# Visual-Inertial Simultaneous Localization and Mapping

Charvi Shukla
*Computer Science and Engineering Department*
*University of California, San Diego*
cshukla@ucsd.edu

***Abstract*—In this project, we implement visual-inertial simultaneous localization and mapping (SLAM) using an extended Kalman Filter (EKF). The data used in this project is collected by Clearpath Jackal robots navigating on MIT's campus in Cambridge, MA. We use linear and angular velocity measurements from an Inertial Measurement Unit and $(600 \times 480)$ gray scale images from a stereo camera. We implement a EKF prediction step to estimate the pose $T_t \in SE(3)$ of the IMU over time $t$, using linear and angular velocity measurements from the IMU. Then, we use an EKF update step to estimate the positions of the landmarks observed in the images, assuming that the prior trajectory is correct. Finally, we implement a full visual SLAM pipeline to simultaneously estimate the world frame IMU pose ${}_WT_I \in SE(3)$ and world frame landmark coordinates $m_j \in \mathbb{R}^3$ of the point landmarks, generated from the provided visual features $z_{j,i} \in \mathbb{R}^4$. We also do numerous optimizations to improve the runtime complexity of our program.**

***Index Terms*—Robotics, SLAM, Camera**

## I. Introduction

Simultaneous Localization and mapping, or SLAM, is the problem of using sensor measurements to simultaneously estimate the robot's state trajectory and build a map m of the environment. This is an important problem for autonomous systems since it is not always possible to have prior information about a robot's surroundings. Therefore, real-world application rely on robust SLAM pipelines, including autonomous vehicles like Waymo and Zoox, to even free-roaming platforms like the autonomous Olaf robotic character recently developed by Disney and NVIDIA!

In this project, we implement a visual-inertial SLAM. Specifically, we fuse visual data from a stereo camera with kinematic measurements from an Inertial Measurement Unit (IMU) using an Extended Kalman Filter. Since the pose kinematics of the differential drive Jackal robot are highly non-linear, we must use an EKF to propagate the robot's state across time. The IMU provides high frequency data, and while the control inputs and the discrete time differential drive motion model, can be used to estimate the robot trajectory, there is a lot of drift. To correct for this drift, we use a camera for visual feature tracking. By anchoring the robot's estimated pose to these observed visual landmarks, the EKF periodically corrects the trajectory and refines the landmark map, leading to a more robust localization and mapping.

This project is implemented in three main parts:

1) IMU localization via EKF prediction
2) Landmark mapping via EKF update
3) Visual-inertial SLAM Implementation and Optimizations
4) EKF noise and hyperparameter tuning

## II. Problem Formulation

### A. Robot State, Observation and Landmark Formulation

The state of the robot at timestamp $t$ is described by its position and orientation in the world frame $\{W\}$. The robot pose is defined as a member of the Special Euclidean Group, $T_t \in SE(3)$, represented by a $4 \times 4$ matrix:

$$T_t = \begin{bmatrix} R_t & p_t \\ 0^\top & 1 \end{bmatrix}$$

where $R_t \in SO(3)$ is the rotation matrix and $p_t \in \mathbb{R}^3$ is the position vector.The environment consists of $M$ static landmarks. Each landmark $j \in \{1, \ldots, M\}$ is represented by its 3D coordinates in the world frame:

$$m_j = \begin{bmatrix} x_j & y_j & z_j \end{bmatrix}^\top \in \mathbb{R}^3$$

At time $t$, the robot observes a subset of these landmarks via a stereo camera. The measurement for a detected landmark $j$ is defined by its corresponding pixel coordinates in the left and right image frames:

$$\mathbf{z}_{t,j} = \begin{bmatrix} u_{L,j} & v_{L,j} & u_{R,j} & v_{R,j} \end{bmatrix}^\top \in \mathbb{R}^4$$

where $(u_{L,j}, v_{L,j})$ are the $x$ and $y$ pixel coordinates in the left camera , and $(u_{R,j}, v_{R,j})$ are the coordinates in the right camera

### B. Motion and Observation Models

(i) Motion Model: The robot receives generalized velocity control inputs $\zeta(t)$ in the body frame $\{B\}$ from the IMU, consisting of linear velocity $\mathbf{v}(t)$ and angular velocity $\boldsymbol{\omega}(t)$:

$$\zeta(t) = \begin{bmatrix} \mathbf{v}(t) \\ \boldsymbol{\omega}(t) \end{bmatrix} \in \mathbb{R}^6$$

Assuming $\zeta(t)$ is constant over the time interval $\tau_t = t_{k+1} - t_k$, the discrete-time motion model governing the state evolution is given by the matrix exponential:

$$T_{t+1} = T_t \exp(\tau_t \hat{\zeta}_t) \exp(\mathbf{w}_t)$$

where $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, W)$ is the zero-mean Gaussian motion noise modeled as a perturbation in the Lie algebra $\mathfrak{se}(3)$. The

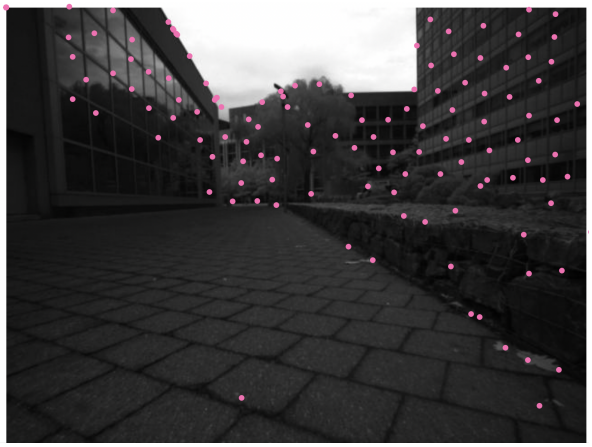


Fig. 1. Image of the environment observed by the left camera of the robot. The pink dots represent the visual features. That is, they are the pixel coordinates of the feature in the left camera frame.

objective of the localization step is to estimate the posterior distribution of $T_t$ given the inputs $\zeta_{0:t}$.

(ii) Observation Model At time $t$, the robot observes visual features from the stereo camera. The observation $\mathbf{z}_{t,j} \in \mathbb{R}^4$ represents the left and right pixel coordinates of the projected landmark $m_j$. The non-linear observation model $h(\cdot)$ maps a 3D landmark $m_j$ in the world frame to the stereo image planes:

$$\mathbf{z}_{t,j} = h(T_t, m_j) + \mathbf{v}_{t,j} = K_s \pi({}_O T_I T_t^{-1} \underline{m}_j) + \mathbf{v}_{t,j}$$

where:

- $K_s$ is the stereo intrinsic calibration matrix.
- ${}_O T_I$ is the known static extrinsic transformation from the IMU frame to the optical frame.
- $\pi(\cdot)$ is the perspective projection function.
- $\mathbf{v}_{t,j} \sim \mathcal{N}(\mathbf{0}, V)$ is the Gaussian pixel observation noise.

### C. Visual Inertial Odometry Problem (Localization)

The process of estimating the robot state trajectory $\mathbf{x_{0:T}}$, given a map $m$ of the environment, sensor measurements $\mathbf{z_{0:T}}$ with observation model $h$, and control inputs $\mathbf{u_{0:T-1}}$ with motion model $f$ is known as **localization**. While doing so, we wish to minimize the error:

$$\min_{\mathbf{x}} \sum_{t=0}^{T} \|\mathbf{z}_t - h(\mathbf{x}_t, \mathbf{m})\|_2^2$$

In the first part of the project, we wish to get an initial estimate of the robot state trajectory (i.e. its poses $T_t$ over time) using the angular $\omega(t)$ and linear velocity, $\mathbf{v}(t)$, measurements obtained from the IMU. Since we do not have a map of the environment as an input, and we solely rely on the IMU measurements, this turns into an *odometry problem*.

In order to formulate this as a visual-inertial *localization* problem, we incorporate visual feature observations $\mathbf{z}_{0:T}$. To do this, we rely on two main assumptions:

- We *assume* the world-frame landmark coordinates $\mathbf{m} \in \mathbb{R}^{3M}$ are known
- We *assume* a known mapping $\Delta_t : \{1, \dots, M\} \to \{1, \dots, N_t\}$, provided by an external algorithm, which stipulates that landmark $j$ corresponds to observation $\mathbf{z}_{t,i} \in \mathbb{R}^4$ with $i = \Delta_t(j)$ at time $t$.

### D. Landmark Mapping Problem

Mapping refers to the problem of building a map $m$ of the environment, given robot state trajectory $\mathbf{x_{0:T}}$ (in our case the IMU pose trajectory, $T_{0:T}$), and sensor measurements $\mathbf{z_{0:T}}$ with observation model $h$. In our case, we use IMU poses $T_{0:t}$ from (C) as inputs to our mapping problem and we want to estimate the 3D coordinates of the static landmarks

$$\mathbf{m} := [\mathbf{m}_1^\top \dots \mathbf{m}_M^\top]^\top \in \mathbb{R}^{3M}$$

with observations

$$\mathbf{z}_t = \begin{bmatrix} u_{L,j} & v_{L,j} & u_{R,j} & v_{R,j} \end{bmatrix}^\top \in \mathbb{R}^4$$

where $j$ is the index of a given landmark. Moreover, we have two important assumptions:

- The data association $\Delta_t$, which stipulates that a specific landmark $j$ corresponds to a specific observation at time $t$, is already known or provided by an external algorithm.
- Since the landmarks $\mathbf{m}$ are static, it is not necessary to consider a motion model or a prediction step for them.

### E. Visual-Inertial SLAM

The Simultaneous Localization and Mapping (SLAM) problem combines the objectives of localization and mapping into a joint estimation task.
That is, given an initial robot pose $T_0$, a sequence of visual sensor measurements $\mathbf{z}_{0:T}$ with observation model $h$ , and a sequence of IMU generalized velocity control inputs $\zeta_{0:T-1}$ with motion model $f$ , our objective is to simultaneously estimate the robot state trajectory $T_{1:T} \in SE(3)$ and build a structural map by estimating the 3D world-frame coordinates of the static point landmarks $\mathbf{m} \in \mathbb{R}^{3M}$.
This can also be posed as an optimization problem over the entire trajectory. In other words, Visual-Inertial SLAM seeks to find the optimal poses and landmark coordinates that jointly minimize the error from both the motion model and the observation model together.

## III. Technical Approach

### A. Data Processing and Handling

We have the following data:

- **Timestamps**: 1D Array containing time stamps for each measurement. There are $T$ timestamps in total.
- **IMU Measurements**: The linear and angular velocities are each stored in arrays of shape $(T, 3)$, providing the 3D angular and linear velocity vectors for the robot at every time step.
- **Features:** The stereo feature observations are stored in a $(4, F, T)$, where $F$ is the total number of features. and the $4$ corresponds to the pixel coordinates $[u_L, v_L, u_R, v_R]^\top$

- **Calibration Matrices:** We have left and right calibration matrices of the form:

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

- **Extrsinsic Matricies**: Provide the transformations from the IMU to the left and right optical frames, represented as $4 \times 4$ matrices belonging to $SE(3)$. Note that the raw dataset provides the transformations from the *regular* camera frames to the IMU frame (denoted as ${}_IT_L$ and ${}_IT_R$). To construct the required IMU-to-optical extrinsic matrix ${}_OT_I$, we must do the transformation:

$${}_OT_I = {}_oT_r {}_IT_L^{-1}$$

This resulting matrix then contains ${}_OR_I \in SO(3)$ and ${}_O\mathbf{p}_I \in \mathbb{R}^3$ to go from $\{B\}$ to $\{\mathrm{O}\}$:

$${}_OT_I = \begin{bmatrix} {}_OR_I & {}_O\mathbf{p}_I \\ \mathbf{0}^\top & 1 \end{bmatrix}$$

We do the following filtering:

1) Removing invalid observations: A given landmark $j$ may not be visible at every time step. Landmarks that are unobservable at a given time $t$ are recorded with a default measurement vector of $\mathbf{z}_{t,j} = \begin{bmatrix} -1 & -1 & -1 & -1 \end{bmatrix}^\top$. The missing observations are masked out by enforcing a valid pixel coordinate check ($u_L \geq 0$).
2) Disparity thresholding: We discard unreliable triangulations by enforcing an upper and lower bound on the disparity. This ensures that we are not using potentially noisy landmarks that are *too* close or *too* far from the robot.

### *B. Extended Kalman Filter (EKF) Formulation*

In this section, we will briefly cover the general formulation of an Extended Kalman Filter. At time $t$, we assume the prior state is normally distributed:

$$\mathbf{x}_t \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1} \sim \mathcal{N}(\boldsymbol{\mu}_{t|t}, \boldsymbol{\Sigma}_{t|t})$$

We define a non-linear motion model $f$ and a non-linear observation model $h$. The EKF approximates the exact belief of the state by linearizing these non-linear models using a first-order Taylor series expansion:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t), \quad \mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, W) \quad \text{(Motion Model)}$$

$$\mathbf{z}_{t+1} = h(\mathbf{x}_{t+1}, \mathbf{v}_{t+1}), \quad \mathbf{v}_{t+1} \sim \mathcal{N}(\mathbf{0}, V) \quad \text{(Obs. Model)}$$

In order to linearize them, we compute their Jacobians with respect to the state and the noise, and evaluate them at the *current* mean estimates:

$$F_t := \frac{\partial f}{\partial \mathbf{x}}(\boldsymbol{\mu}_{t|t}, \mathbf{u}_t, \mathbf{0}), \quad Q_t := \frac{\partial f}{\partial \mathbf{w}}(\boldsymbol{\mu}_{t|t}, \mathbf{u}_t, \mathbf{0})$$

$$H_{t+1} := \frac{\partial h}{\partial \mathbf{x}}(\boldsymbol{\mu}_{t+1|t}, \mathbf{0}), \quad R_{t+1} := \frac{\partial h}{\partial \mathbf{v}}(\boldsymbol{\mu}_{t+1|t}, \mathbf{0})$$

**Predict Step**: The prediction step uses the control input $\mathbf{u_t}$ to propagate the state and its covariance forward in time. By applying the linearized model approximation $f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t) \approx f(\boldsymbol{\mu}_{t|t}, \mathbf{u}_t, \mathbf{0}) + F_t(\mathbf{x}_t - \boldsymbol{\mu}_{t|t}) + Q_t\mathbf{w}_t$, the predicted mean and covariance are:

$$\boldsymbol{\mu}_{t+1|t} = f(\boldsymbol{\mu}_{t|t}, \mathbf{u}_t, \mathbf{0})$$

$$\boldsymbol{\Sigma}_{t+1|t} = F_t \boldsymbol{\Sigma}_{t|t} F_t^\top + Q_t W Q_t^\top$$

**Update Step:**
The EKF update step approximates the joint distribution of the predicted state $\mathbf{x}_{t+1}$ and the new measurement $\mathbf{z}_{t+1}$ as a multivariate Gaussian:

$$\begin{pmatrix} \mathbf{x}_{t+1} \\ \mathbf{z}_{t+1} \end{pmatrix} \sim \mathcal{N}\left( \begin{pmatrix} \boldsymbol{\mu}_{t+1|t} \\ \mathbf{m}_{t+1|t} \end{pmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{t+1|t} & C_{t+1|t} \\ C_{t+1|t}^\top & S_{t+1|t} \end{bmatrix} \right)$$

$$S_{t+1|t} = H_{t+1} \boldsymbol{\Sigma}_{t+1|t} H_{t+1}^\top + R_{t+1} V R_{t+1}^\top$$

where the predicted measurement, $\mathbf{m}_{t+1|t}$, and the cross-covariance, $C_{t+1|t}$ are defined as:

$$\mathbf{m}_{t+1|t} = h(\boldsymbol{\mu}_{t+1|t}, \mathbf{0})$$

$$C_{t+1|t} = \boldsymbol{\Sigma}_{t+1|t} H_{t+1}^\top$$

From the conditional gaussian distribution of $\mathbf{x}_{t+1} \mid \mathbf{z}_{t+1}, \mathbf{z}_{0:t}, \mathbf{u}_{0:t}$, we obtain the following update equations:

$$\boldsymbol{\mu}_{t+1|t+1} = \boldsymbol{\mu}_{t+1|t} + K_{t+1|t}(\mathbf{z}_{t+1} - h(\boldsymbol{\mu}_{t+1|t}, \mathbf{0}))$$

$$\boldsymbol{\Sigma}_{t+1|t+1} = (I - K_{t+1|t} H_{t+1}) \boldsymbol{\Sigma}_{t+1|t}$$

$$K_{t+1|t} = C_{t+1|t}(H_{t+1} \boldsymbol{\Sigma}_{t+1|t} H_{t+1}^\top + R_{t+1} V R_{t+1}^\top)^{-1}$$

We will be using these general formulas in our implementation of Localization, Mapping and SLAM in the sections described next.

### *C. IMU Localization via EKF Prediction*

The visual inertial odometry problem, formulated as a localization task estimates the world-frame IMU poses $T_t := {}_WT_{I,t} \in SE(3)$ over time. To simplify the prediction step, we use kinematic rather than dynamic equations of motion. We assume that we have the history of IMU measurements $u_{0:T}$ with $u_t := [v_t^\top, \omega_t^\top]^\top \in \mathbb{R}^6$.
The motion model for the continuous-time IMU pose $T(t)$ with noise $w(t)$ is given by $\dot{T} = T(\hat{u} + \hat{w})$. To specify a Gaussian distribution over $T_t$, we express it as a deterministic nominal pose $\mu \in SE(3)$ with a very tiny Gaussian perturbation $\hat{\delta\mu} \in \mathfrak{se}(3)$:

$$T = \mu \exp(\hat{\delta\mu}) \approx \mu(I + \hat{\delta\mu})$$

Substituting this into the continuous-time kinematics allows us to split the motion into a nominal part ($\dot{\mu} = \mu\hat{u}$) and a perturbation part ($\dot{\delta\mu} = -\hat{u}\delta\mu + w$). In discrete time with discretization $\tau_t$, the kinematics become:

$$\text{nominal:} \quad \mu_{t+1} = \mu_t \exp(\tau_t \hat{u}_t)$$

$$\text{perturbation:} \quad \delta\mu_{t+1} = \exp(-\tau_t \hat{u}_t)\delta\mu_t + \sqrt{\tau_t} w_t$$

We assume a prior Gaussian distribution $T_t \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1} \sim \mathcal{N}(\mu_{t|t}, \Sigma_{t|t})$, where $\boldsymbol{\mu} \in SE(3)$ and $\Sigma_{t|t} \in \mathbb{R}^{3\times3}$. This means that we have $T_t = \mu_{t|t}\exp(\hat{\delta\mu}_{t|t})$ with $\delta\mu_{t|t} \sim \mathcal{N}(0, \Sigma_{t|t})$. Using the discrete-time kinematics and assuming the motion noise is $w_t \sim \mathcal{N}(\mathbf{0}, W)$, the EKF prediction step updates the mean and covariance as follows:

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

$$\Sigma_{t+1|t} = \mathbb{E}[\delta\mu_{t+1|t}\delta\mu_{t+1|t}^\top] = \exp(-\tau_t\hat{u}_t)\Sigma_{t|t}\exp(-\tau_t\hat{u}_t)^\top + \tau_t W$$

where we use the overloaded notation $\hat{u}_t$ to represent the $4\times4$ twist matrix in $\mathfrak{se}(3)$ for the mean update, and the $6\times6$ adjoint representation for the covariance update:

$$\hat{u}_t = \begin{bmatrix}\hat{\omega}_t & v_t \\ \mathbf{0}^\top & 0\end{bmatrix} \in \mathbb{R}^{4\times4}, \qquad \hat{u}_t = \begin{bmatrix}\hat{\omega}_t & \hat{v}_t \\ \mathbf{0} & \hat{\omega}_t\end{bmatrix} \in \mathbb{R}^{6\times6}$$

The exponential function is defined as: $\exp(X) = \sum_{n=0}^{\infty}\frac{1}{n!}X^n$.

$$\hat{\xi} = \tau_k\hat{\zeta}_k = \begin{bmatrix}\hat{\theta} & \rho \\ 0^\top & 0\end{bmatrix} \in \mathfrak{se}(3)$$

$$\exp(\hat{\xi}) = \begin{bmatrix}\exp(\hat{\theta}) & J_L(\theta)\rho \\ 0^\top & 1\end{bmatrix}$$

where $\exp(\hat{\theta}) \in SO(3)$ is computed via **Rodrigues' formula**, and $J_L(\theta)$ is the left Jacobian of $SO(3)$.

Note that while this section is framed within the context of IMU "Localization", the absence of a map $\mathbf{m}$ means we are just performing *inertial odometry* (dead-reckoning). This simplifies things as we do not use the visual landmarks and do not need to evaluate the observation model $h(T_t, \mathbf{m}_j)$. Consequently, we completely bypass the EKF update step, which bypasses the computation the Jacobian $H_t$ and the Kalman gian computation. This system relies entirely on integrating the twist from the motion model $f$, and while this runs really fast, it accumulates a lot of drift due to a lack of corrections.

### *D. Landmark Initialization: Triangulation*

Given a stereo observation $\mathbf{z} = [u_L, v_L, u_R, v_R]^\top$, we first compute the horizontal disparity $d = u_L - u_R$. Assuming the stereo cameras are perfectly rectified, we can get the depth as:

$$z_{opt} = \frac{f_u \cdot b}{d}$$

where $f_u$ is the horizontal focal length and $b$ is the distance between the cameras. Then, we get:

$$x_{opt} = \frac{(u_L - c_u)\cdot z_{opt}}{f_u}, \quad y_{opt} = \frac{(v_L - c_v)\cdot z_{opt}}{f_v}$$

This yields the 3D landmark coordinate in the optical frame, $\mathbf{q} = [x_{opt}, y_{opt}, z_{opt}, 1]^\top$. We map this to the world frame as follows:

$$\underline{\mathbf{m}}_j = T_t \cdot {}_IT_L \cdot {}_OT_I^{-1} \cdot \mathbf{q}$$

Here $f_v$ is the vertical focal length and $(c_u, c_v)$ are the origin of the image frame. We conduct this step to initialize the landmark in the global frame so they can be used as a prior mean estimate during our application of EKF.

### *E. Landmark Mapping via EKF Update*

We assume that the map of landmarks, $\mathbf{m} \in \mathbb{R}^{3M}$ do not move over time. Therefore, we do not need an EKF Predict step, which uses the motion model. Instead, the map is iteratively constructed and refined purely by applying the EKF update step as new visual feature observations $\mathbf{z}_{t+1} \in \mathbb{R}^{4N_{t+1}}$ are obtained.

Assuming the IMU pose $T_{t+1} \in SE(3)$ is known, we estimate the 3D coordinates of the landmarks. We assume that the prior landmark state distribution is a Gaussian, with $\boldsymbol{\mu}_t^m \in \mathbb{R}^{3M}$ and $\boldsymbol{\Sigma}_t^m \in \mathbb{R}^{3M\times3M}$:

$$\mathbf{m}_t \mid \mathbf{z}_{0:t} \sim \mathcal{N}(\boldsymbol{\mu}_t^m, \boldsymbol{\Sigma}_t^m)$$

As we mentioned before, because the landmarks are static, the prediction step is trivial:

$$\boldsymbol{\mu}_{t+1|t}^m = \boldsymbol{\mu}_t^m$$

$$\boldsymbol{\Sigma}_{t+1|t}^m = \boldsymbol{\Sigma}_t^m$$

The observation model $h(\cdot)$ projects a single 3D landmark $\mathbf{m}_j \in \mathbb{R}^3$ into the stereo camera's left and right image planes to produce a $4 \times 1$ pixel coordinate vector:

$$\begin{aligned}\mathbf{z}_{t+1,j} &= h(T_{t+1}, \mathbf{m}_j) + \mathbf{v}_{t+1,j} \\ &= K_s\pi(OT_IT t + 1^{-1}\underline{\mathbf{m}}_j) + \mathbf{v}_{t+1,j}\end{aligned}$$

where $K_s$ is the stereo intrinsic matrix , ${}_OT_I$ is the known IMU-to-optical frame extrinsic calibration , $\underline{\mathbf{m}}_j \in \mathbb{R}^4$ is the homogeneous representation of the landmark , and $\mathbf{v}_{t+1,i} \sim \mathcal{N}(\mathbf{0}, V)$ is the measurement noise.

Using this non-linear observation model, the EKF update step corrects the landmark position estimates. The predicted observation based on the prior mean is $\tilde{\mathbf{z}}_{t+1} = h(T_{t+1}, \boldsymbol{\mu}_t^m)$. The Kalman Gain $K_{t+1}$ and the updated landmark state are computed as:

$$K_{t+1} = \boldsymbol{\Sigma}_t^m H_{t+1}^\top (H_{t+1}\boldsymbol{\Sigma}_t^m H_{t+1}^\top + I \otimes V)^{-1}$$

$$\boldsymbol{\mu}_{t+1}^m = \boldsymbol{\mu}_t^m + K_{t+1}(\mathbf{z}_{t+1} - \tilde{\mathbf{z}}_{t+1})$$

$$\boldsymbol{\Sigma}_{t+1}^m = (I - K_{t+1}H_{t+1})\boldsymbol{\Sigma}_t^m$$

In the computation for $K_{t+1}$, $I \otimes V$ represents the diagonal observation noise covariance for all $N_t$ observed features. Furthermore, the Jacobian $H_{t+1} \in \mathbb{R}^{4N_t\times3M}$ with respect to the 3D landmark position $\mathbf{m}_j \in \mathbb{R}^3$ is computed as follows. We evaluate the derivative at the *best current mean*, $\boldsymbol{\mu}_{t,j}^m$:

$$H_{t+1,i,j} = \frac{\partial}{\partial\mathbf{m}_j}h(T_{t+1}, \mathbf{m}_j)\bigg|_{\mathbf{m}_j = \boldsymbol{\mu}_{t,j}^m}$$

We apply the chain rule:

$$\frac{\partial h}{\partial\mathbf{m}_j} = K_s\frac{\partial\pi}{\partial\mathbf{q}}\cdot\frac{\partial\mathbf{q}}{\partial\mathbf{m}_j}$$

where

$$\mathbf{q} = {}_O T_I T_{t+1}^{-1} \underline{\mathbf{m}}_j$$

We need to get the derivative of $\mathbf{q}$ with respect to the landmark $\mathbf{m}_j$. The transformations are treated as constants wih respect to $\mathbf{m}_j$, and then we get:

$$\underline{\mathbf{m}}_j = \begin{bmatrix} m_x \\ m_y \\ m_z \\ 1 \end{bmatrix}, \quad \mathbf{m}_j = \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix}$$

$$\implies \frac{\partial \underline{\mathbf{m}}_j}{\partial \mathbf{m}_j} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = P^\top$$

$$\implies \frac{\partial \mathbf{q}}{\partial \mathbf{m}_j} = {}_O T_I T_{t+1}^{-1} P^\top$$

We substitute $\frac{\partial \mathbf{q}}{\partial \mathbf{m}_j}$ back into our chain rule equation shown earlier:

$$H_{t+1,i,j} = K_s \frac{\partial \pi}{\partial \mathbf{q}} ({}_O T_I T_{t+1}^{-1} \underline{\mathbf{m}}_j) \cdot {}_O T_I T_{t+1}^{-1} P^\top$$

Assuming the data association $\Delta_{t+1}(j) = i$ is known , the specific block elements $H_{t+1,i,j} \in \mathbb{R}^{4\times 3}$ are defined as:

$$H_{t+1,i,j} = \begin{cases} K_s \frac{\partial \pi}{\partial \mathbf{q}} ({}_O T_I T_{t+1}^{-1} \underline{\boldsymbol{\mu}}_{t,j}^m) {}_O T_I T_{t+1}^{-1} P^\top, & \text{if } \Delta_{t+1}(j) = i \\ \mathbf{0}, & \text{otherwise} \end{cases}$$

Finally, to construct the complete observation Jacobian $H_{t+1}$ for all $N_{t+1}$ currently observed features, we stack these individual $H_{t+1,i,j}$ blocks. For each i-j correspondence of observation and landmark, $H_{t+1,i,j}$ contains the Jacobian of observation $i$, with respect to landmark $j$. Due to the corresponding $\Delta$, only one element of each row in $H_{t+1}$ will be a non-zero matrix at one given timestep:

$$H_{t+1} = \begin{bmatrix} H_{t+1,1,1} & H_{t+1,1,2} & \cdots & H_{t+1,1,M} \\ H_{t+1,2,1} & H_{t+1,2,2} & \cdots & H_{t+1,2,M} \\ \vdots & \vdots & \ddots & \vdots \\ H_{t+1,N_{t+1},1} & H_{t+1,N_{t+1},2} & \cdots & H_{t+1,N_{t+1},M} \end{bmatrix}$$

Hence, by using the known IMU poses, the EKF update equations, and the observations from the stereo camera, we iteratively construct the 3D map $\mathbf{m}$ of the environment.

### *F. Visual SLAM*

SLAM combines the objectives of odometry and mapping into a joint estimation task. Now we do not make the assumption that we know the map or a previous trajectory and hope to estimate them together.

We define a joint state consisting of the current IMU pose $T_t \in SE(3)$ and the 3D coordinates of all static landmarks $\mathbf{m} \in \mathbb{R}^{3M}$. We maintain the joint Gaussian of this combined state. The joint covariance matrix $\boldsymbol{\Sigma}_t \in \mathbb{R}^{(6+3M)\times(6+3M)}$ captures not only the individual uncertainties of the pose ($\boldsymbol{\Sigma}^{TT}$) and the map ($\boldsymbol{\Sigma}^{mm}$), but crucially, the cross-correlations between them ($\boldsymbol{\Sigma}^{Tm}$)

$$\boldsymbol{\Sigma}_t = \begin{bmatrix} \boldsymbol{\Sigma}_t^{TT} & \boldsymbol{\Sigma}_t^{Tm} \\ (\boldsymbol{\Sigma}_t^{Tm})^\top & \boldsymbol{\Sigma}_t^{mm} \end{bmatrix}$$

Because the landmarks are completely static, the prediction step only actively propagates the IMU pose $T_t$ using the generalized velocity inputs $\zeta_t$. Applying the separated nominal and perturbation kinematics derived earlier, we get the following updated means:

$$\boldsymbol{\mu}_{t+1|t} = \boldsymbol{\mu}_{t|t} \exp(\tau_t \hat{\zeta}_t)$$

$$\boldsymbol{\mu}_{t+1|t}^m = \boldsymbol{\mu}_{t|t}^m$$

The joint covariance is updated by applying the adjoint transformation to the pose block and the cross-covariance block. Because the map does not move, the map-map covariance block remains completely unchanged during prediction:

$$\boldsymbol{\Sigma}_{t+1|t} = \begin{bmatrix} \boldsymbol{\Sigma}_{t+1|t}^{TT} & \boldsymbol{\Sigma}_{t+1|t}^{Tm} \\ (\boldsymbol{\Sigma}_{t+1|t}^{Tm})^\top & \boldsymbol{\Sigma}_{t+1|t}^{mm} \end{bmatrix}$$

where the individual block components are updated as follows:

$$\boldsymbol{\Sigma} t+1|t^{TT} = \exp(-\tau_t \hat{\zeta} t) \boldsymbol{\Sigma}_{t|t}^{TT} \exp(-\tau_t \hat{\zeta} t)^\top + \tau_t W$$

$$\boldsymbol{\Sigma}_{t+1|t}^{Tm} = \exp(-\tau_t \hat{\zeta} t) \boldsymbol{\Sigma}_{t|t}^{Tm}$$

$$\boldsymbol{\Sigma}_{t+1|t}^{mm} = \boldsymbol{\Sigma}_{t|t}^{mm}$$

Once the observation is obtained, we will conduct the EKF Update step, which will correct the map and IMU pose. To compute the Kalman gain, we construct a joint observation Jacobian $H_{t+1} \in \mathbb{R}^{4N_{t+1}\times(6+3M)}$. By the chain rule, this is simply the concatenation of the derivative with respect to the pose and the derivative with respect to the map:

$$H_{t+1} = \begin{bmatrix} H_{t+1}^{pose} & H_{t+1}^{\mathbf{m}} \end{bmatrix}$$

We derived the $H_{t+1}^{\mathbf{m}} \in \mathbb{R}^{4N_{t+1}\times 3M}$ matrix in the previous section (E). Next, the pose block $H_{t+1}^{pose} \in \mathbb{R}^{4N_{t+1}\times 6}$ requires us to compute the observation model Jacobian with respect to the IMU pose $T_{t+1}$, evaluated at the predicted pose mean $\boldsymbol{\mu}_{t+1|t}$. For a specific observation $i$ corresponding to landmark $j$, we will use a first-order Taylor series approximation with a Lie algebra perturbation $\delta\boldsymbol{\mu} \in \mathbb{R}^6$. Starting with the observation model and using the inverse perturbation approximation $\exp(\delta\hat{\boldsymbol{\mu}})^{-1} \approx I - \delta\hat{\boldsymbol{\mu}}$:

$$\begin{aligned} z_{t+1,i} &= K_s \pi({}_O T_I(\boldsymbol{\mu}_{t+1|t} \exp(\delta\hat{\boldsymbol{\mu}}))^{-1} \underline{\mathbf{m}} j) + v_{t+1,i} \\ &\approx K_s \pi({}_O T_I (I - \delta\hat{\boldsymbol{\mu}}) \boldsymbol{\mu}_{t+1|t}^{-1} \underline{\mathbf{m}} j) + v_{t+1,i} \\ &= K_s \pi({}_O T_I \boldsymbol{\mu}_{t+1|t}^{-1} \underline{\mathbf{m}}_j - {}_O T_I (\boldsymbol{\mu}_{t+1|t}^{-1} \underline{\mathbf{m}}_j)^\odot \delta\boldsymbol{\mu}) + v_{t+1,i} \end{aligned}$$

Note: $\hat{\boldsymbol{\xi}} \underline{\mathbf{s}} = \underline{\mathbf{s}}^\odot \boldsymbol{\xi}$

Now, we do a Taylor Series Approximation to get the Jacobian:

$$\approx \underbrace{K_s \pi\left({}_O T_I \boldsymbol{\mu}_{t+1|t}^{-1} \underline{\mathbf{m}}_j\right)}_{\tilde{\mathbf{z}}_{t+1,i}} \underbrace{-K_s \frac{\partial \pi}{\partial \mathbf{q}} \left({}_O T_I \boldsymbol{\mu}_{t+1|t}^{-1} \underline{\mathbf{m}}_j\right) {}_O T_I \left(\boldsymbol{\mu}_{t+1|t}^{-1} \underline{\mathbf{m}}_j\right)^\odot}_{H_{t+1,i}^{pose}}$$

$$\delta\boldsymbol{\mu} + \mathbf{v}_{t+1,i}$$

Hence, the Jacobian for observation $i$ is:

$$H_{t+1,i}^{pose} = -K_s \frac{\partial \pi}{\partial \mathbf{q}}({}_O T_I \boldsymbol{\mu}_{t+1|t}^{-1} \underline{\mathbf{m}}_j)\, {}_O T_I (\boldsymbol{\mu}_{t+1|t}^{-1} \underline{\mathbf{m}}_j)^{\odot} \in \mathbb{R}^{4\times 6}$$

where:

- $K_s \in \mathbb{R}^{4\times 4}$ is the known stereo intrinsic calibration matrix.
- $\frac{\partial \pi}{\partial \mathbf{q}}(\mathbf{q}) \in \mathbb{R}^{4\times 4}$
- ${}_O T_I \in \mathbb{R}^{4\times 4}$ is the extrinsics
- $\odot$ operator transforms the homogeneous coordinates into a $4 \times 6$ matrix format:

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

We just stack these for all $N_{t+1}$ current observations and get the jacobian $H_{t+1}^{pose}$. Once we combine this with the Jacobian for landmarks, we obtain the full joint jacobian. Then, we can compute the Kalman Gain as:

$$K_{t+1} = \boldsymbol{\Sigma}_{t+1|t} H_{t+1}^{\top} (H_{t+1} \boldsymbol{\Sigma}_{t+1|t} H_{t+1}^{\top} + I \otimes V)^{-1}$$

Our joint state contains both a Lie group and a Euclidean vector space. Therefore, we partition this update vector into a "pose component" and a "map component":

$$\begin{bmatrix} \delta \boldsymbol{\mu}^{pose} \\ \delta \boldsymbol{\mu}^{\mathbf{m}} \end{bmatrix} = K_{t+1}(\mathbf{z}_{t+1} - \tilde{\mathbf{z}}_{t+1})$$

Then, pose is updated via the exponential map on the $SE(3)$ manifold, and the landmark coordinates are updated via standard vector addition in $\mathbb{R}^{3M}$:

$$\boldsymbol{\mu}_{t+1|t+1} = \boldsymbol{\mu}_{t+1|t} \exp((\delta \boldsymbol{\mu}^{pose})^{\wedge})$$
$$\boldsymbol{\mu}_{t+1|t+1}^{\mathbf{m}} = \boldsymbol{\mu}_{t+1|t}^{\mathbf{m}} + \delta \boldsymbol{\mu}^{\mathbf{m}}$$

Finally, the joint covariance matrix is updated to reflect the reduction in uncertainty after including our observation:

$$\boldsymbol{\Sigma}_{t+1|t+1} = (I - K_{t+1} H_{t+1}) \boldsymbol{\Sigma}_{t+1|t}$$

Through this joint update step, we refine the 3D positions of the mapped landmarks and also simultaneously correct the robot's pose trajectory. **Algorithm**

### G. *Computational Intensity and Optimizations*

In our SLAM solution, our joint covariance matrix $\boldsymbol{\Sigma}_t$, tracks 6 pose variables and $3M$ landmark variables. For large number of landmarks, this covariance matrix will be very large, and with floating point numbers, could take multiple gigabytes of RAM. However, this matrix is very sparse since most landmarks are not correlated with eachother. To accelerate our calculations, we utilize `SciPy`'s sparse matrix formats.

1) **LIL Matrix**:
   - This stores a matrix as two Python arrays of lists. There is a **data** array that contains non-zero values and another **rows** array that contains indices for those non-zero values with respect to our main matrix.
   - When we initialize the large covariance matrices and noise, we need to set the initial values. We use a LIL matrix here for speeding up the initialization
2) **CSR (Compressed Sparse Row) Matrix**
   - Uses 3 flat 1D Numpy arrays.
   - The **data** array every non-zero value in the matrix and reads left to right and then top to bottom. An **indices** array containing column index for every value in data. An **indptr** array that tells the computer where the new row starts.
   - Since data is tightly packed into arrays, it is contiguous in memory and therefore there is more spatial locality and less cache misses. This allows us to perform very fast arithmetic operations. This is not very good when we want to insert more elements as it would require us to shift all indices.
   - CSR matrix allows us to slice out the parts of the joint covariance matrix that we want to use.
3) In short, we construct matrices using `lil_matrix` and do operations using `csr_matrix`.

Moreover, during an EKF update at time $t$, the observation Jacobian $H_t$, as we know, is very sparse as not all landmarks are always visible.Instead of doing dense matrix multiplication, we use CSR indexing to extract only the active rows and columns. If we observe 30 features, we slice out a dense $(6+30\times 3)\times(6+30\times 3)$ sub-matrix. Imagine a system with one Pose ($P$) and three mapped landmarks ($L_1, L_2, L_3$). If the camera currently only observes $L_2$, we slice out exactly the rows and columns corresponding to $P$ and $L_2$:

$$\boldsymbol{\Sigma}_{joint} = \begin{bmatrix} \boldsymbol{\Sigma}_{PP} & \boldsymbol{\Sigma}_{PL_1} & \boldsymbol{\Sigma}_{PL_2} & \boldsymbol{\Sigma}_{PL_3} \\ \boldsymbol{\Sigma}_{L_1P} & \boldsymbol{\Sigma}_{L_1L_1} & \boldsymbol{\Sigma}_{L_1L_2} & \boldsymbol{\Sigma}_{L_1L_3} \\ \boldsymbol{\Sigma}_{L_2P} & \boldsymbol{\Sigma}_{L_2L_1} & \boldsymbol{\Sigma}_{L_2L_2} & \boldsymbol{\Sigma}_{L_2L_3} \\ \boldsymbol{\Sigma}_{L_3P} & \boldsymbol{\Sigma}_{L_3L_1} & \boldsymbol{\Sigma}_{L_3L_2} & \boldsymbol{\Sigma}_{L_3L_3} \end{bmatrix}$$

$$\implies \boldsymbol{\Sigma}_{current} = \begin{bmatrix} \boldsymbol{\Sigma}_{PP} & \boldsymbol{\Sigma}_{PL_2} \\ \boldsymbol{\Sigma}_{L_2P} & \boldsymbol{\Sigma}_{L_2L_2} \end{bmatrix}$$

We perform all of the computationally heavy matrix inversions and Kalman gain calculations on this small block. Finally, we map the computed block back into the massive sparse joint covariance matrix.

Additionally do not include all landmarks in our joint covariance matrix. Instead, we use the following 2 filter criterion:

- **Disparity Thresholding:** As mentioned before, we exclude the points that are too close or too far to the robot based on a `MIN_DISPARITY` and `MAX_DISPARITY` threshold. This allows us to prevent triangulation errors
- **Random Filtering:** Including all landmarks will make our joint covariance matrix really large (making the inverse and Kalman gain calculations very expensive). Therefore, for some specified `SKIP_VALUE`, we skip features whose `idx % SKIP_VALUE` is 0. This helps keep the size of the covariance matrix in control.

This also reduces our overall computational complexity and reduces the time taken to run the program, and we dont have

to invert and multiply matrices with thousands of rows and columns.

### H. Hyperparameter Tuning

For this project, we conduct the parameter tuning manually for the following parameters:

- Observation Noise Covariance ($V$ / `PIXEL_NOISE`)
- Motion Noise Covariance ($W$ / `MOTION_NOISE`)
- Initial Landmark Covariance ($\sigma^2_{landmark}$)
- Active Feature Limit (`NUM_FEATURES`)
- Map Sparsity (`LANDMARK_SKIP`)
- Minimum and maximum disparity (`MIN_DISPARITY`, `MAX_DISPARITY`)
- Prior Covariance (`SIGMA`)

## IV. Results

In this section, we will discuss the results in the following order:

- Localization-only (odometry) Results
- Mapping Results
- Visual Inertial SLAM results
- Videos of robot trajectory generation over time

### A. IMU Based Robot Localization Results

Figures 2 and 3 show our estimated poses $T_t \in SE(3)$ over time. The red dashed line represents the trajectory of the robot achieved using Pose Kinematics. We can note that the trajectory is smooth and does nto have any unrealistic sudden jumps. The blue *directional* segments overlaid on that red trajectory represent the robot's estimated heading (yaw) at sampled intervals. These vectors are derived from the SO(3) rotational matrix of the SE(3) state estimate, providing a 2D projection of the vehicle's forward facing side.

We can note that the orientation vectors remains aligned to the direction of travel. This tangential alignment indicates that the initial estimate of the poses respects the SE(3) pose kinematics (and there are no sudden oscillations or noise in the yaw).

However, it is important to note that since we did not use the map to correct our trajectory, it may be prone to drifting. The results presented here only rely on the EKF Predict step. Since we do not use a known map to correct our pose, the error keeps accumulating over time steps and is known as drift. For example, if one compares this trajectory to the `.mp4` video of the motion, we can note that the robot does make some sudden movements (e.g. dodging a pole, going around a walking person, etc.), which get smoothed out in these visual inertial odometry result.

### B. Landmark Mapping Results

We present our mapping results in Figures 4 and 5. The solid red line represents the robot's state trajectory (the IMU poses $T_{0:t}$), which is provided as a known, fixed input.The surrounding blue dots represent the estimated 3D coordinates of the static landmarks $\mathbf{m} \in \mathbb{R}^{3M}$. Because here we assumed that the landmarks are static, the mapping process bypasses

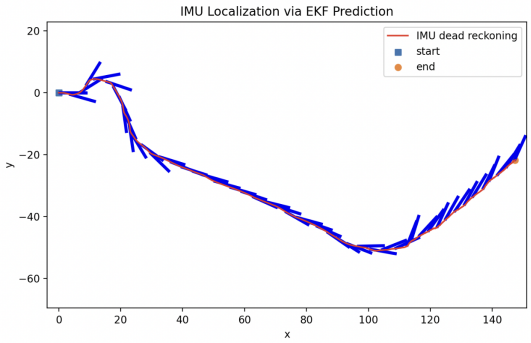



Fig. 2. Visualization of Robot Pose trajectory ($T_{0:T}$) over time in the world frame. Using **Dataset 00**

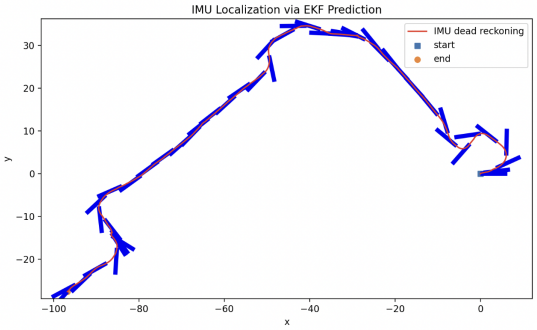



Fig. 3. Visualization of Robot Pose trajectory ($T_{0:T}$) over time in the world frame. Using **Dataset 01**

any motion model or prediction step. Instead, the map is constructed entirely through iterative EKF updates.

For this mapping, we removed landmarks that are too far from the camera using a minimum disparity threshold. Disparity is the difference in horizontal pixel location of the same landmark between the left and right images, and is related to the depth by $Z = \frac{fb}{d}$, where $f$ is the focal length, $b$ is the distance between the cameras and $d = |u_L - u_R|$ is the disparity. If this number is very small, it means that the point is very far (and could be noisy). We **empirically** found better results when we filtered out far-away points, as it helped us ensure that the EKF update step only incorporates well triangulated and proximal landmarks.

### C. Visual Inertial SLAM Results

In Figures 8 and 9, we present the results from our visual inertial SLAM algorithm. We have projected our 3D poses and 3D landmark positions into $x, y$ coordinates for visualization. The red dashed line represents the IMU odometry, the blue solid line represents the fully corrected VI-SLAM trajectory, and the green dots represent the final estimated 3D positions of the mapped landmarks ($\mathbf{m}$).

In the case of both the images, we see that initially, the odometry and corrected VISLAM trajectories agree, but they start diverging as the drift from odometry keeps accumulating towards the end of the trajectory. We can note that the

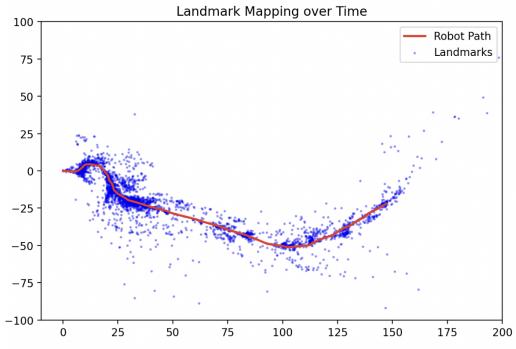



Fig. 4. Visualization of mapped landmarks using the EKF Update Step on **Dataset 00**

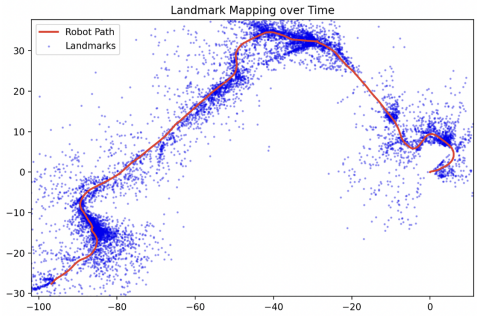



Fig. 5. Visualization of mapped landmarks using the EKF Update Step on **Dataset 01**

corrections from the SLAM algorithm pull the blue trajectory closer to the clusters of landmarks. This indicates that the SLAM algorithm is correcting the pose over time. In particular, we see a small bump in the trajectory for Dataset 1 ( 8), indicating the point where the robot dodges a street light, before reaching the open intersection and turning. This can be seen in the video at timestamp 00:45. Dataset 01 ( 9), we see that the landmarks have shifted away from the corrected trajectory. We suspect that this could be due to noisy features. In the beginning of the video for Dataset 01, certain moving points on a person were identified as features, towards the middle we noticed noisy features on the ground due to many fallen leaves, and towards the end, there was low lighting at the entrance of a building. We suspect these may have contributed to the slightly worse performance.

In addition, we observe that the starting points of both, the red and the blue trajectories, agree but the end points don't agree due to odometry drift.

### *D. **Result Videos***

In this section, we present videos demonstrating our SLAM over-time.

- VISLAM on Dataset 00: https://youtu.be/PSY-kURmLVM
- VISLAM on Dataset 01: https://youtu.be/-57VRfxNehM

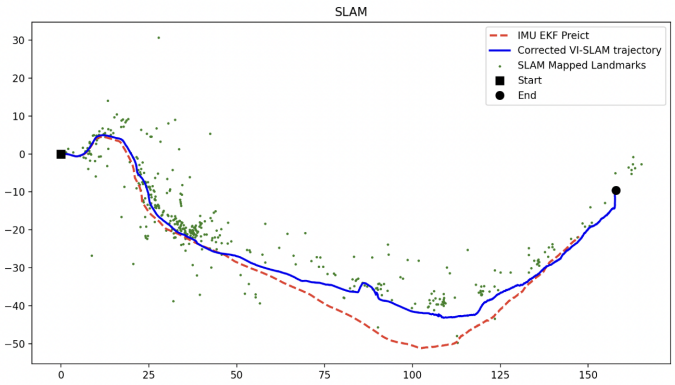



Fig. 6. Visualization of the final result from our Visual Inertial SLAM Algorithm on **Dataset 00**. The green dots represent the landmarks $m$, the blue line is the simultaneously estimated trajectory and the red line is the result from visual odometry.

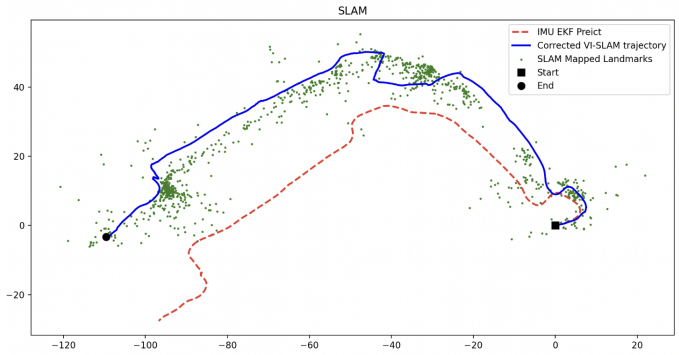



Fig. 7. Visualization of the final result from our Visual Inertial SLAM Algorithm on **Dataset 01**. The green dots represent the landmarks $m$, the blue line is the simultaneously estimated trajectory and the red line is the result from visual odometry.

## V. Extra Credit Feature Detection and Matching

### *A. Problem Formulation*

Corresponding points in two camera views are 2D projections of the same 3D point in space. The correspondence problem is to establish establish which point $\mathbf{z_2} \in \mathbb{R}^2$ in the second image $I_2$ corresponds to a given point $\mathbf{z_1} \in \mathbb{R}^2$ in the first image $I_1$ in the sense of being the same point $\mathbf{m} \in \mathbb{R}^3$ in 3D space.

Features are defined as points whose local window regions are rich enough to allow solving the correspondence problem. They establish a link between photometric measurements and geometric primitives.Our problem in this section is to detect such feature points and match them across frames.

For each timestep $t$, we create an observation matrix $\mathbf{z}_t \in \mathbb{R}^{4\times M}$, where $M$ is the total number of unique features in all of the images and each column corresponds to a specific

feature.

$$\mathbf{z}_t = \begin{bmatrix} l_{x,1} & l_{x,2} & l_{x,3} & \dots & l_{x,M} \\ l_{y,1} & l_{y,2} & l_{y,3} & \dots & l_{y,M} \\ r_{x,1} & r_{x,2} & r_{x,3} & \dots & r_{x,M} \\ r_{y,1} & r_{y,2} & r_{y,3} & \dots & r_{y,M} \end{bmatrix}$$

where:

- $l_{x,j}$ is the $x$-pixel-coordinate of the $j$-th feature of the left camera.
- $l_{y,j}$ is the $y$-pixel-coordinate of the $j$-th feature of the left camera.
- $r_{x,j}$ is the $x$-pixel-coordinate of the $j$-th feature of the right camera.
- $r_{y,j}$ is the $y$-pixel-coordinate of the $j$-th feature of the right camera.

Furthermore, we want to set any invalid features to $-1$.

### B. Technical Approach

We use OpenCV's implementations of a corner detector and optical flow to first detect the features and then track features that are common between the two frames across different time stamps. We do so as follows:

- Detection: Shi Tomasi corner detector uses the following scoring function:

  $$R = \min(\lambda_1, \lambda_2)$$

  A pixel is classified as a feature if $R$ exceeds a predefined quality threshold. This scoring function ensures we select points with high gradients in two orthogonal directions. In our implementation we maintain `TRACKED_FEATURES = 150` in each frame. If the number of features drops, we introduce new features. Note that we use the `goodFeaturesToTrack` implementation of this detector on `CV2`.
  - `maxCorners=300`. We return a max of 300 corners.
  - `qualityLevel=0.01`. This is a *relative* threshold. We reject any any corner whose score is less than 1% of that maximum score.
  - `minDistance=15`. This is the minimum Euclidean distance allowed between corners.
  - `blockSize=7`. The size of the pixel neighborhood used when looking for features.
- Once features are detected in the left frame at time $t$, their positions in the subsequent frame $t+1$ are estimated using the `calcOpticalFlowPyrLK()` method from `CV2`. We make assumptions such as the fact that the intensity of the pixels does not change significantly across frames, and that the pixels move together across frames.
- To recover depth, we must macth each point to its corresponding location in the right camera. To do this we call `calcOpticalFlowPyrLK()`, and run a status check to see filter out any invalid matches:
  - $u_L - u_R > 0$ (Depth should be positive)
  - $|v_L - v_R| < 3.0$ pixels (Vertical drift should be minimal).
  - Only features with a successful "status" flag from both temporal and stereo tracking steps are kept.

Each feature is assigned a global index in our full features dataset. As the robot moves, we maintain a `feature_history` matrix of shape $(4 \times M \times T)$, where $M$ is the total number of unique landmarks and $T$ is the number of timesteps. Once features go out of the field of view (as mentioned in 3.), we set them to -1.
Finally, we use these features in the same program as the previous sections to run VISLAM.

### C. Results

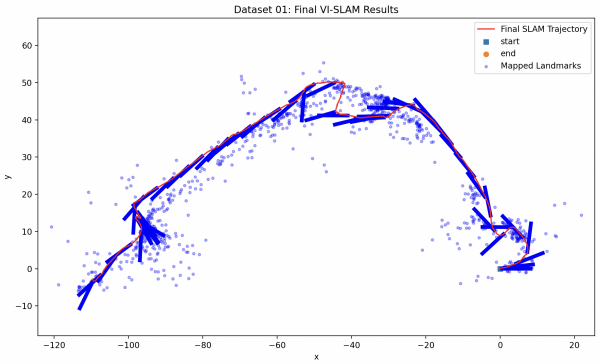



Fig. 8. Visualization of the final result from our Visual Inertial SLAM Algorithm on **Dataset 01**, using **tracked features**. The features may be noisy as we can see that the poses towards the middle of the trajectory have sharp turns that would not be physically possible. We suspect that this could be improved by playing with the optical flow and detection function parameters.

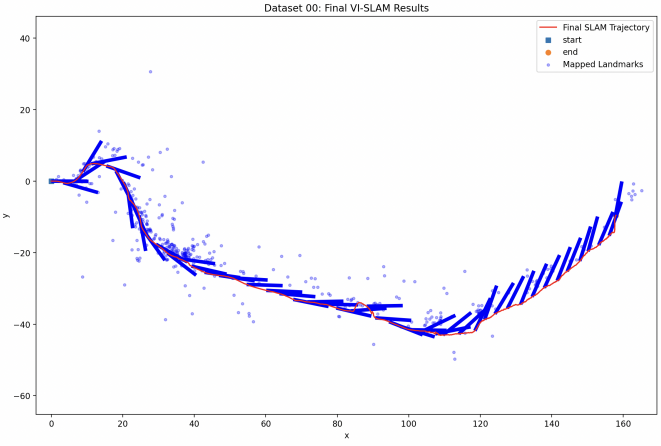



Fig. 9. Visualization of the final result from our Visual Inertial SLAM Algorithm on **Dataset 00**, using **tracked features**

## REFERENCES


[1] Localization and Odometry from Point Features Lecture: https://natanaso.github.io/ece276a/ref/ECE276A_6_LocalizationOdometry.pdf.
[2] Robot Motion and Observation Models Lecture: https://natanaso.github.io/ece276a/ref/ECE276A_4_MotionAndObservationModels.pdf.
[3] Extended and Unscented Kalman Filter Lecture: https://natanaso.github.io/ece276a/ref/ECE276A_10_EKF_UKF.pdf.
[4] Visual-Inertial SLAM Lecture: https://natanaso.github.io/ece276a/ref/ECE276A_12_VI_

[5] Matrix Lie Algebra Lecture: https://natanaso.github.io/ece276a/ref/ECE276A_11_MatrixLieGroups.pdf.
[6] SciPy CSR Documentation: https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html.
[7] SciPy LIL Documentation: https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.lil_matrix.html.
[8] Piazza:

- https://piazza.com/class/mhr7oh6rr7c3k8/post/268
- https://piazza.com/class/mhr7oh6rr7c3k8/post/268.
- https://piazza.com/class/mhr7oh6rr7c3k8/post/201

[9] Lucas-Kanade Optical Flow in OpenCV: https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html.
[10] OpenCV Shi-Tomasi Detector: https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html.