Visual-Inertial SLAM via Extended Kalman Filtering

Siddharth Rama Sushil
Dept. of Electrical and Computer Engineering
University of California, San Diego
San Diego, USA

***Abstract*—This report presents a visual-inertial simultaneous localization and mapping (SLAM) pipeline using an extended Kalman filter (EKF). We estimate the 6-DOF pose of a Clearpath Jackal robot from IMU velocity measurements, triangulate 3D landmark positions from stereo feature correspondences, and jointly refine both estimates via a tightly-coupled EKF on $SE(3)$. Extensive parameter tuning was performed to balance IMU and camera trust. For dataset 02 (extra credit), we implemented our own Shi-Tomasi corner detection and Lucas-Kanade optical flow pipeline.**

I. Introduction

Visual-inertial SLAM is a cornerstone capability for autonomous mobile robots, enabling simultaneous estimation of the robot's trajectory and a map of the environment without relying on external positioning infrastructure such as GPS. This is critical in GPS-denied environments where robots must rely on onboard sensors.

In this project, we address the visual-inertial SLAM problem using data from Clearpath Jackal robots navigating MIT's campus. The robot is equipped with an IMU providing linear and angular velocity measurements and a stereo camera providing pixel-level observations of visual landmarks. We implement EKF prediction from IMU data on $SE(3)$ kinematics and EKF updates from stereo-camera observations to perform SLAM across three datasets. For dataset 02, we additionally implement our own feature detection and tracking pipeline for extra credit.

II. Problem Formulation

A. State Representation

The robot's pose at time t is represented as a homogeneous transformation $T_t \in SE(3)$ from the IMU body frame to the world frame:

$$T_t = \begin{bmatrix} R_t & p_t \\ 0^\top & 1 \end{bmatrix} \in SE(3) \tag{1}$$

where $R_t \in SO(3)$ is the 3×3 rotation matrix encoding the orientation and $p_t \in \mathbb{R}^3$ is the position. The world frame is fixed at the robot's initial position and orientation at $t = 0$ (i.e., $T_0 = I_4$). The IMU frame follows the convention x = forward, y = left, z = up.

The environment contains M static point landmarks at unknown positions $\mathbf{m}_j \in \mathbb{R}^3$, $j = 1, \ldots, M$. The full SLAM state is conceptually:

$$\mathbf{x} = \left[\delta\boldsymbol{\xi}^\top, \mathbf{m}_1^\top, \ldots, \mathbf{m}_M^\top\right]^\top \in \mathbb{R}^{6+3M} \tag{2}$$

where $\delta\boldsymbol{\xi} \in \mathbb{R}^6$ is the pose perturbation in the Lie algebra $\mathfrak{se}(3)$, used because the pose T_t lives on a manifold and cannot be directly parameterized with additive Gaussian noise. The associated covariance matrix $\Sigma \in \mathbb{R}^{(6+3M)\times(6+3M)}$ encodes the joint uncertainty.

B. Data and Measurements

At each timestep t we receive:

- **IMU measurements:** Linear velocity $\mathbf{v}_t \in \mathbb{R}^3$ and angular velocity $\boldsymbol{\omega}_t \in \mathbb{R}^3$, both expressed in the body frame of the IMU. These are treated as control inputs driving the motion model.
- **Stereo observations:** $\mathbf{z}_t \in \mathbb{R}^{4\times M}$, where each column $\mathbf{z}_{t,j} = [u_L, v_L, u_R, v_R]^\top$ contains the pixel coordinates of landmark j in the left (u_L, v_L) and right (u_R, v_R) camera frames. Landmarks not observed at time t are marked $[-1, -1, -1, -1]^\top$.
- **Timestamps:** τ_t in UNIX time; the time step is $\Delta t = \tau_{t+1} - \tau_t$.
- **Intrinsic calibration:** Camera matrices $K_l, K_r \in \mathbb{R}^{3\times3}$ with focal lengths f_u, f_v and principal points c_u, c_v.
- **Extrinsic calibration:** Transformations ${}^I T_{C_L}$ and ${}^I T_{C_R}$ from each camera frame to the IMU frame. From these, we derive the camera-to-IMU transform ${}^C T_I = ({}^I T_C)^{-1}$ and the stereo baseline $b = \|p_{C_L} - p_{C_R}\|$ (the distance between the optical centers of left and right cameras).

C. Motion Model

The robot's motion is governed by $SE(3)$ kinematics. Given the body-frame twist $\boldsymbol{\xi}_t = [\mathbf{v}_t^\top, \boldsymbol{\omega}_t^\top]^\top \in \mathbb{R}^6$, the discrete-time propagation is:

$$T_{t+1} = T_t \cdot \exp\Big(\Delta t \cdot \hat{\boldsymbol{\xi}}_t\Big) \cdot \mathbf{w}_t \tag{3}$$

where $\hat{\boldsymbol{\xi}}_t \in \mathfrak{se}(3)$ is the 4×4 Lie algebra element and $\mathbf{w}_t \sim \mathcal{N}(0, W)$ represents additive process noise.

D. Observation Model

The stereo observation of landmark $\mathbf{m}_j$ from pose T_t follows:

$$\mathbf{z}_{t,j} = h(T_t, \mathbf{m}_j) + \mathbf{n}_{t,j}, \quad \mathbf{n}_{t,j} \sim \mathcal{N}(0, V) \tag{4}$$

where the observation function h projects the world-frame landmark through the camera:

$$h(T_t, \mathbf{m}_j) = K_s \cdot \pi\left({}^{C}T_I \cdot T_t^{-1} \cdot \begin{bmatrix}\mathbf{m}_j \\ 1\end{bmatrix}\right) \tag{5}$$

Here $\pi(\mathbf{q}) = [q_1/q_3,\ q_2/q_3,\ q_1/q_3,\ q_2/q_3]^\top$ is the stereo projection and $K_s \in \mathbb{R}^{4\times4}$ is the stereo calibration matrix incorporating both cameras and the baseline.

E. Desired Output

1) The estimated IMU trajectory $\{T_t\}_{t=0}^{T}$ in $SE(3)$.
2) The estimated 3D landmark positions $\{\mathbf{m}_j\}_{j=1}^{M}$.
3) The associated covariance Σ encoding pose and landmark uncertainty.

III. Technical Approach

A. Task 1: IMU Localization via EKF Prediction

1) Mathematical Formulation

The pose is propagated using the $SE(3)$ exponential map:

$$T_{t+1} = T_t \cdot \exp(\Delta t \cdot \hat{\boldsymbol{\xi}}_t) \tag{6}$$

The twist $\boldsymbol{\xi}_t = [\mathbf{v}_t^\top, \boldsymbol{\omega}_t^\top]^\top$ is constructed from the IMU measurements. The 4×4 Lie algebra element is:

$$\hat{\boldsymbol{\xi}} = \begin{bmatrix}[\boldsymbol{\omega}]_\times & \mathbf{v} \\ 0 & 0\end{bmatrix} \in \mathfrak{se}(3) \tag{7}$$

The matrix exponential is computed via the Rodrigues formula.

The covariance is propagated as:

$$\Sigma_{t+1} = F_t\, \Sigma_t\, F_t^\top + \Delta t\, W \tag{8}$$

where $F_t = \exp(-\Delta t \cdot \mathrm{ad}(\boldsymbol{\xi}_t))$ uses the 6×6 adjoint representation:

$$\mathrm{ad}(\boldsymbol{\xi}) = \begin{bmatrix}[\boldsymbol{\omega}]_\times & [\mathbf{v}]_\times \\ \mathbf{0} & [\boldsymbol{\omega}]_\times\end{bmatrix} \tag{9}$$

2) Implementation Details

The prediction loop iterates over all T timesteps. At each step, we compute Δt from consecutive timestamps, construct the 6×1 twist vector, and compute $\exp(\Delta t \cdot \hat{\boldsymbol{\xi}})$ via the provided `twist2pose` utility (Rodrigues formula). The pose is updated via right-multiplication, and the covariance is propagated using the matrix exponential of the 6×6 adjoint (computed via SciPy's `expm`). The process noise $W = \mathrm{diag}(W_v, W_v, W_v, W_\omega, W_\omega, W_\omega)$ has separate scalars for linear and angular velocity, allowing independent tuning of translation vs. rotation trust.

B. Task 3: Landmark Mapping via EKF Update

1) Stereo Triangulation

When a landmark is first observed, its 3D position is initialized via stereo triangulation. Given disparity $d = u_L - u_R > 0$:

$$z = f_u \cdot b/d, \quad x = (u_L - c_u) \cdot z/f_u, \quad y = (v_L - c_v) \cdot z/f_v \tag{10}$$

The camera-frame point $\mathbf{p}_{\text{cam}} = [x, y, z]^\top$ is transformed to world frame: $\mathbf{m}_j = T_t \cdot ({}^{I}T_C)^{-1} \cdot [\mathbf{p}_{\text{cam}}^\top, 1]^\top$.

2) Feature Filtering

Not all features in the dataset are useful. We apply the following filters:

1) **Validity:** Features marked $[-1, -1, -1, -1]$ are skipped.
2) **Disparity bounds:** Only features with $d_{\min} \le d \le d_{\max}$ are used. Features with very small disparity ($< d_{\min}$) are extremely far and have unreliable triangulated depth. Features with very large disparity ($> d_{\max}$) lie too close (potentially on the robot body). See Table I for values.
3) **Subsampling:** If more than `max_features` pass filtering, a random subset is selected to cap computational cost.
4) **Depth check:** During updates, features that project behind the camera ($q_z < 0.1$) are excluded.

3) EKF Update for Landmarks

Each landmark j has an independent 3×3 covariance Σ_{jj}, initialized to $\sigma_{\text{init}}^2 \cdot I_3$. The observation Jacobian with respect to the landmark position is:

$$H_j = K_s \cdot \frac{\partial \pi}{\partial \mathbf{q}} \cdot {}^{C}T_W\,[:, 1{:}3] \tag{11}$$

where ${}^{C}T_W = {}^{C}T_I \cdot T_t^{-1}$ transforms from world to camera frame and $\frac{\partial \pi}{\partial \mathbf{q}}$ is the 4×4 projection Jacobian. The EKF update follows:

$$S_j = H_j \Sigma_{jj} H_j^\top + V \tag{12}$$

$$K_j = \Sigma_{jj} H_j^\top S_j^{-1} \tag{13}$$

$$\mathbf{m}_j \leftarrow \mathbf{m}_j + K_j(\mathbf{z}_{t,j} - \hat{\mathbf{z}}_{t,j}) \tag{14}$$

We use the **Joseph form** for the covariance update: $\Sigma_{jj} \leftarrow (I - K_j H_j)\Sigma_{jj}(I - K_j H_j)^\top + K_j V K_j^\top$, which guarantees numerical positive semi-definiteness.

4) Implementation Details

The mapping loop iterates over all timesteps, using the pre-computed IMU poses from Task 1. At each timestep, valid features are extracted and filtered by disparity bounds. New landmarks are initialized via stereo triangulation. For each previously-initialized landmark, the observation is projected using the current IMU pose, the innovation is computed, and the independent 3×3 EKF update is applied. The independent per-landmark covariance blocks keep the update cost $O(M)$ instead of $O(M^3)$.

C. Task 4: Visual-Inertial SLAM

1) Joint Covariance Structure

The covariance is maintained in block form:

$$\Sigma = \begin{bmatrix} \Sigma_{pp} & \Sigma_{pm} \\ \Sigma_{mp} & \Sigma_{mm} \end{bmatrix} \tag{15}$$

where $\Sigma_{pp} \in \mathbb{R}^{6\times 6}$ is the pose covariance, Σ_{mm} is block-diagonal with 3×3 per-landmark blocks, and $\Sigma_{pm} \in \mathbb{R}^{6\times M\times 3}$ stores pose-landmark cross-covariances. In our implementation, Σ_{pm} is stored as a 3D array indexed by landmark ID, avoiding the memory cost of a dense $(6+3M)\times(6+3M)$ matrix.

2) Prediction Step

Same as Task 1:

$$\begin{aligned} \Sigma_{pp}^{-} &= F_t \Sigma_{pp}^{+} F_t^\top + \Delta t\, W \\ \Sigma_{pm}^{-} &= F_t\, \Sigma_{pm}^{+} \end{aligned} \tag{16}$$

Landmark covariances Σ_{mm} are unchanged during prediction.

3) Update Step with Pose and Landmark Jacobians

For each observed landmark j, the full innovation covariance is:

$$S = H_p \Sigma_{pp} H_p^\top + H_p \Sigma_{pj} H_m^\top + H_m \Sigma_{pj}^\top H_p^\top + H_m \Sigma_{jj} H_m^\top + V \tag{17}$$

The landmark Jacobian H_m is identical to Task 3. The **pose Jacobian** $H_p \in \mathbb{R}^{4\times 6}$ uses the circle-dot operator, differentiating the observation w.r.t. a right perturbation of the pose:

$$H_p = K_s \cdot \frac{\partial \pi}{\partial \mathbf{q}} \cdot {}^{C}T_I \cdot (-\mathbf{p}^{\odot}) \tag{18}$$

where $\mathbf{p}^{\odot} \in \mathbb{R}^{4\times 6}$ is the "circle-dot" operator applied to the point $\mathbf{p}$ in the IMU frame:

$$\mathbf{p}^{\odot} = \begin{bmatrix} I_3 & -[\mathbf{p}_{1:3}]_\times \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} \end{bmatrix} \tag{19}$$

The Kalman gains for pose and landmark are:

$$K_p = (\Sigma_{pp} H_p^\top + \Sigma_{pj} H_m^\top) S^{-1} \tag{20}$$

$$K_m = (\Sigma_{pj}^\top H_p^\top + \Sigma_{jj} H_m^\top) S^{-1} \tag{21}$$

The pose correction $\delta\boldsymbol{\xi} = K_p \mathbf{e}$ is applied via right-multiplication on $SE(3)$: $T \leftarrow T \cdot \exp(\widehat{\delta\boldsymbol{\xi}})$. The landmark is updated additively: $\mathbf{m}_j \leftarrow \mathbf{m}_j + K_m \mathbf{e}$.

4) Covariance Update and Numerical Stability

Our initial implementation used the standard subtraction form $\Sigma_{pp} \leftarrow \Sigma_{pp} - K_p S K_p^\top$. When processing $\sim$200 landmarks *sequentially* per timestep, accumulated floating-point errors caused Σ_{pp} to develop **negative eigenvalues**. This triggered a cascading failure: incorrect Kalman gains produced negative Mahalanobis distances, which bypassed outlier rejection (our original check $d > 50$ failed to catch $d = -174$), and the filter diverged with innovations reaching 70,000 px.

The fix involved three changes:

1) **Joseph-form covariance update:**

$$\Sigma_{pp} \leftarrow (I - K_p H_p)\Sigma_{pp}(I - K_p H_p)^\top + K_p V K_p^\top \tag{22}$$

2) **Negative Mahalanobis rejection:** $\mathbf{e}^\top S^{-1}\mathbf{e} < 0 \Rightarrow$ reject.
3) **Eigenvalue clamping:** After all updates, eigenvalues of Σ_{pp} below 10^{-6} are clamped.

The Jacobians were verified via numerical differentiation (maximum error < 0.0004), confirming the bug was purely in the covariance arithmetic.

5) Outlier Rejection

Before applying each update, landmarks are rejected if:

- Mahalanobis distance $\mathbf{e}^\top S^{-1}\mathbf{e} > \theta_{\text{mahal}}$ or < 0
- Innovation norm $\|\mathbf{e}\| > \theta_{\text{innov}}$ pixels
- Negative depth $q_z < 0.1$ (behind camera)

6) Implementation Details

The SLAM loop follows Algorithm 1.

Algorithm 1 Visual-Inertial SLAM

```
Initialize T_0 = I_4, Σ_pp = 10^{-2} I_6
for t = 0 to T − 1 do
    Predict: T_{t+1} = T_t · exp(Δt · ξ̂_t)
    Update Σ_pp, Σ_pm via Eqs. (17)
    Get valid features with disparity in [d_min, d_max]
    Subsample to at most max_features
    Initialize new landmarks via stereo triangulation
    for each previously-seen landmark j do
        Project: ẑ = K_s · π(^C T_I · T^{-1} · m̃_j)
        Innovation: e = z_j − ẑ
        Compute S, check outlier gates
        Compute K_p, K_m; apply Joseph-form update
        T ← T · exp(K_p e); m_j += K_m e
    end for
    Eigenvalue-clamp Σ_pp
end for
```

The cross-covariance Σ_{pm} is updated in two parts: for each observed landmark j, the exact update is applied: $\Sigma_{pj}^{+} = (I - K_p H_p)\Sigma_{pj}^{-} - K_p H_m \Sigma_{jj}$. For unobserved landmarks, the accumulated product $\prod_i (I - K_p^{(i)} H_p^{(i)})$ is applied via a single matrix multiplication using `einsum`.

7) Parameter Tuning

Extensive tuning was required to balance IMU and camera trust. The key challenge is that without loop closure, even small systematic biases in the stereo observations accumulate over thousands of sequential EKF updates, causing the SLAM trajectory to drift from the IMU trajectory.

Tuning process: We started with $V = 50 \cdot I_4$ (high camera trust), which produced severe drift—the SLAM trajectory jumped ± 400 m from the IMU path. Increasing to $V = 200 \cdot I_4$ improved results, but the SLAM path still diverged. Interestingly, very high values ($V = 1000 \cdot I_4$) performed *worse*

because landmarks failed to converge, leading to accumulated bias from uncertain positions.

The optimal configuration was found by iteratively adjusting:

1) Reducing features per timestep from 200 to 25 (dramatic reduction in cumulative bias).
2) Using moderate observation noise ($V = 400 \cdot I_4$).
3) Tightening outlier thresholds ($\theta_{\text{innov}} = 20\,\text{px}$, $\theta_{\text{mahal}} = 15$).
4) Increasing minimum disparity ($d_{\min} = 15\,\text{px}$) to exclude distant landmarks.
5) Reducing process noise for linear velocity ($W_v = 5 \times 10^{-4}$) to tighten the IMU prediction.

TABLE I. Final EKF Parameters

Parameter	Value	Role
W_v (linear vel.)	5×10^{-4}	Process noise (translation)
W_ω (angular vel.)	1×10^{-4}	Process noise (rotation)
V (observation)	$400 \cdot I_4$	Stereo pixel noise
Max features/step	25	Limits cumulative correction
Min disparity $d_{\min}$	15 px	Filters distant landmarks
Max disparity $d_{\max}$	150 px	Filters near landmarks
θ_{mahal}	15	Mahalanobis outlier gate
θ_{innov}	20 px	Innovation norm gate
Σ_{init}	$30 \cdot I_3$	Initial landmark covariance

D. Task 2: Feature Detection and Matching (Extra Credit)

For dataset 02, which does not provide pre-computed feature tracks, we implemented a complete detection and tracking pipeline. This required significant tuning to produce features of sufficient quality for SLAM.

1) Stereo Matching

At each timestep, Shi-Tomasi corner detection (`goodFeaturesToTrack` from OpenCV) identifies salient features in the left image. These features are matched to the right image using Lucas-Kanade optical flow (`calcOpticalFlowPyrLK`) with a 21×21 window and 3 pyramid levels. An epipolar constraint ($|v_L - v_R| < \epsilon_{\text{epi}}$ pixels) filters out geometrically inconsistent matches, since the stereo cameras are approximately horizontally aligned.

2) Temporal Tracking

Features are tracked across time using **forward-backward optical flow consistency**. The forward tracker computes $\mathbf{p}_{t+1}$ from $\mathbf{p}_t$; a backward pass recomputes $\hat{\mathbf{p}}_t$ from $\mathbf{p}_{t+1}$. Only features with forward-backward error $\|\mathbf{p}_t - \hat{\mathbf{p}}_t\| < \epsilon_{\text{fb}}$ are retained. This bidirectional validation reduces tracking drift.

New features are detected every `detect_interval` frames or whenever the active count drops below half the target. Each feature receives a unique global ID.

3) Tuning the Feature Pipeline

Initial attempts produced a very large number of feature matches (up to 2000 per frame), many of which were poor: matched across background sky, repetitive textures, or with large vertical offsets indicating incorrect correspondences. These noisy features degraded SLAM performance.

The tuning process involved:

1) **Quality threshold** (`qualityLevel`): Increased from 0.005 to 0.01 to require stronger corners, reducing spurious detections on low-texture surfaces.
2) **Minimum distance** (`minDistance`): Set to 10 px to prevent clusters of adjacent features that add redundant information.
3) **Epipolar constraint** (ϵ_{epi}): Tightened from 5 px to 3 px. The stereo rig is not perfectly horizontal (as noted in the), so some vertical offset is expected, but allowing too much admits false matches.
4) **Forward-backward threshold** (ϵ_{fb}): Set to 2 px. Tracks failing bidirectional consistency are lost features being "hallucinated" by the tracker.
5) **Maximum corners**: Capped at 500 per frame to balance coverage vs. quality.
6) **Detection interval**: Every 5 frames, preventing excessive re-detection that disrupts existing tracks.

TABLE II. Feature Detection Parameters

Parameter	Value	Role
Max corners	500	Features detected per frame
Quality level	0.01	Shi-Tomasi strength threshold
Min distance	10 px	Spacing between detections
LK window	21×21	Optical flow search window
Pyramid levels	3	Multi-scale tracking
Epipolar tol. ϵ_{epi}	3 px	Stereo vertical consistency
FB error tol. ϵ_{fb}	2 px	Temporal tracking consistency
Detect interval	5 frames	Re-detection frequency

4) Feature Matrix Assembly

The output is a $4 \times M \times T$ matrix in the same format as the provided datasets, where M is the total number of distinct features ($M \approx 46{,}000$). Each column at time t contains $[u_L, v_L, u_R, v_R]^\top$ if the feature is visible, or $[-1, -1, -1, -1]^\top$ otherwise.

IV. Results

We present results for datasets 00, 01, and 02. For dataset 02, we ran two configurations: with the provided features and with self-detected features (extra credit). Unless otherwise noted, all dataset 02 results use the **provided features**; the extra credit results are discussed separately in Section IV-F.

A. SLAM Trajectories and Landmark Maps

Fig. 1–3 show the SLAM landmark maps for each dataset: the IMU-only trajectory (orange), the SLAM-corrected trajectory (blue), and estimated landmarks (gray dots).

Across all datasets, the SLAM trajectory applies visible corrections to the IMU path, particularly in curved sections where heading drift has the largest positional impact. The

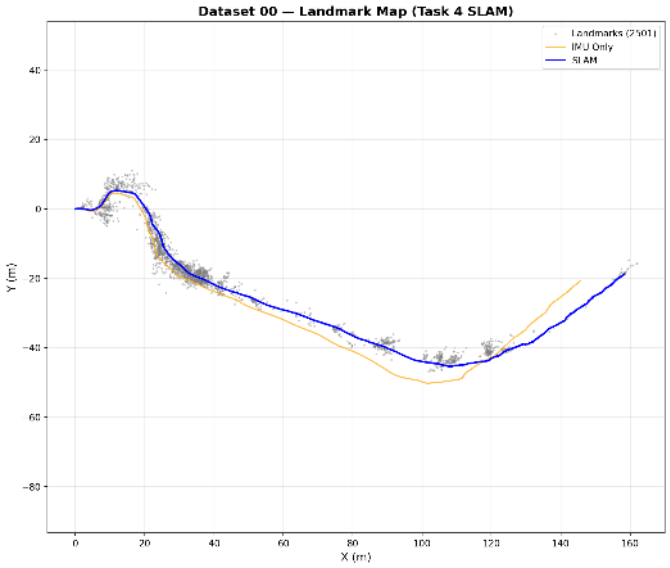


Fig. 1. Dataset 00: SLAM landmark map. 2501 landmarks mapped along a rectangular loop spanning $\sim$160 m. The SLAM (blue) and IMU (orange) trajectories diverge after $x \approx 60$ m, with the IMU drifting lower in y. Landmarks tightly trace the trajectory.

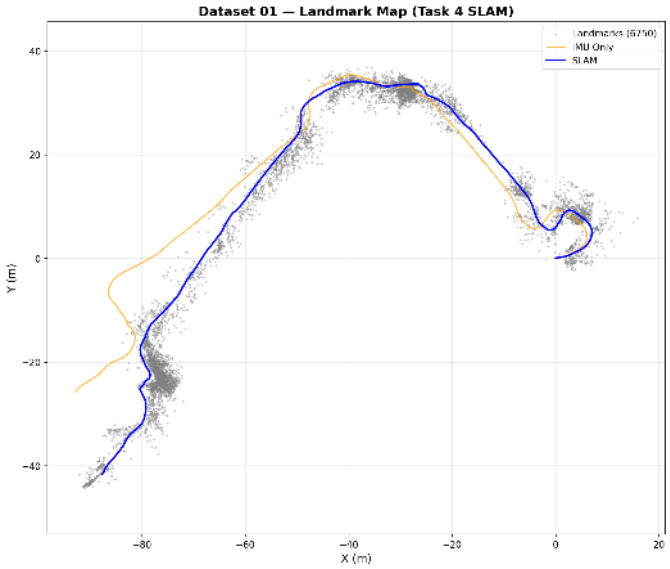


Fig. 2. Dataset 01: SLAM landmark map. Largest dataset with 6750 landmarks along a wide loop. The IMU trajectory drifts outward by $\sim$10–15 m between $x = -40$ and $x = -80$. The SLAM trajectory corrects back toward the landmark cluster centers.

landmarks cluster tightly along the SLAM trajectory, consistent with the forward-facing stereo camera's limited field of view.

B. Feature Count per Timestep

The feature count reveals the availability of usable landmarks at each timestep. Dataset 00 has several extended gaps where no features pass the disparity filter, corresponding to rapid turns or open areas. Dataset 01 maintains the most consistent feature supply. Dataset 02 is sparsest overall, with strict

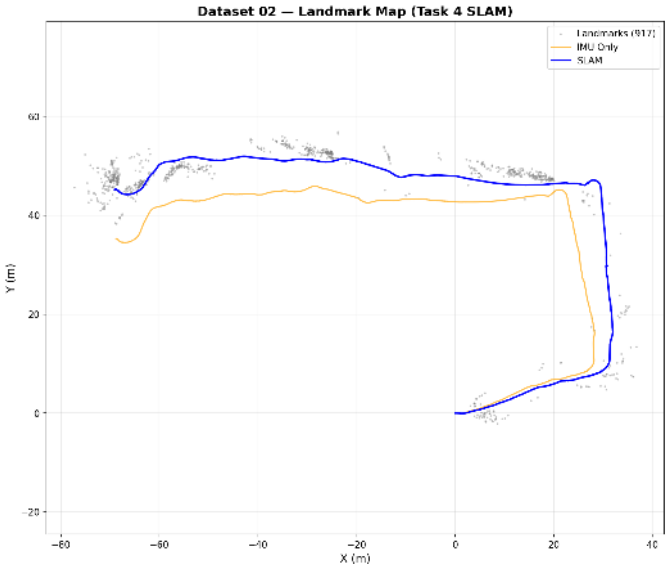


Fig. 3. Dataset 02 (provided features): Sparser map with 917 landmarks after strict disparity filtering. The IMU path reaches $y \approx 45$ m while the SLAM path extends to $y \approx 52$ m.

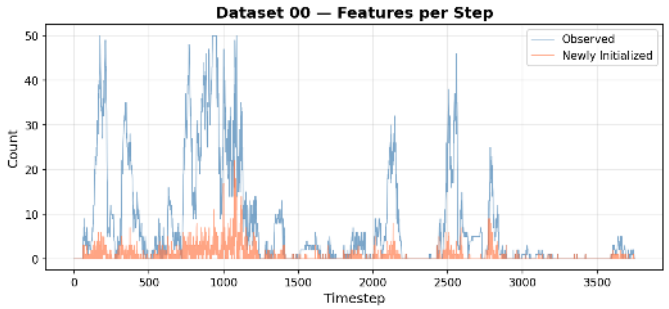


Fig. 4. Dataset 00: Feature count. With our cap of 25, the count stays bounded. Gaps around timesteps 500–600 and 1500–1800 correspond to feature-poor regions.

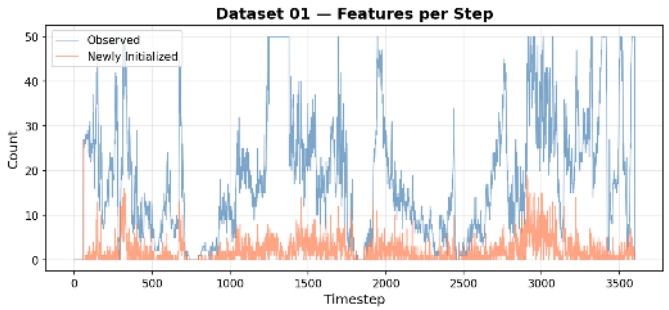


Fig. 5. Dataset 01: Feature count. More consistent availability than dataset 00 due to the campus walk scene's rich texture.

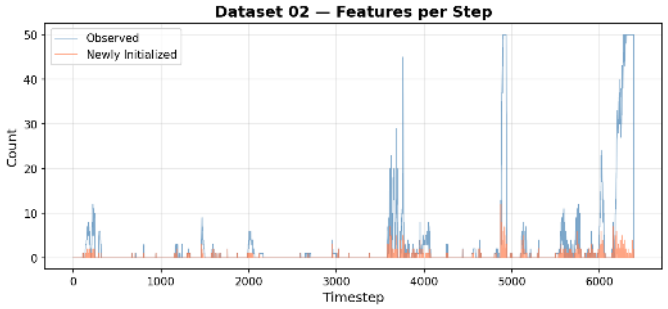


Fig. 6. Dataset 02: Feature count. Sparsest of the three; many timesteps have zero valid features after disparity filtering.

disparity filtering eliminating many candidates. In all datasets, re-observed landmarks dominate over new initializations, indicating effective feature persistence.

C. Pose Covariance Trace

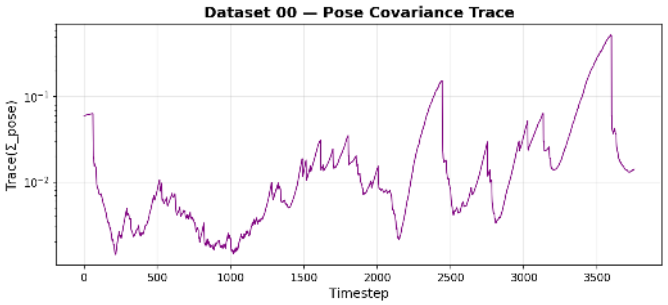


Fig. 7. Dataset 00: Covariance trace. Sawtooth pattern: grows during prediction, drops during updates. Spike near $t = 3500$ corresponds to the feature gap in Fig. 4.

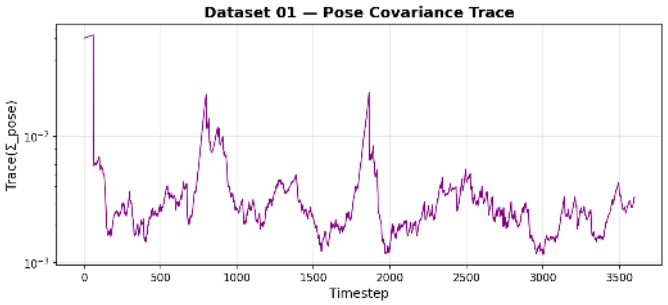


Fig. 8. Dataset 01: Smoothest covariance profile due to consistent feature availability.

The covariance trace $\mathrm{tr}(\Sigma_{pp})$ exhibits a characteristic **sawtooth** pattern across all datasets: it grows during prediction steps and drops during updates. The covariance stays bounded between $\sim 10^{-3}$ and $\sim 10^{-1}$, confirming the Joseph-form update maintains stability. Dataset 00 shows a notable spike near $t = 3500$ (during the feature gap), where unchecked prediction growth reaches ~ 0.5 before features are re-acquired.

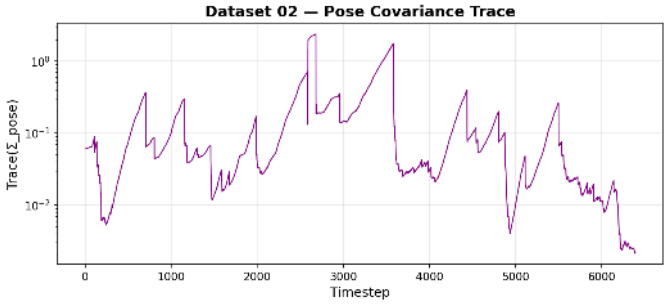


Fig. 9. Dataset 02: Most volatile covariance due to sparse features.

Dataset 01 has the smoothest profile. Dataset 02 has the most volatile trace, reflecting its sporadic feature availability.

D. Innovation Magnitude

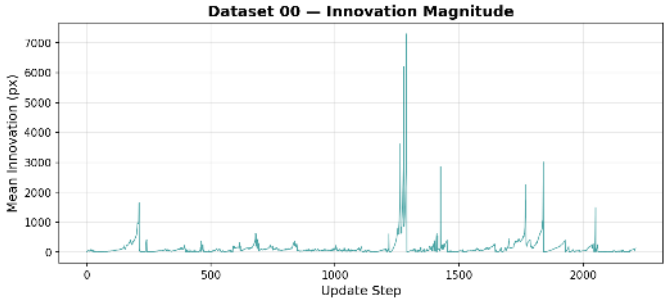


Fig. 10. Dataset 00: Mean innovation. Mostly low with spikes at update step $\sim$1300 (re-observing after a blind period).

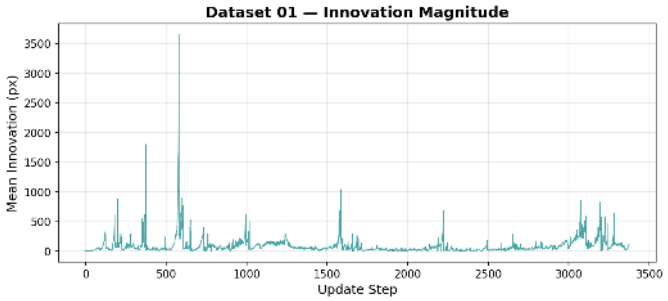


Fig. 11. Dataset 01: Most consistent innovations, reflecting stable feature tracking.

Innovation magnitude indicates how well predicted observations match actual measurements. For most timesteps, mean innovations stay low. Dataset 00 has a prominent spike at update step $\sim$1300 (exceeding several thousand pixels), corresponding to the moment after the long feature gap when re-observed landmarks carry large residuals. Our tight outlier gate ($\theta_{\text{innov}} = 20\,\text{px}$) prevents these from corrupting the state. Dataset 01 has the most consistent innovation profile. Dataset 02 shows higher baseline innovation.

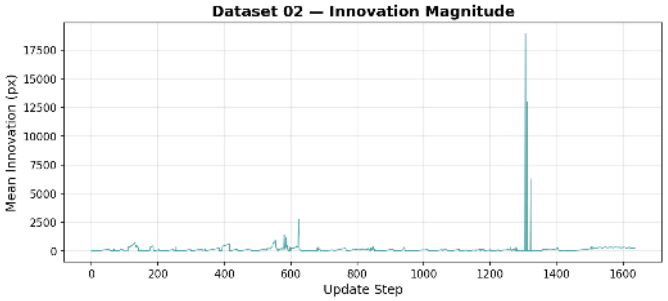


Fig. 12. Dataset 02: Higher baseline innovation due to sparser feature set.

E. Timing Analysis

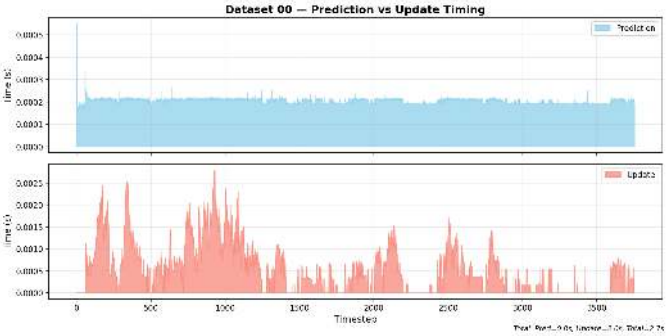


Fig. 13. Dataset 00: Per-timestep computation. Prediction is < 1 ms; update is the bottleneck.

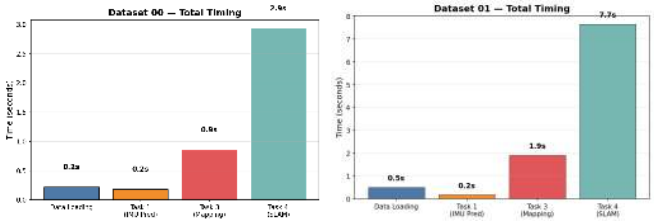


Fig. 14. Total timing breakdown for datasets 00 (left) and 01 (right). Data loading dominates.

The prediction step is consistently fast (< 1 ms) as it involves only a single matrix exponential. The update step cost scales with the number of observed landmarks. With our cap of 25 features, total per-step time stays under $\sim$5 ms. Data loading dominates wall-clock time due to the large `.npy` files.

F. Extra Credit: Self-Detected Features on Dataset 02

For dataset 02, we implemented our own Shi-Tomasi + Lucas-Kanade feature pipeline and ran the full SLAM pipeline. Fig. 15–18 show the results.

With self-detected features, the landmark count more than doubled (2023 vs. 917), producing a denser map that more completely traces the path. However, the SLAM trajectory diverges further from the IMU path ($y \approx 68$ m vs. 52 m). This is

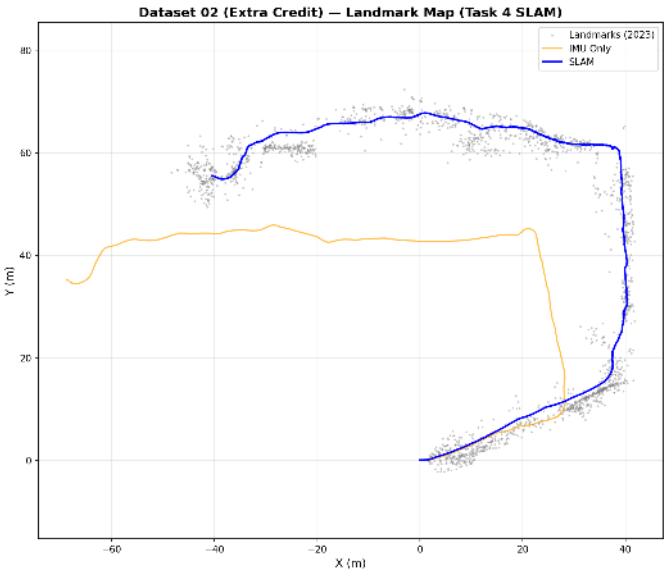


Fig. 15. Extra credit: SLAM landmark map with self-detected features. 2023 landmarks (vs. 917 with provided features). The SLAM trajectory diverges further from IMU, reaching $y \approx 68$ m.

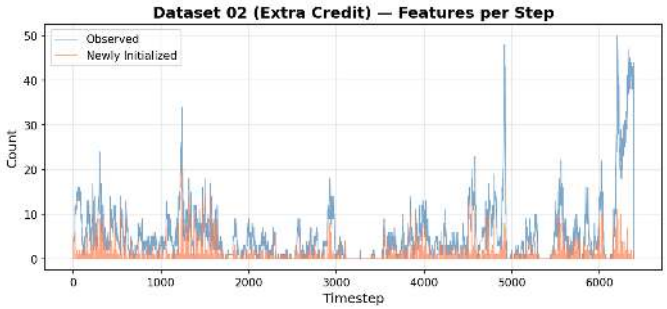


Fig. 16. Extra credit: Feature count showing the periodic re-detection pattern (every 5 frames) and declining tracks between detections.

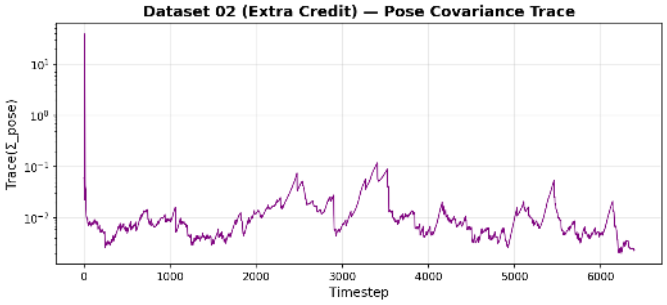


Fig. 17. Extra credit: Covariance trace. More volatile than the provided-features run.

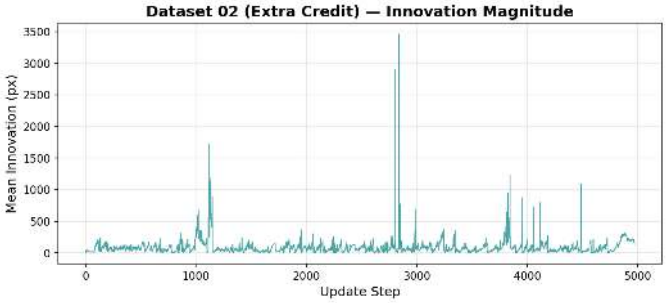


Fig. 18. Extra credit: Innovation magnitude. Higher baseline than provided features.

because the optical flow tracker introduces small but systematic pixel-level biases that accumulate over 6393 timesteps, and short-lived tracks with less temporal consistency add noise. The feature count profile shows the characteristic "sawtooth" pattern of periodic re-detection. The covariance trace is more volatile and innovations are generally higher, reflecting noisier feature quality.

We tuned the feature detection pipeline by tightening the epipolar constraint (from 5 to 3 px), requiring forward-backward consistency (2 px threshold), and increasing the quality threshold to filter weak corners. These changes reduced the number of bad matches but could not fully eliminate the systematic biases inherent in optical flow tracking.

1) Stereo Match Visualizations

Fig. 19 shows stereo feature matches at 5 sample frames, validating that our matches are predominantly horizontal (as expected for a roughly horizontal stereo rig). Feature density varies with scene content: structured urban scenes produce more reliable matches than open areas.

V. Conclusion

This project successfully demonstrated the implementation of a tightly-coupled visual-inertial SLAM pipeline using an Extended Kalman Filter on the $SE(3)$ manifold. By fusing high-frequency IMU kinematics with stereo-vision landmark observations, we effectively estimated the 6-DOF pose of a mobile robot while concurrently building a sparse 3D map of its environment across multiple datasets.

A major takeaway from this implementation is the extreme sensitivity of the EKF to both numerical instability and systematic measurement errors. Processing high volumes of sequential landmark updates revealed that standard covariance subtraction quickly degenerates. Applying the Joseph-form update, strict Mahalanobis outlier gating, and eigenvalue clamping proved strictly necessary to maintain positive semi-definiteness and prevent filter divergence. Furthermore, extensive parameter tuning was required to safely balance the trust between the IMU's dead-reckoning and the camera's visual corrections.

Finally, the integration of a custom Shi-Tomasi and Lucas-Kanade feature tracking pipeline for the extra credit dataset highlighted the practical limitations of filtering-based SLAM.

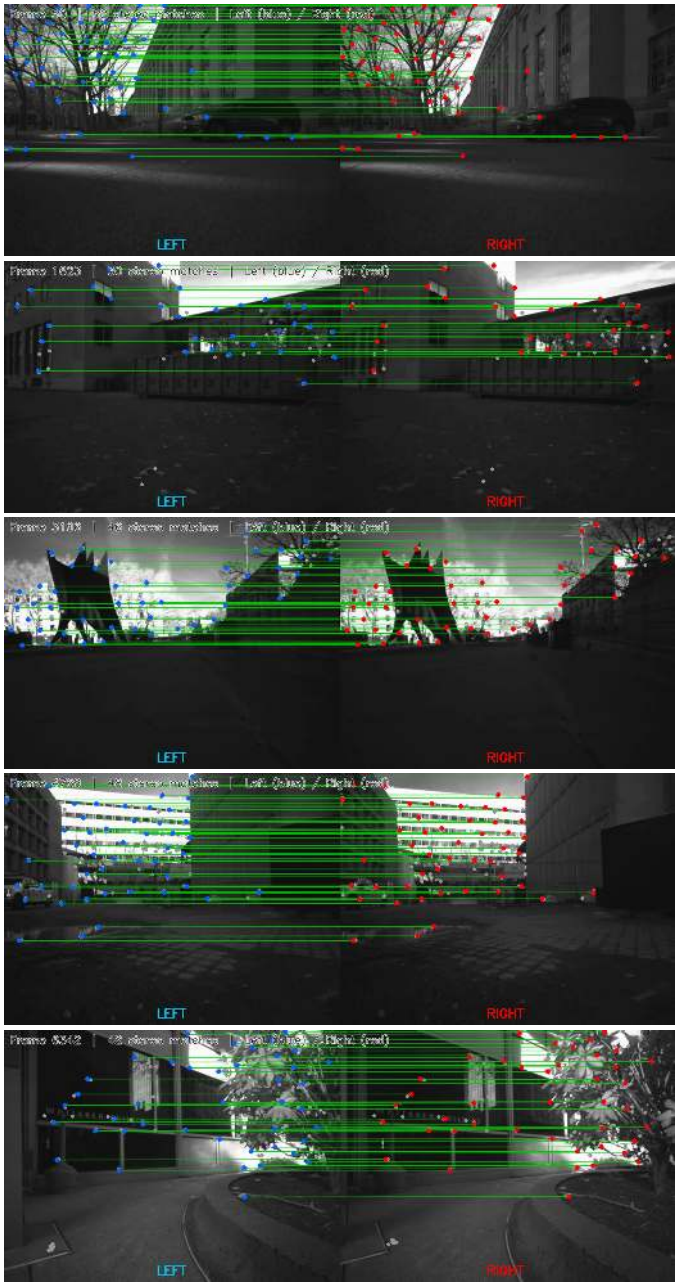


Fig. 19. Stereo feature matches at 5 sample frames from dataset 02. Blue: left image features. Red: right image matches. Green lines connect correspondences. The predominantly horizontal match lines validate the epipolar filtering.

Without global loop closure, even minor sub-pixel tracking biases inherent to optical flow accumulate over time, leading to trajectory drift. Ultimately, this work underscores that while EKF-based SLAM is computationally efficient, its real-world robustness relies heavily on rigorous outlier rejection and high-quality, drift-free feature extraction.

Acknowledgment

I express sincere gratitude to Prof. Atanasov for designing this project, which provided an invaluable learning experience

in Extended Kalman Filtering, Lie group kinematics, and visual-inertial sensor fusion. Special thanks to the Teaching Assistants for their guidance and support.

References and Attribution

The mathematical formulation for the SLAM framework was derived primarily from the lecture notes "Sensing and Estimation in Robotics" by Prof. Nikolay Atanasov (UCSD, 2026). Standard scientific computing libraries, including NumPy and SciPy, were utilized for core matrix operations and Lie algebra matrix exponentials.