Lab 7 Report

Robotics Integration Group Project I

Yuwei ZHAO (23020036096)
Group #31 2025-12-27

Abstract

…

See Resources on github.com/RamessesN/Robotics_MIT.

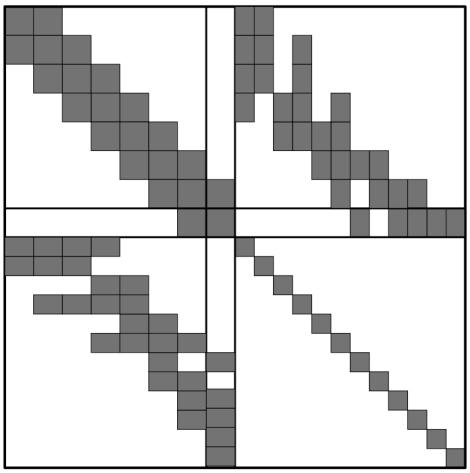
1 Introduction

2 Procedure

2.1 Individual Work

2.1.1 Spy Game

Consider the following spy-style plot of an information matrix (i.e., coefficient matrix in Gauss-Newton's normal equations) for a landmark-based SLAM problem where dark cells correspond to non-zero blocks:



Assuming robot poses are stored sequentially, answer the following questions:

1. **How many robot poses exist in this problem?**

   Based on the top-left block (Pose-Pose block) of the spy plot in `spy_game2.png`, there are 5 distinct diagonal blocks. Therefore, there are 5 robot poses.

2. **How many landmarks exist in the map?**

   Based on the bottom-right block (Landmark-Landmark block), there are 6 distinct diagonal blocks. Therefore, there are 6 landmarks.

3. **How many landmark have been observed by the current (last) pose?**

   The current pose corresponds to the last row (5th row) of the pose block. Looking at the off-diagonal block (top-right), there are 3 dark cells in this row. Thus, the current pose observed 3 landmarks (specifically landmarks 4, 5, and 6).

4. **Which pose has observed the most number of landmark?**

   By counting the number of dark blocks in each row of the top-right quadrant:

   - Pose 1: 1 landmark
   - Pose 2: 2 landmarks
   - Pose 3, Pose 4, and Pose 5: Each observed 3 landmarks.

   Therefore, Poses 3, 4, and 5 tie for the most observations.

5. **What poses have observed the 2nd landmark?**

   Locating the 2nd column in the landmark section (top-right quadrant), there are dark blocks corresponding to Pose 2 and Pose 3. Thus, these two poses observed the 2nd landmark.

6. **Predict the sparsity pattern of the information matrix after marginalizing out the 2nd feature.**

   Marginalizing out the 2nd landmark will create a fill-in (dependency) between all poses that observed it. Since Pose 2 and Pose 3 observed Landmark 2, marginalizing it will strengthen the connection between Pose 2 and Pose 3. Since these poses are already sequentially connected ($P_{\{k-1\}}$ to P_k), the block structure will remain largely visually similar, but the density within the Pose 2-3 block will increase.

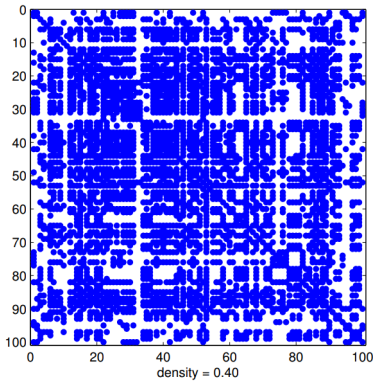
7. **Predict the sparsity pattern of the information matrix after marginalizing out past poses (i.e., only retaining the last pose).**

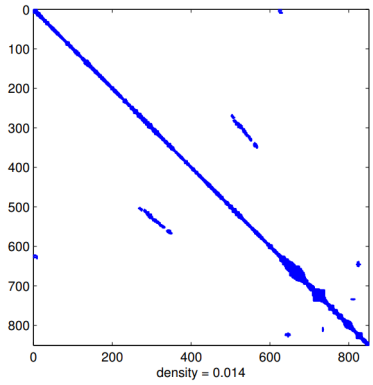
   Marginalizing out past poses (filtering approach) causes the information matrix to become dense (fully connected). Eliminating a pose induces correlations between all landmarks observed by that pose and the subsequent pose. Repeating this process entangles all historic landmarks, destroying the sparse structure.

8. **Marginalizing out which variable (chosen among both poses or landmarks) would preserve the sparsity pattern of the information matrix?**

   Marginalizing out the landmarks (Schur complement on structure) preserves the sparsity pattern. This results in the "Reduced Camera System" matrix, which typically retains a band-diagonal sparse structure, as poses are only connected to other poses with which they share common landmark observations (usually temporal neighbors).

9. The following figures illustrate the robot (poses-poses) block of the information matrix obtained after marginalizing out (eliminating) all landmarks in bundle adjustment in two different datasets. What can you say about these datasets (e.g., was robot exploring a large building? Or perhaps it was surveying a small room? etc) given the spy images below?





- Left Figure (Band-Diagonal): The non-zero elements are concentrated strictly around the main diagonal. This indicates that the robot was likely exploring a large environment (e.g., a long corridor) without revisiting previous locations. Current poses only share information with spatially/temporally adjacent poses.
- Right Figure (Off-Diagonal / Loop Closures): There are significant non-zero blocks far from the main diagonal (the "wings" in the corners). This indicates Loop Closures, where the current pose observes the same scene as a much earlier pose. This suggests the robot was likely surveying a small room or circling back to a previously visited area.

2.1.2 Well-begun is Half Done

Pose graph optimization is a non-convex problem. Therefore, iterative solvers require a (good) initial guess to converge to the right solution. Typically, one initializes nonlinear solvers (e.g., Gauss-Newton) from the odometric estimate obtained by setting the first pose to the identity and chaining the odometric measurements in the pose graph.

Considering that chaining more relative pose measurements (either odometry or loop closures) accumulates more noise (and provides worse initialization), propose a more accurate initialization method that also sets the first pose to the identity but chains measurements in the pose graph in a more effective way. A 1-sentence description and rationale for the proposed approach suffices.

…

2.1.3 Feature-based methods for SLAM

Read the ORB-SLAM paper (available here) and answer the following questions:

1. **Provide a 1 sentence description of each module used by ORB-SLAM (Fig. 1 in the paper can be a good starting point).**

   …

2. **Consider the case in which the place recognition module provides an incorrect loop closure. How does ORB-SLAM check that each loop closure is correct? What happens if an incorrect loop closure is included in the pose-graph optimization module?**

   …

2.1.4 Direct methods for SLAM

Read the LSD-SLAM paper (available here, see also the introduction below before reading the paper) and answer the following questions:

1. **Provide a 1 sentence description of each module used by LSD-SLAM and outline similarities and differences with respect to ORB-SLAM.**

   …

2. **Which approach (between feature-based or direct) is expected to be more robust to changes in illumination or occlusions? Motivate your answer.**

   …

2.1.5 From landmark-based SLAM to rotation estimation

Consider the following landmark-based SLAM problem:

$$\min_{t_i\in\mathbb{R}^3,\, R_i\in\mathrm{SO}(3),\, p_i\in\mathbb{R}^3} \sum_{(i,k)\in\mathcal{E}_l} \|R_i^T(p_k - t_i) - \bar{p}_{ik}\|_2^2 + \sum_{(i,j)\in\mathcal{E}_o} \|R_i^T(t_j - t_i) - \bar{t}_{ij}\|_2^2 + \|R_j - R_i\bar{R}_{ij}\|_F^2$$

Where the goal is to compute the poses of the robot $(t_i, R_i), i = 1, ..., N$ and the positions of point-landmarks $p_k, k = 1, ..., M$ given odometric measurements $\left(\bar{t}_{\mathrm{ij}}, \overline{R}_{\mathrm{ij}}\right)$ for each odometric edge $(i, j) \in \varepsilon_0$ (here ε_0 denotes the set of odometric edges), and landmark observations $\overline{p}_{\mathrm{ik}}$ of landmark k from pose i for each observation edge $(i, k) \in \varepsilon_l$ (here ε_l denotes the set of pose-landmark edges).

1. **Prove the following claim: "The optimization problem (1) can be rewritten as a nonlinear optimization over the rotations $R_i, i = 1, ..., N$ only." Provide an expression of the resulting rotation-only problem to support the proof.**

   …

2. **The elimination of variables discussed at the previous point largely reduces the size of the optimization problem (from 6N+3L variables to 3N variables).However, the rotation problem is not necessarily faster to solve. Discuss what can make the rotation-only problem more computationally-demanding to solve.**

   …

2.2 Team Work

2.2.1 Prepare the Dataset

Download the datasets, then run them via the commands below:

```
./run_docker.sh orbslam:latest
./run_docker.sh kimera:latest
```

Then, we can see the running results as follows:

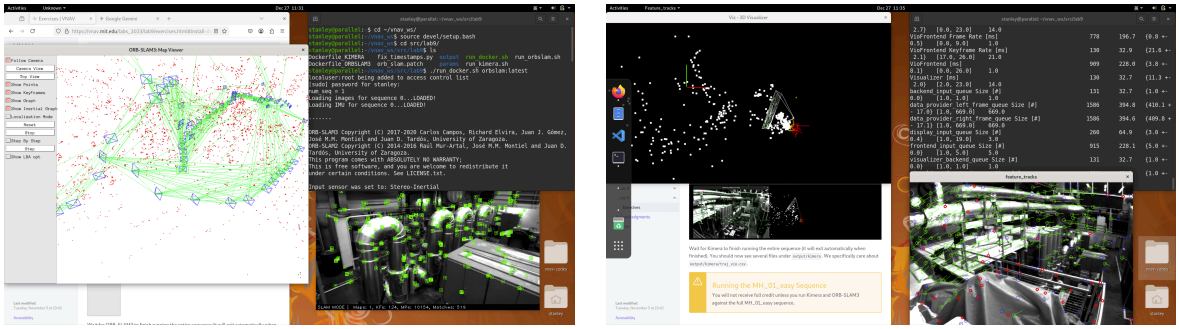


Figure 4: **orbslam** & **kimera** Running Snapshot

2.2.2 Performance Comparison

After running the `fix_timestamps.py` to fix the timestamps of the trajectory files, then use `evo_traj` to compare **OrbSlam** and **Kimera**:

```
evo_traj tum output/kimera/kimera.txt output/orbslam/orb_slam3.txt --plot
```

And we have

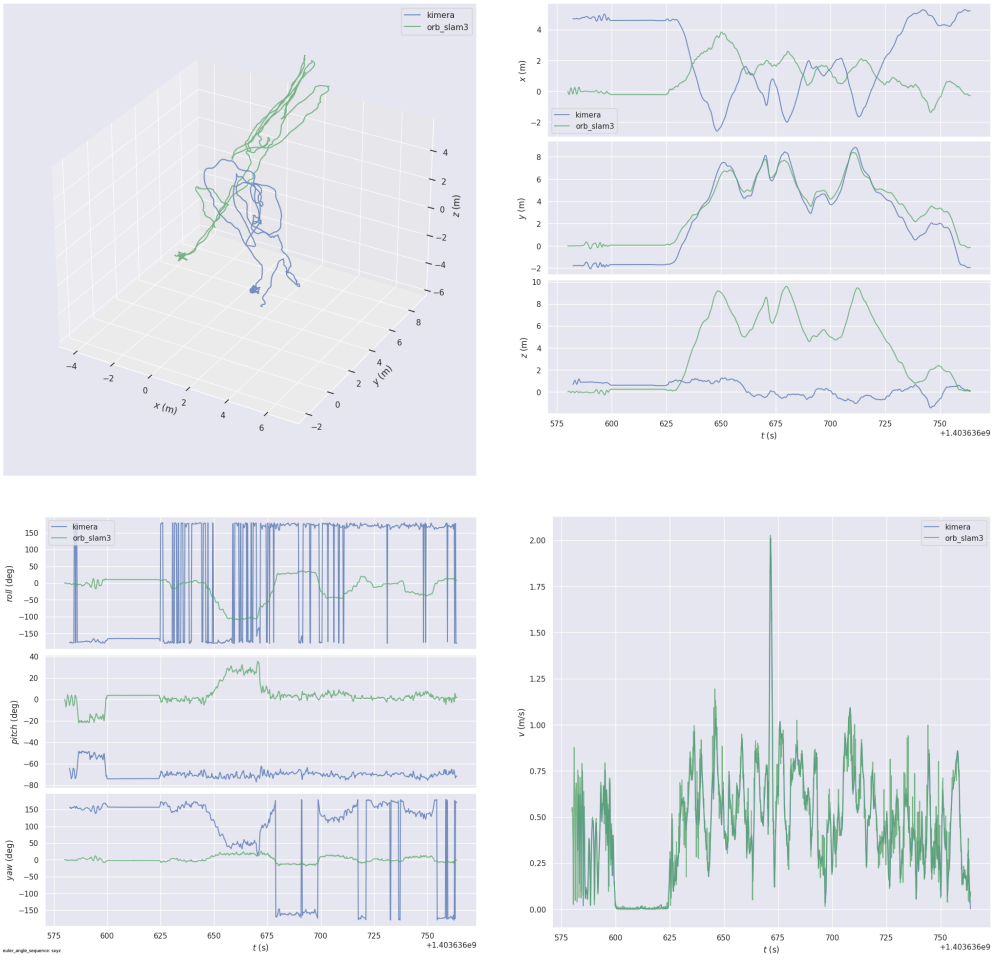


Figure 5: **OrbSlam / Kimera** (without aligning) Performance Comparison

From the figures above, it can be seen that without alignment, the trajectories of **Kimera** (blue) and **OrbSlam3** (green) are spatially distinct, which is expected. The **3D Trajectory** and **XYZ** plots show different starting origins and orientations, indicating that each algorithm initialized its own local world coordinate frame at startup. The **RPY** plot confirms a significant constant offset in Yaw (approx. 150^{o} difference), while Roll and Pitch are more consistent due to gravity alignment. However, the **Speed** plot demonstrates high consistency in velocity estimation, proving that both algorithms are capturing the drone's dynamics correctly despite the coordinate frame mismatch.

After aligning the trajectories through `evo_traj euroc ~/datasets/vnav/MH_01_easy/mav0/state_groundtruth_estimate0/data.csv --save_as_tum` and then we run the comparison `evo_traj tum output/kimera/kimera.txt output/orbslam/orb_slam3.txt --ref data.tum --plot --align` to compare them:

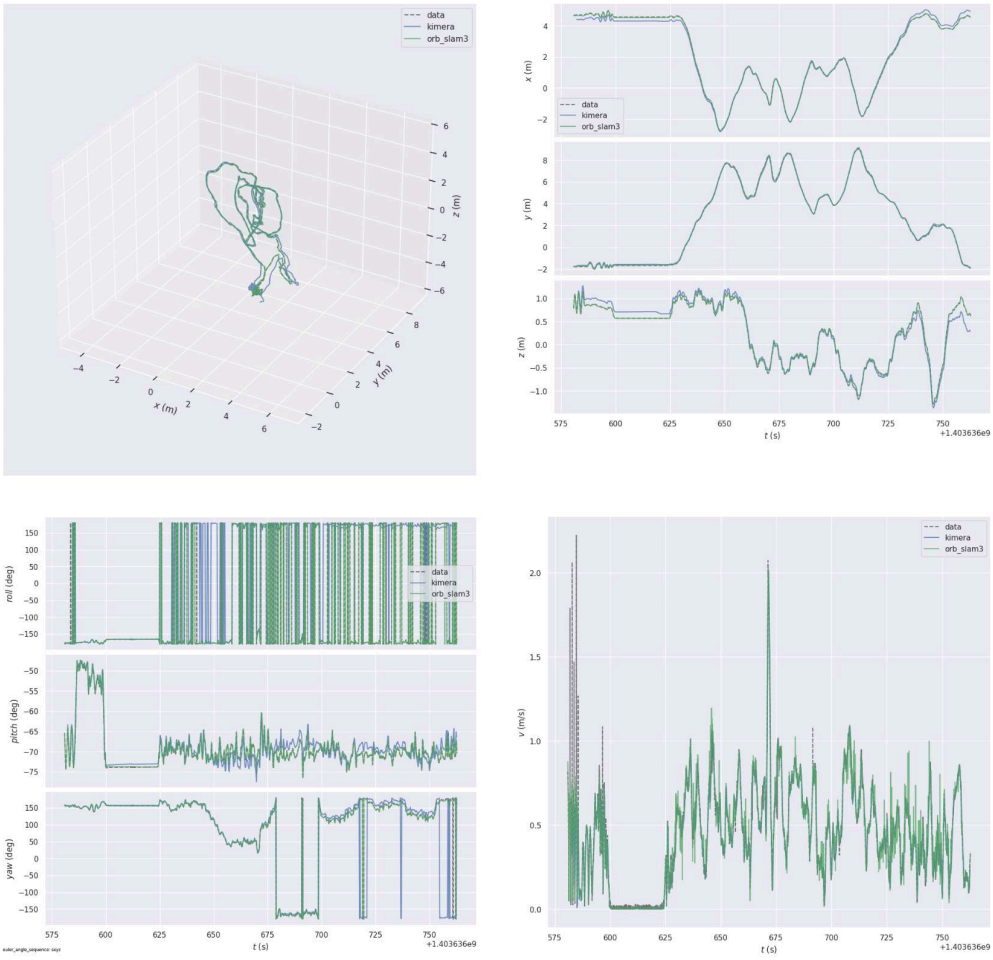


Figure 6: **OrbSlam / Kimera** Performance Comparison

From the aligned results above, it can be seen that after performing alignment against the Ground Truth, the estimated trajectories from both **Kimera** and **OrbSlam3** closely match the reference path. The **3D Trajectory** and **XYZ** plots show minimal drift, with both algorithms successfully tracking the complex motion of the MAV. The **RPY** plots indicate precise attitude estimation, accurately capturing rapid orientation changes. In the **Speed** plot, although both estimates contain typical high-frequency noise inherent to IMU-based prediction, they accurately follow the velocity profile of the ground truth. Overall, both systems demonstrate reliable state estimation performance on the EuRoC dataset.

2.2.3 LDSO

We successfully ran the LDSO (LiDAR-Direct-Sparse-Odometry) pipeline on the EuRoC dataset. As shown in the snapshot below, the viewer visualizes the sparse 3D point cloud reconstructed from the environment, along with the active keyframes and the current camera pose. The semi-dense reconstruction clearly outlines the structural features of the Machine Hall.

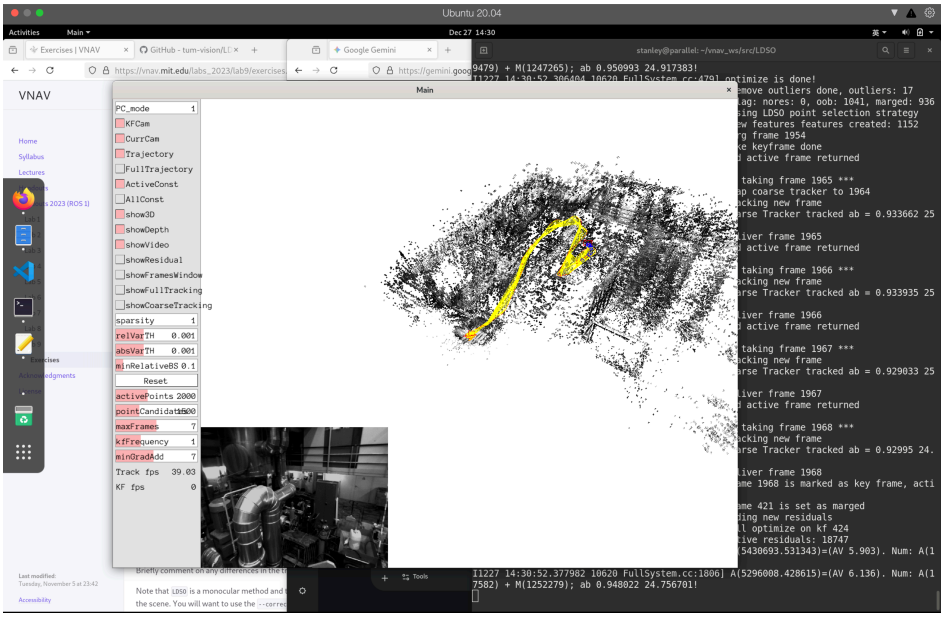


Figure 7: **LDSO** Running Snapshot

Since LDSO is a **monocular** direct visual odometry method, it inherently suffers from **scale ambiguity** (i.e., it cannot observe the absolute metric scale of the world). Therefore, when comparing its trajectory against the Ground Truth and stereo/VIO pipelines (Kimera and OrbSlam3), it is mandatory to use Sim3 alignment (alignment with rotation, translation, and **scale correction**). We enabled this using the `--correct_scale` flag in `evo`.

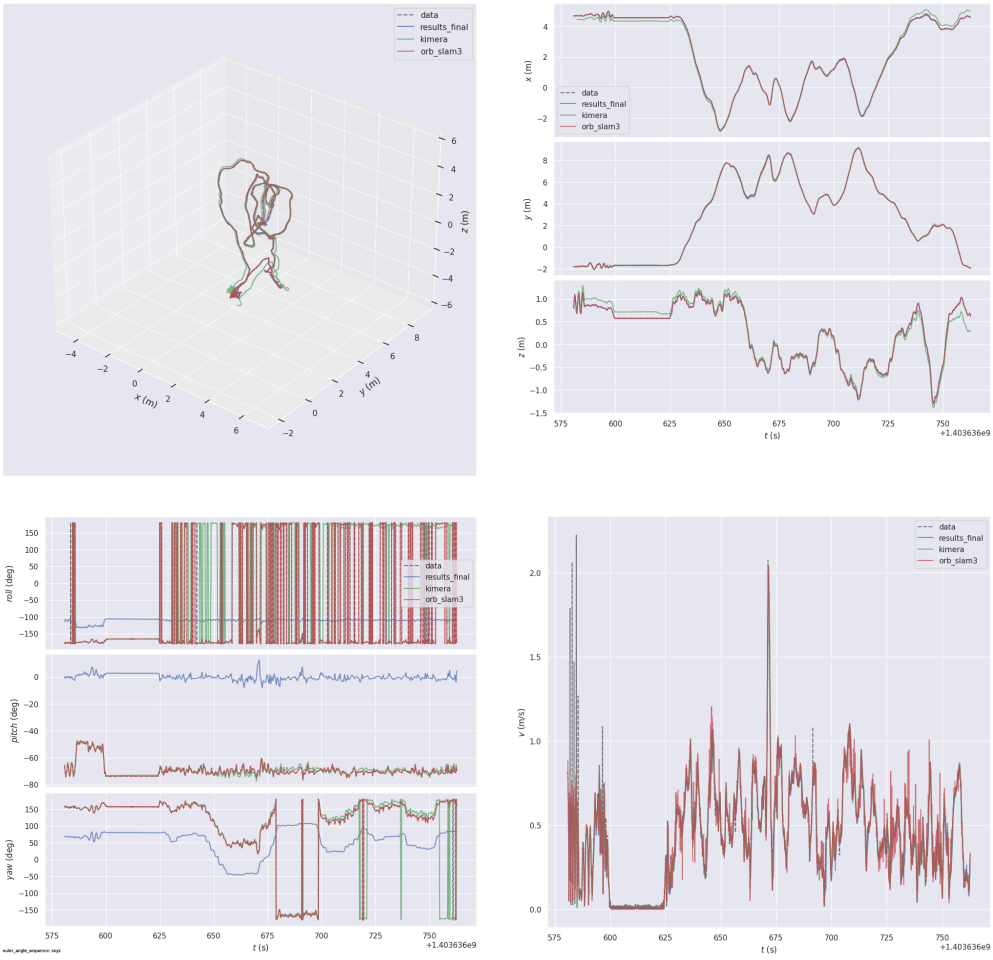


Figure 8: **OrbSlam / Kimera / LDSO** Performance Comparison

From the comparison plots above, several observations can be made:

1. **Trajectories:** After applying scale correction, the LDSO trajectory (blue line, labeled `results_final`) aligns remarkably well with the Ground Truth (`data`), Kimera, and OrbSlam3. This demonstrates that despite lacking IMU data and stereo baselines, the direct photometric optimization of LDSO is capable of recovering the geometric structure of the camera path with high accuracy in feature-rich environments like EuRoC.

2. **XYZ:** The LDSO trajectory appears quite smooth, particularly in the XYZ position plots. Direct methods often benefit from using information from all pixels with sufficient gradient, which can result in robust tracking even when specific corner features might be sparse, although it can be sensitive to photometric calibration and lighting changes.

3. **RPY:** The Roll, Pitch, and Yaw estimates of LDSO track the ground truth variations accurately. However, unlike VIO methods (Kimera/OrbSlam3) which have an observable gravity vector from the accelerometer to constrain Roll and Pitch, monocular VO can

sometimes exhibit slow drift in these axes over long durations. In this sequence, however, LDSO maintains its orientation stability effectively.

4. **Speeds:** The velocity profile derived from the aligned LDSO trajectory matches the VIO pipelines. This confirms that the temporal consistency of the estimated pose is preserved, meaning the "virtual speed" in the scaled monocular frame corresponds linearly to the real-world speed after Sim3 alignment.

3 Reflection and Analysis

4 Conclusion

5 Source Code

•
•