps.

III. Technical Approach

A. IMU Localization via EKF Prediction

My implementation of IMU localization uses the kinematic model of rigid body motion in $SE(3)$, along with an Extended Kalman Filter (EKF) to propagate both the pose estimate and its uncertainty over time. The procedure operates entirely in the Lie group $SE(3)$ and its associated Lie algebra $\mathfrak{se}(3)$, leveraging exponential and adjoint mappings to handle the non-Euclidean nature of pose composition and uncertainty.

The prediction is implemented in the function `imu_ekf_prediction`, and follows these key steps:

1) Initialization: The initial IMU pose is set to the identity matrix:

$$\mu_0 = I_{4\times 4},$$

and the initial covariance is chosen to be a small diagonal matrix:

$$\Sigma_0 = 10^{-3} \cdot I_6,$$

unless explicitly specified. This reflects high confidence in the starting pose.

2) Input Formatting: At each time step t, the IMU provides a 6D twist vector:

$$u_t = \begin{bmatrix} v_t \\ \omega_t \end{bmatrix} \in \mathbb{R}^6,$$

where v_t and ω_t are linear and angular velocities in the body frame. These are converted into their corresponding representations in the Lie algebra:

- $\hat{u}_t = \texttt{axangle2twist}(u_t) \in \mathfrak{se}(3)$ — a twist matrix used for propagating the pose,
- $\overset{\curlywedge}{u}_t = \texttt{pose2adpose}(\hat{u}_t) \in \mathbb{R}^{6\times 6}$ — the adjoint operator, used to propagate covariance.

3) EKF Prediction Step with $SE(3)$ Kinematics: The EKF prediction step is applied in the context of $SE(3)$, where the state is a transformation matrix $T_t \in SE(3)$ and the uncertainty is represented in its tangent space $\mathfrak{se}(3)$. Given the control input $u_t = [v_t^\top, \omega_t^\top]^\top \in \mathbb{R}^6$ and the time interval τ_t, the EKF prediction step consists of the following:

Nominal Pose Propagation:

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

where $\exp(\cdot)$ is the matrix exponential map from $\mathfrak{se}(3)$ to $SE(3)$.

Covariance Propagation:

$$\Sigma_{t+1|t} = \exp(\tau_t \overset{\curlywedge}{u}_t)\Sigma_{t|t}\exp(\tau_t \overset{\curlywedge}{u}_t)^\top + \tau_t W,$$

where $\overset{\curlywedge}{u}_t \in \mathbb{R}^{6\times 6}$ is the adjoint representation of the twist, and W is the process noise covariance.

B. Feature Detection and Matching

The implementation of feature detection and matching combines Shi-Tomasi corner detection with Lucas-Kanade optical flow to identify and track distinctive image features across stereo image pairs and temporal frames. The output is a measurement array $z_t \in \mathbb{R}^{4\times M}$ for each time step t, with each column representing a feature's pixel coordinates in both the left and right images.

1) Initial Feature Detection: On the first frame, features are detected in the left image using the Shi-Tomasi corner detector via OpenCV's `cv2.goodFeaturesToTrack` function. This method is based on the minimum eigenvalue criterion applied to the structure tensor:

$$G(z) = \sum_{y\in W(z)} \begin{bmatrix} I_u(y)^2 & I_u(y)I_v(y) \\ I_u(y)I_v(y) & I_v(y)^2 \end{bmatrix},$$

where $W(z)$ is a local window around pixel z, and I_u, I_v are the image gradients in the horizontal and vertical directions. A pixel is considered a valid corner if:

$$\lambda_{\min}(G(z)) > \rho.$$

2) Stereo Matching and Temporal Tracking: Feature correspondences across stereo image pairs (left to right) and over time (left to left) are computed using OpenCV's pyramidal Lucas-Kanade optical flow algorithm, `cv2.calcOpticalFlowPyrLK`. This is based on the brightness constancy assumption and a local translational motion model:

$$I(y,t) \approx I(y+\nu, t+1),$$

leading to the optimization:

$$\nu^* = \arg\min_{\nu} \sum_{y\in W(z)} \left\| \nabla I(y)^\top \nu + I_t(y) \right\|^2,$$

with closed-form solution:

$$\nu^* = -G(z)^{-1}b(z), \quad \text{where } b(z) = \sum_{y\in W(z)} \begin{bmatrix} I_u(y)I_t(y) \\ I_v(y)I_t(y) \end{bmatrix}.$$

If the status is true, the updated feature location is accepted; otherwise, the measurement is set to $[-1,-1,-1,-1]$.

3) Dynamic Re-Detection and Feature Management: To ensure persistent tracking, the system monitors the number of active features. When this number drops below a threshold (e.g., 100), new features are detected in the current left image. Detections too close to existing features are discarded based on a minimum Euclidean distance threshold (10 pixels). New features are then matched to the right image to initialize their stereo coordinates.

4) Measurement Matrix Construction: All observations are compiled into a measurement matrix of shape $z_t \in \mathbb{R}^{4\times M}$, where each column represents:

$$z_{t,i} = [l_{x,i}, l_{y,i}, r_{x,i}, r_{y,i}]^\top.$$

Unobserved or dropped features are recorded with placeholder values $[-1,-1,-1,-1]^\top$.

Algorithm: Feature Tracking and Stereo Matching:

Algorithm 1 Stereo Feature Tracking and Matching

```
Require: Left/right image sequences $I_L^t, I_R^t$, number of frames $N$
1: Detect Shi-Tomasi corners on $I_L^0$ using cv2.goodFeaturesToTrack
2: Perform stereo matching to $I_R^0$ using cv2.calcOpticalFlowPyrLK
3: for $t = 1$ to $N - 1$ do
4:     Track active features to $I_L^t$ using cv2.calcOpticalFlowPyrLK
5:     Perform stereo matching to $I_R^t$
6:     Remove failed tracks; update measurements
7:     if # Active features $<$ threshold then
8:         Detect new Shi-Tomasi features
9:         Filter out close detections
10:        Match to $I_R^t$ and initialize
11:    end if
12: end for
13: Return measurement matrix $z \in \mathbb{R}^{4\times M\times N}$
```

A video is then generated to show how the features are overlayed on top of the images over time and this video can be found in the results folder of the project submission.

C. Landmark mapping via EKF update

In my implementation, the landmark state is a stacked vector:

$$m = \begin{bmatrix} m_1^\top & m_2^\top & \dots & m_M^\top \end{bmatrix}^\top, \quad m_j \in \mathbb{R}^3.$$

Each landmark is initialized using stereo triangulation at the first time it is observed. To find the first valid observation for each landmark, I iterate through the stereo feature matrix and identify the earliest time index where each feature has nonnegative pixel coordinates. The corresponding left and right pixel locations are then extracted for triangulation.

I use OpenCV's `cv2.triangulatePoints()` to estimate 3D landmark positions in the left camera frame. The left and right projection matrices are constructed using the intrinsic matrices and a found stereo baseline along the x-axis:

$$P_L = K_L \cdot [I \mid 0], \quad P_R = K_R \cdot [I \mid t_{stereo}],$$

where $t_{stereo} = [-b, 0, 0]^\top$ represents the baseline displacement from the left to the right camera in the optical frame and the notation $[R|p]$ represents a pose with orientation P and translation p

The resulting 3D points from triangulation are expressed in homogeneous coordinates and converted to Cartesian form by dividing by the fourth entry:

$$m_j^{(cam)} = \frac{1}{X_4} \begin{bmatrix} X_1 & X_2 & X_3 \end{bmatrix}^\top.$$

To convert these landmarks into the world frame, I apply the inverse transformations of the left camera and the regular-to-optical frame alignment. Each 3D point is transformed using:

$$m_j^{(world)} = T_t \cdot ({}_rT_I)^{-1} \cdot ({}_oT_r)^{-1} \cdot m_j^{(cam)},$$

where T_{t_j} is the IMU pose at the first observation of landmark j, and T_{opt} is the rotation matrix aligning the ROS frame convention to the optical frame convention:

$${}_oT_r = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

After transforming all landmarks into the world frame, I perform geometric filtering to reject spurious landmarks. I compute the minimum Euclidean distance between each landmark and the IMU trajectory. Landmarks whose closest distance to any IMU pose exceeds a predefined threshold are considered outliers and discarded:

$$\text{keep if } \min_t \|m_j - p_t^{IMU}\| \le d_{max}.$$

Only the filtered landmarks are retained for further EKF updates, and an index mapping is maintained to convert original feature indices to valid landmark indices.

Once initialized and filtered, each landmark is refined using EKF updates. For each observed landmark, we:

1) Predict the measurement:

$$\hat{z}_{t,i} = K_s \pi({}_oT_r \cdot_r T_I \cdot (T_t)^{-1} m_j),$$

where $\pi(\cdot)$ is the stereo projection model.

2) Compute the Jacobian of $\tilde{z}_{t+1,i}$ with respect to m_j evaluated at $\mu_{t,j}$:

$$H_{t+1,i,j} = \begin{cases} K_s \frac{d\pi}{dq} \left({}_oT_I T_{t+1}^{-1} \mu_{t,j}\right) ({}_oT_I) T_{t+1}^{-1} P^\top, & \text{if } \Delta_t(j) = i, \\ 0, & \text{otherwise} \end{cases}$$

3) Update the landmark state via EKF:

$$\begin{aligned} K_{t+1} &= \Sigma_t H_{t+1}^\top \left(H_{t+1} \Sigma_t H_{t+1}^\top + I \otimes V\right)^{-1}, \\ \mu_{t+1} &= \mu_t + K_{t+1} \left(z_{t+1} - \tilde{z}_{t+1}\right), \\ \Sigma_{t+1} &= \left(I - K_{t+1} H_{t+1}\right) \Sigma_t \end{aligned}$$

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

To help with the run-time and numerical stability of the code, library packages such as `csr_matrix`, `csc_matrix`, and `lil_matrix` were use to initialize the sparse covariance, jacobian, and kalman gain matrices. For run-time specifically, I use `scipy.sparse.linalg.spsolve`, which solves the linear system of equation $Ax = b$. When I set $b = I$, then I are effectively solving for $x = A^{-1} * I = A^{-1}$. This is helpful when finding the inverse of large and sparse matrices.

D. Visual-Inertial SLAM

My implementation of visual-inertial simultaneous localization and mapping (SLAM) combines inertial prediction and visual update using an Extended Kalman Filter (EKF) framework. The state includes the IMU pose $T_t \in SE(3)$ and the landmark positions $m_j \in \mathbb{R}^3$. At each time step, I perform an EKF prediction step based on SE(3) kinematics and an EKF update step using stereo visual observations.

1) State Initialization: I initialize the full system state as a joint Gaussian over the IMU pose and the 3D landmark positions. The mean μ_0 consists of the initial IMU pose and the triangulated landmark estimates:

$$\mu_0 = \begin{bmatrix} \mu_0^{\text{IMU}} \\ \mu_0^{\text{landmarks}} \end{bmatrix},$$

where $\mu_0^{\text{IMU}} = I_{4\times 4}$ and $\mu_0^{\text{landmarks}} \in \mathbb{R}^{3M}$ is computed via stereo triangulation and transformed into the world frame.

$$\Sigma_0 = \begin{bmatrix} \Sigma_0^{\text{IMU}} & 0 \\ 0 & \Sigma_0^{\text{landmarks}} \end{bmatrix}$$

$$\Sigma_0^{\text{IMU}} \in \mathbb{R}^{6\times 6}, \quad \Sigma_0^{\text{landmarks}} = \sigma^2 I_{3M}$$

2) Prediction Step: IMU Kinematics on $SE(3)$: The IMU pose evolves according to a right-invariant kinematic model:

$$\begin{aligned} \mu_{t+1|t} &= \mu_{t|t} \cdot \exp(\tau_t \hat{u}_t), \\ \Sigma_{t+1|t} &= \exp(\tau_t \overset{\curlywedge}{u}_t)\Sigma_{t|t} \exp(\tau_t \overset{\curlywedge}{u}_t)^\top + \tau_t W, \end{aligned}$$

where $\mu_t \in SE(3)$ is the mean pose, $\Sigma_t \in \mathbb{R}^{(6+3M)\times(6+3M)}$ is the joint covariance, $\hat{u}_t$ is the twist matrix from linear and angular velocity measurements, and $\overset{\curlywedge}{u}_t$ is the adjoint representation used for covariance propagation.

3) Update Step: Joint State Correction: At each time step, I apply the following sequence of operations, closely aligned with the structured EKF update strategy:

1) Compute the Jacobian with respect to the IMU pose ($4N \times 6$):

$$H_{t+1,i}^{\text{pose}} = -K_s \frac{d\pi}{d\mathbf{q}} \left({}_oT_I \mu_{t+1|t}^{-1} \mathbf{m}_j \right) ({}_oT_I) \left(\mu_{t+1|t}^{-1} \mathbf{m}_j \right)^{\circ}$$

2) Compute the Jacobian with respect to each landmark ($4N \times 3M$):

$$H_{t+1,i,j} = \begin{cases} K_s \dfrac{d\pi}{dq} \left({}_oT_I \mu_{t+1|t}^{-1} \mu_{t,j} \right) ({}_oT_I) \mu_{t+1|t}^{-1} P^\top, \\ \text{if } \Delta_t(j) = i, \\ 0, \\ \text{otherwise} \end{cases}$$

3) Concatenate the trajectory and landmark Jacobians to form a $4N \times (6+3M)$ Jacobian matrix H_t.
4) Compute the Kalman gain:

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

5) Update the landmark mean:

$$\mu_{t+1}^{\text{landmarks}} = \mu_t^{\text{landmarks}} + K_{t+1}^{\text{lm}} (z_{t+1} - \hat{z}_{t+1})$$

6) Update the trajectory mean using the Lie algebra:

$$\mu_{t+1|t+1}^{\text{IMU}} = \mu_{t+1|t} \cdot \exp\left(K_{t+1}^{\text{pose}} (z_{t+1} - \hat{z}_{t+1})^\wedge \right)$$

7) Update the joint covariance:

$$\Sigma_{t+1|t+1} = (I - K_{t+1} H_t) \Sigma_t$$

4) Numerical Stabilization and Filtering: To improve numerical stability:

- A regularization term $1e{-}6 \cdot I$ is added to the innovation covariance.
- Updates are skipped if the innovation is too large or the matrix inversion fails.
- Landmarks are filtered based on minimum distance to the IMU trajectory.

IV. Discussion

The project involved iterative development, debugging, and refinement to achieve the final results. Several challenges were encountered and resolved during the process:

A. IMU Localization via EKF Prediction

This section of the project was probably the easiest out of all of them. The hardest part about this section was really understanding how the pose kinematics work with full understanding of the math behind it, particularly the involvement of Lie Algebra. Despite the ease in simply implementing equations given in the slides, understanding how the equations work and how perturbations allow for these equations to be generated is equally, if not more important.

The only other part of this section which could have been prone to error was the noise initialization. I ended up with the following initialization of the noise covariance:

$$W = \text{diag}(0.01,\ 0.01,\ 0.01,\ 0.001,\ 0.001,\ 0.001)$$

I chose this specific initialization to reflect the expected magnitude of noise in the IMU's velocity and angular rate measurements. Translational motion is expected to vary more significantly than rotational drift, hence the larger diagonal values for the linear velocity components. This initialization was kept constant throughout the project and may have been a cause of issues later down the line but often, noise issues were tweaked by adjusting the covariance initialization for the landmarks or the variable V.

B. Feature Detection and Matching

The biggest struggle relating to feature detection and matching was honestly understanding what the multiple `opencv` functions do as well as the math behind them. It was simple enough to understand the high level overview of what each function did, especially with the help of the lecture slides. However, gaining a deep understand of what features are and how they are found with gradients needed its own time.

One of the early struggles with this section had to do with the algorithm that I was implementing. Implementing the Shi-Tomasi corner detector (using `goodFeaturesToTrack`) and Lucas–Kanade optical flow (with `cv2.calcOpticalFlowPyrLK`) worked reasonably well for both stereo matching (left to right) and temporal tracking (left to left). However, I soon noticed that I was basically only looking for good features in the first frame and this is what I were tracking over time. This meant that in later frames, all these features would disappear and thus, there would be no more features to be tracked. This led to a dynamic re-detection approach, where I periodically detect new features if the active set of tracked points falls below a threshold. That approach introduced its own complications—especially ensuring that newly detected features did not duplicate points already being tracked, and dealing with 0-dimensional arrays whenever only a single feature was tracked.

C. Landmark mapping via EKF update

Implementing the EKF update for landmark mapping posed several challenges, primarily stemming from frame inconsistencies and subtle implementation bugs. Initially, I observed that the landmark positions μ remained unchanged after the EKF update step. Upon investigation, I discovered that although the Kalman gain and the innovation were being computed, the update was incorrectly applied. The full state correction vector ($K_t \cdot$ innovation) was being reshaped into a matrix corresponding to all landmarks, but only a subset of those landmarks were observed at each timestep. Consequently, the relevant corrections were not being applied properly. This was resolved by reshaping the correction only for the observed landmarks and updating μ using those entries.

Another early issue was related to object references in Python. I attempted to verify updates by computing the norm between μ and μ_{orig}, expecting a non-zero value. However, this consistently returned zero. It turned out that $\mu_{\text{orig}} = \mu$ created a reference copy, not a deep copy, meaning both variables pointed to the same array. Using `mu.copy()` corrected this and allowed us to properly track changes to the state.

A more fundamental issue arose from the coordinate frames used in projection and landmark initialization. Initially, landmarks were triangulated and stored in the regular camera frame, while the EKF update step assumed they were in the optical frame, leading to a significant increase in reprojection error. This inconsistency stemmed from the fact that the extrinsic matrix `extL_T_imu` provided in the dataset represented the transformation from the IMU frame to the regular camera frame. However, projection equations and the stereo calibration matrix K_s operate in the optical camera frame, where x points right, y down, and z forward. To resolve this, I introduced an explicit rotation matrix $R_{\text{optical}\leftarrow\text{regular}}$ and applied it consistently both during initialization and the EKF update. This ensured all 3D points and projection operations were represented in the correct frame.

Additionally, the baseline between the left and right cameras, required for constructing K_s, was initially computed in the regular frame. This was inaccurate since the baseline must be aligned with the optical frame's x-axis. I corrected this by rotating the translation vector between the cameras into the optical frame before computing the baseline magnitude.

Once these transformations were applied consistently, the reprojection error dropped significantly, confirming the correctness of the update pipeline. Final projections and Jacobian calculations followed the structure described in the lecture notes: $z = K_s\pi({}_oT_I \cdot T_t^{-1} \cdot m)$ with Jacobian $H = K_s \cdot \frac{d\pi}{dq} \cdot {}_oT_I \cdot T_t^{-1} \cdot P^\top$. The use of projection checks throughout the pipeline helped validate each step, and the reprojection error served as a critical metric for catching inconsistencies.

D. Visual-Inertial SLAM

My final SLAM implementation is not perfect. As can be seen in the results, the final updated trajectory are very noisy and makes very sudden jumps. I believe this can be attributed to either poor noise initialization or incorrect implementation. Some of the noise initialization I tried included $V = (1, 10, 100, 1000, 10000, 100000)$. I find that the higher the number, the less intense the jumps become. Another set of initialization I tried pertained to the covariance matrix for the landmarks. I tried $\sigma^2 = (0.0001, 0.001, 0.1, 1)$. I found that there was a fair variance in the performance between these numbers and there wasn't an obvious trend as there was with V. What I didn't try, and something that I should have if I had the time, would be to try some sort of grid search approach between these two variables and maybe even the IMU initialization. I never fully experimented with doing a mix of these and rather focused only on updating one at a time, typically the V.

Related to this, I had the problem of many singular matrices occurring throughout my iterations. One of the fixes I found to help with this was to increase V. However, run-time becomes an issue. For this reason, my final SLAM results for dataset02 with my own generated features was only ran for the first 500 timestamps. The same trend can be seen within these first set of timestamps as with the overall trend seen in the previous examples.

V. Results

A. IMU Localization via EKF Prediction

See next page

B. Feature Detection and Matching

See next page

C. Landmark mapping via EKF update

See next page

D. Visual-Inertial SLAM

See next page

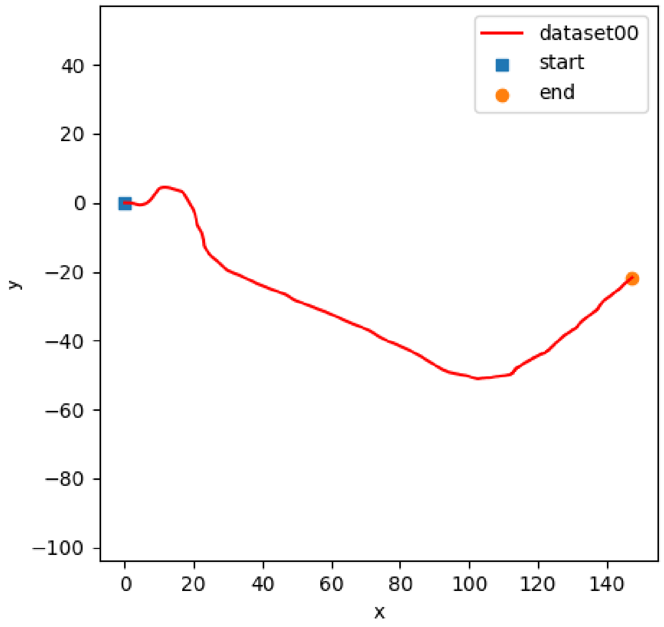


Fig. 1. IMU Localization for Dataset 00

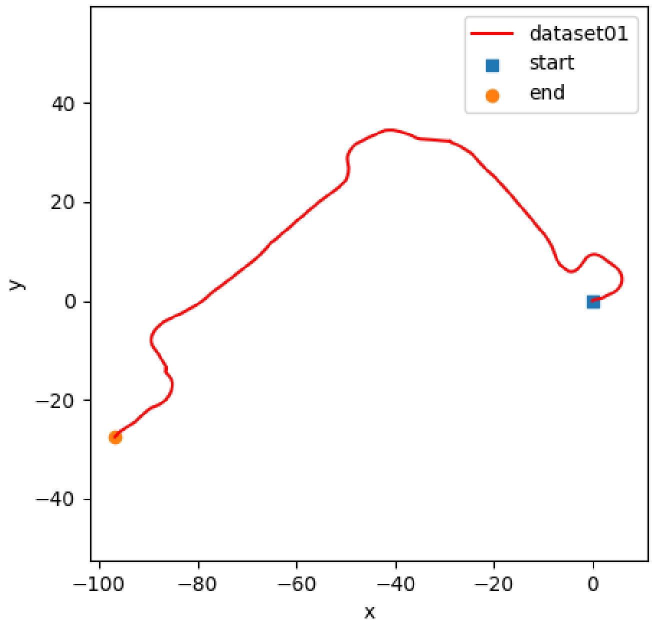


Fig. 2. IMU Localization for Dataset 01

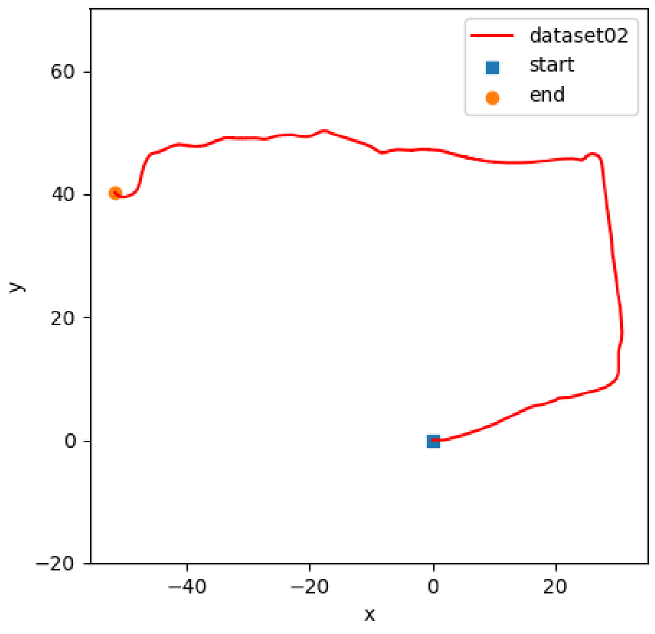
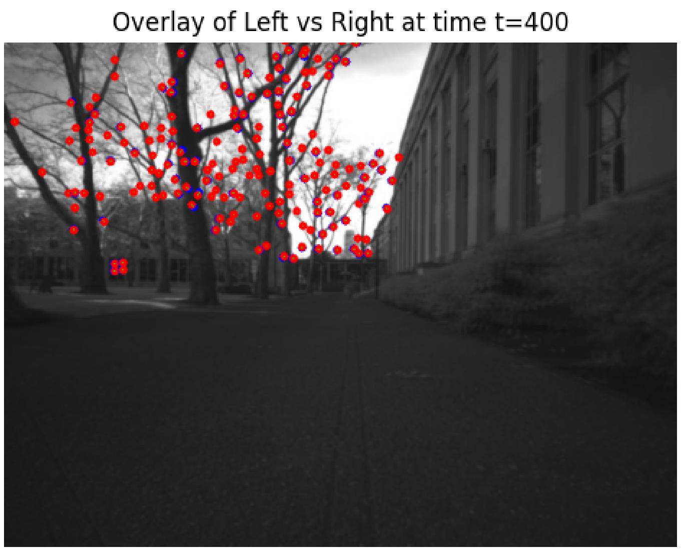


Fig. 3. IMU Localization for Dataset 02



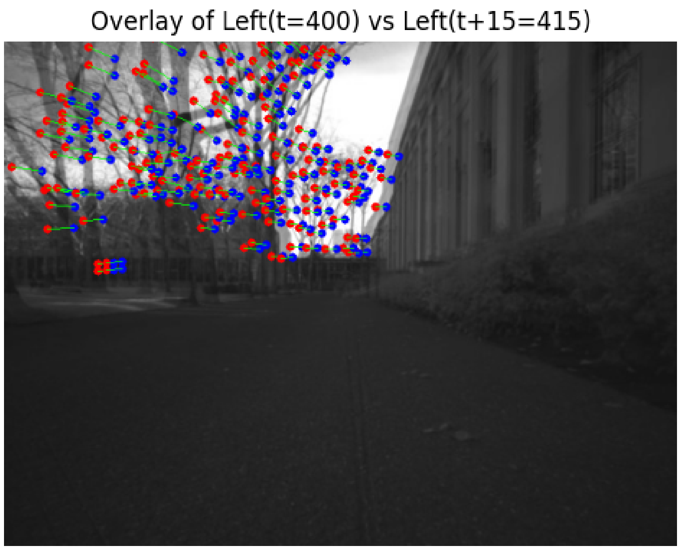


Fig. 4. Features detected from left and right camera overalayed (top image)

Fig. 5. Temporal tracking between features within the same camera (bottom image)

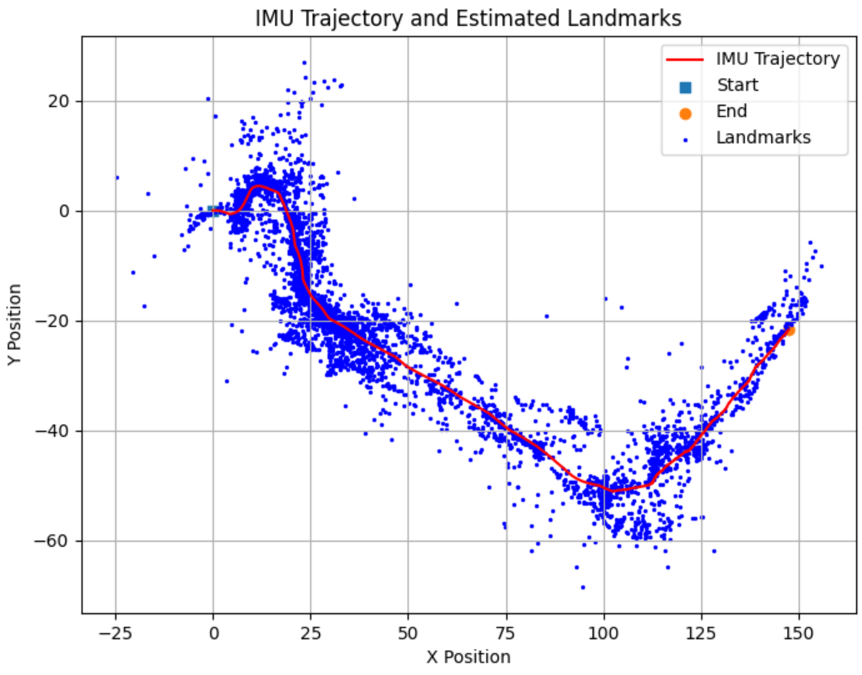


Fig. 6. Landmarks Initialization for Dataset00

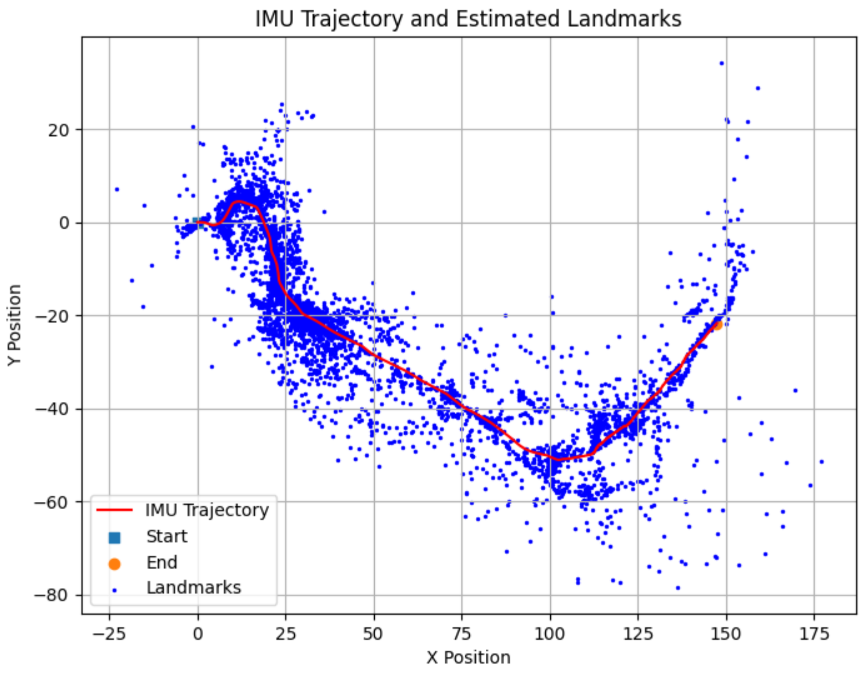


Fig. 7. Updated Landmark Locations for Dataset00

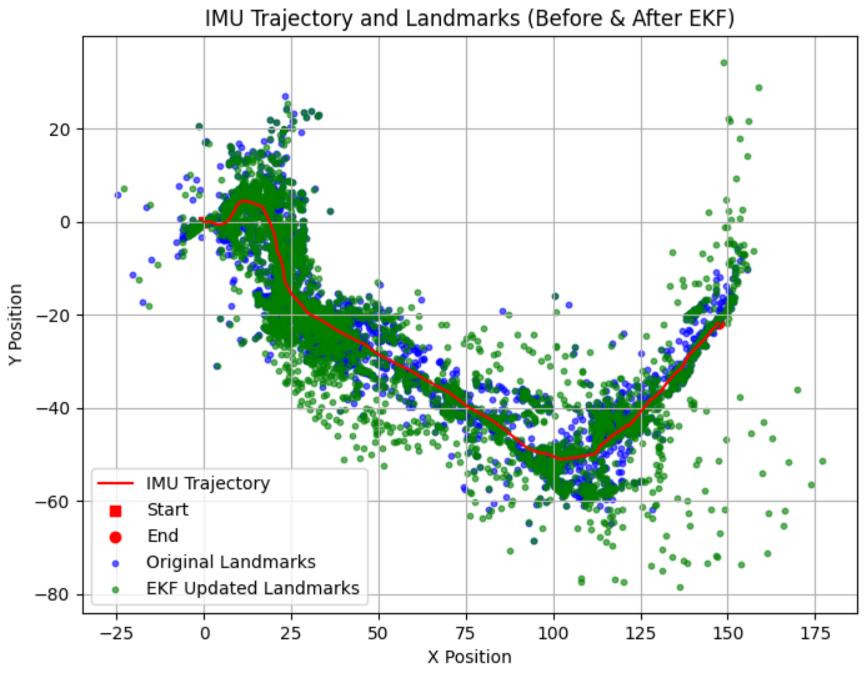


Fig. 8. Initial and Updated Landmarks Overlayed for Dataset00

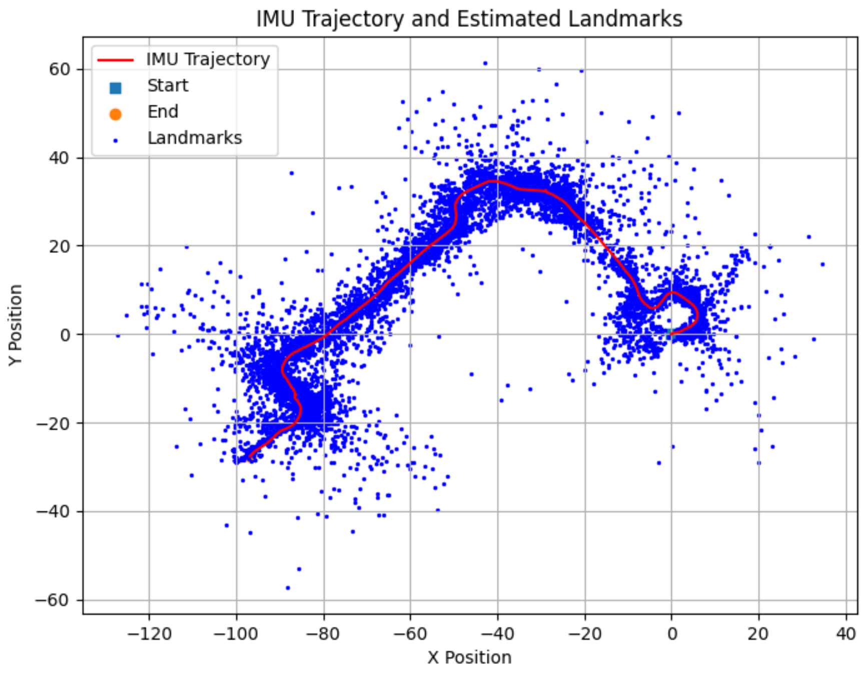


Fig. 9. Landmarks Initialization for Dataset01

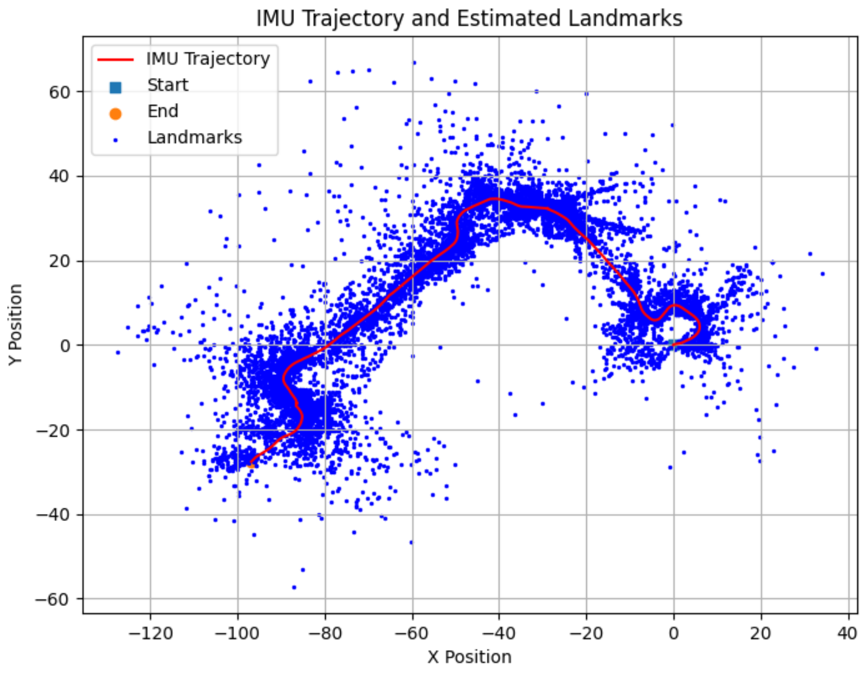


Fig. 10. Updated Landmark Locations for Dataset01

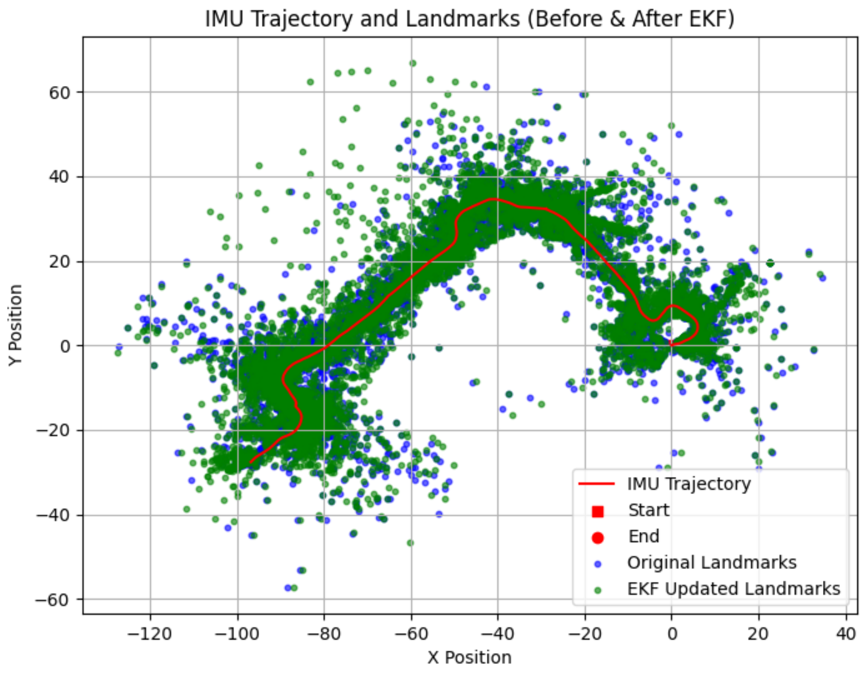


Fig. 11. Initial and Updated Landmarks Overlayed for Dataset01

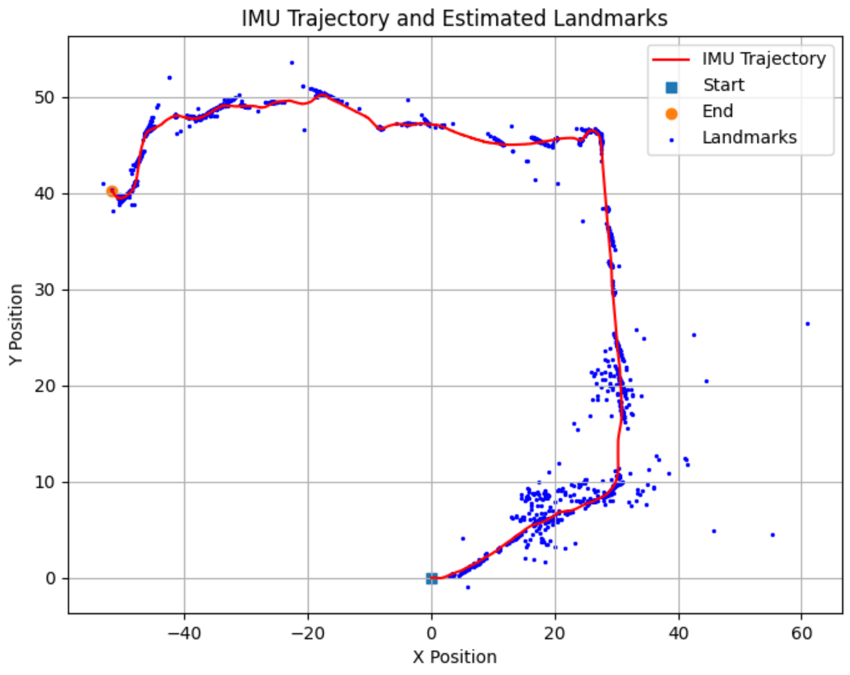


Fig. 12. Landmarks Initialization for Dataset02 (self-generated features)

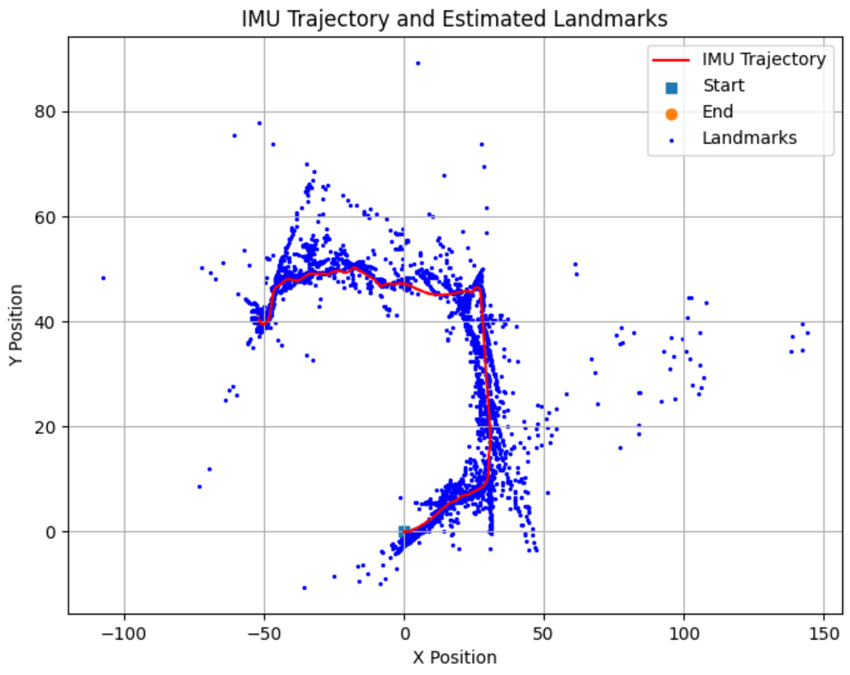


Fig. 13. Updated Landmark Locations for Dataset02 (self-generated features)

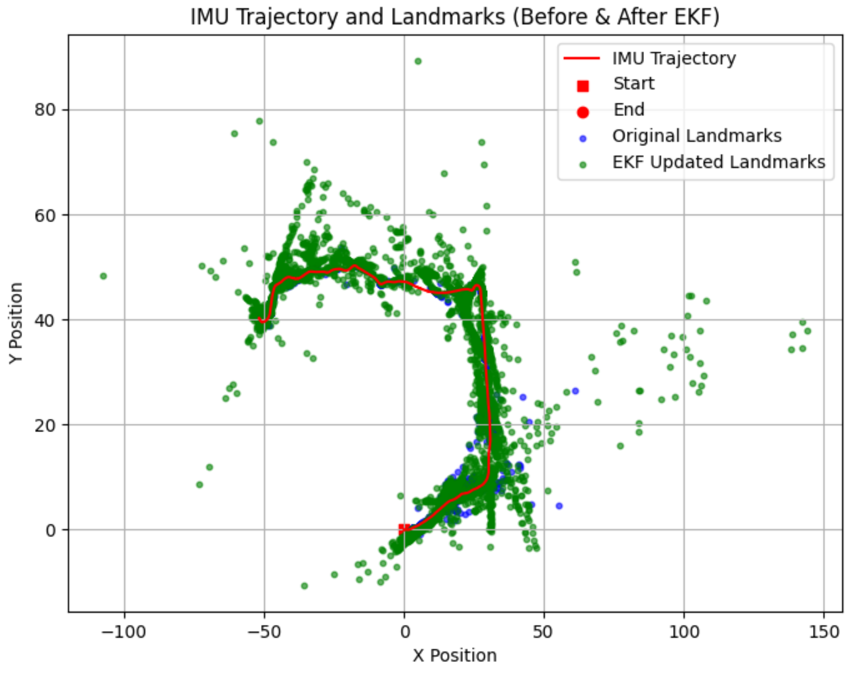


Fig. 14. Initial and Updated Landmarks Overlayed for Dataset02 (self-generated features)

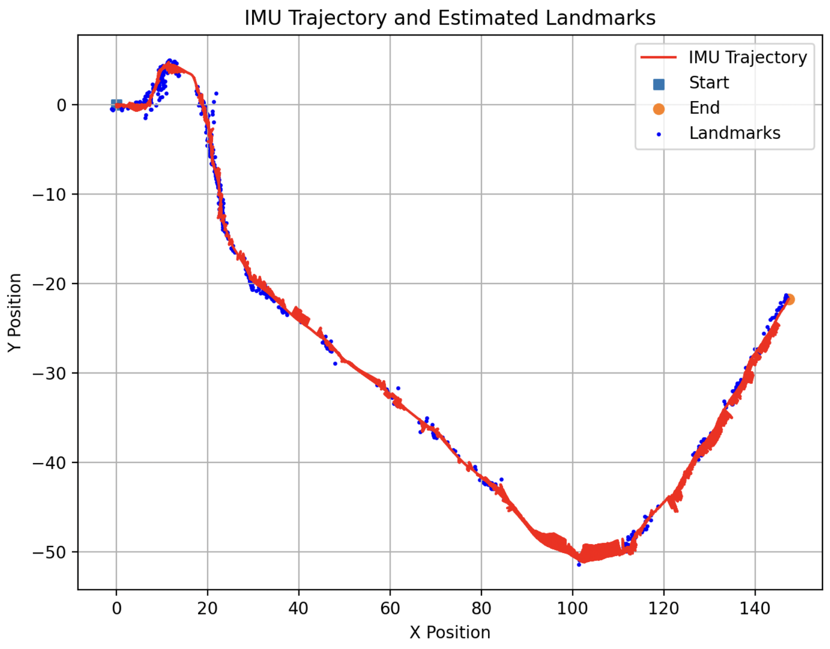


Fig. 15. Final SLAM for Dataset00

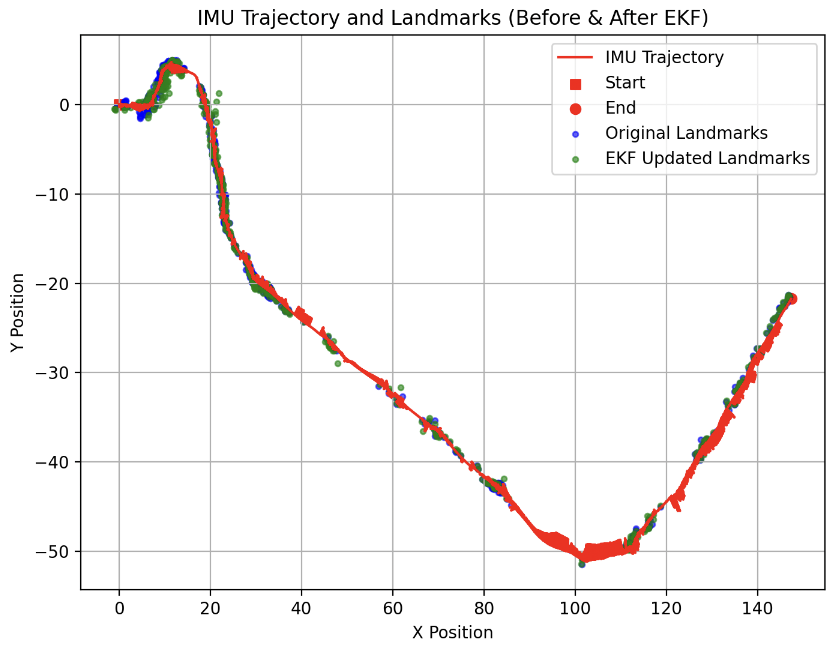


Fig. 16. Final SLAM for Dataset00 with Updated Landmarks

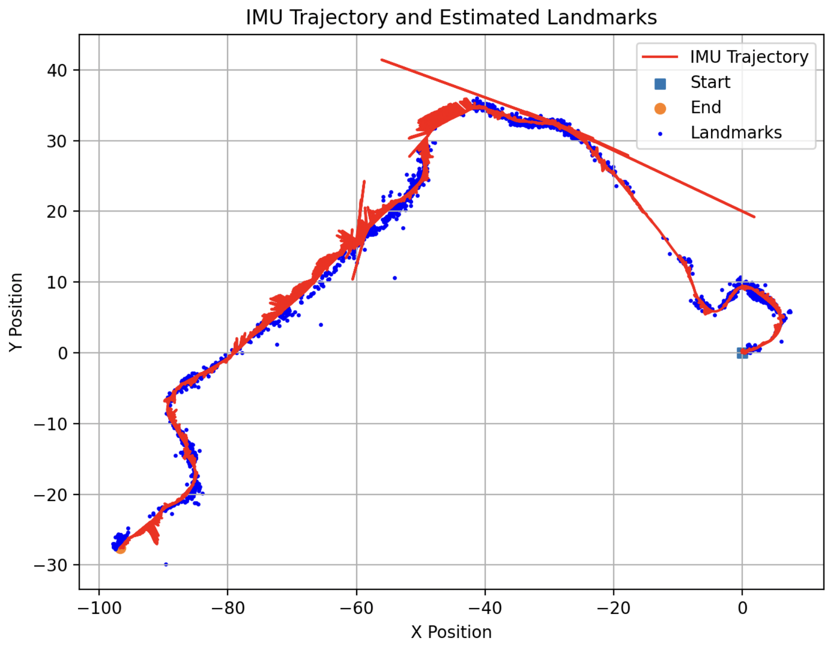


Fig. 17. Final SLAM for Dataset01

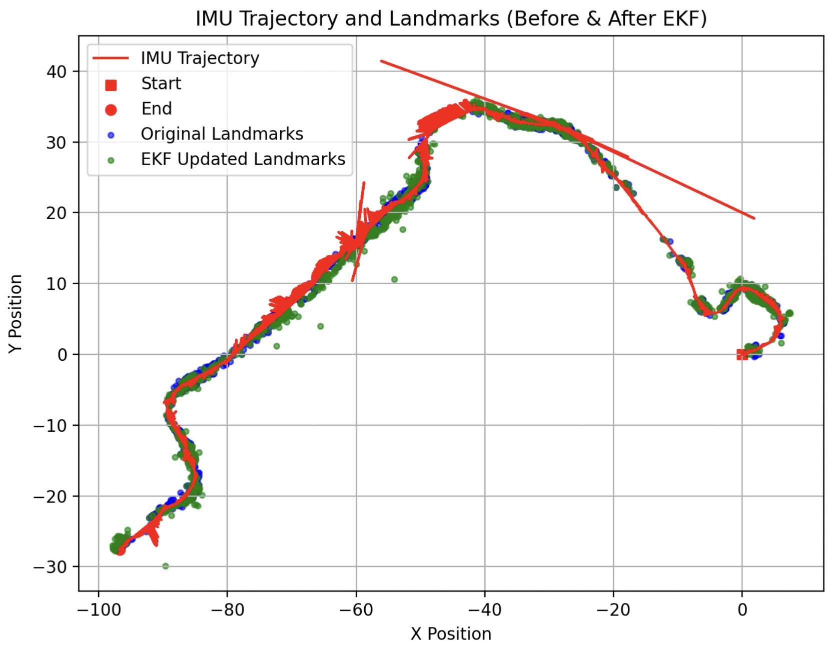


Fig. 18. Final SLAM for Dataset01 with Updated Landmarks

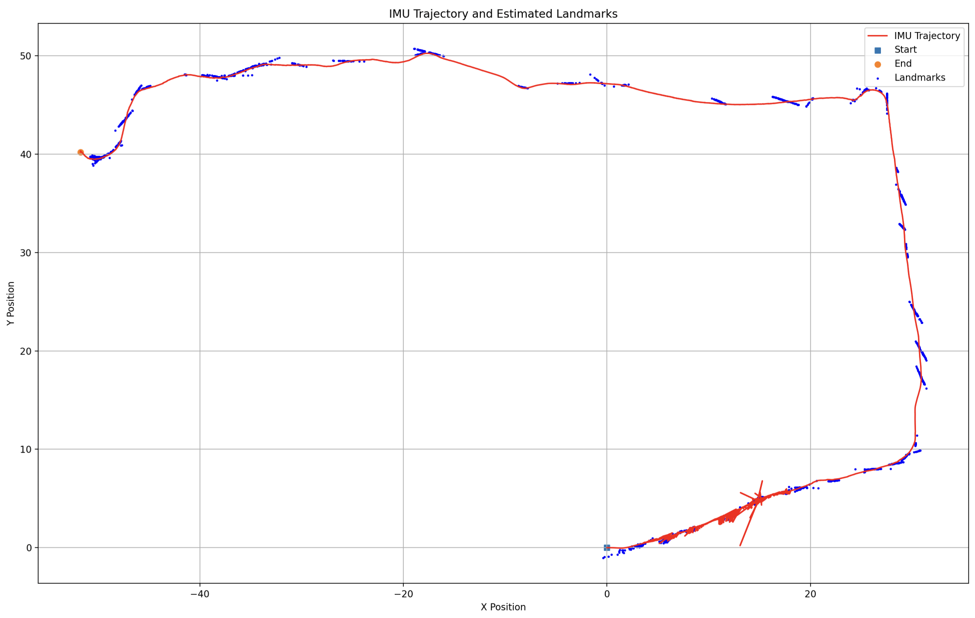


Fig. 19. Final SLAM for Dataset02 (first 500 timestamps

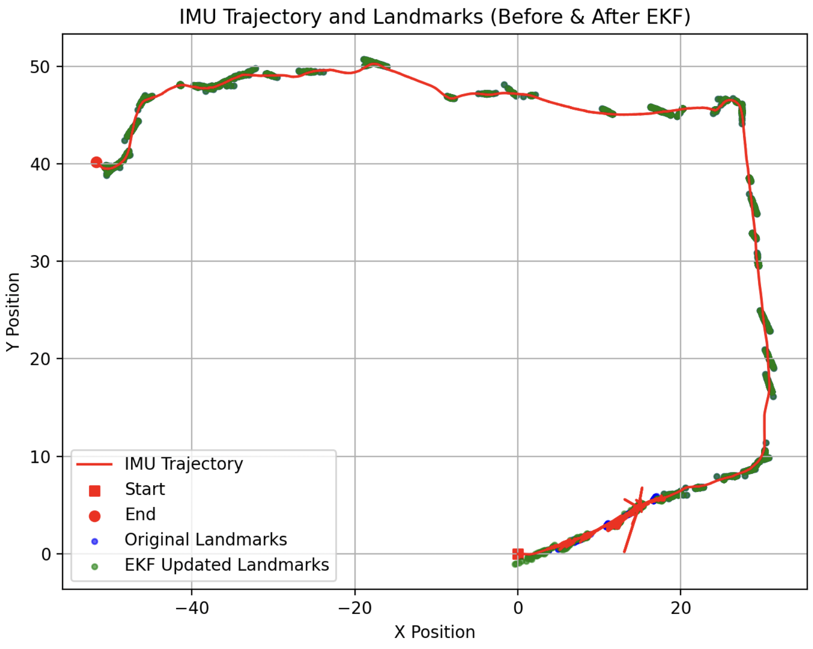


Fig. 20. Final SLAM for Dataset02 with Updated Landmarks (first 500 timestamps