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.

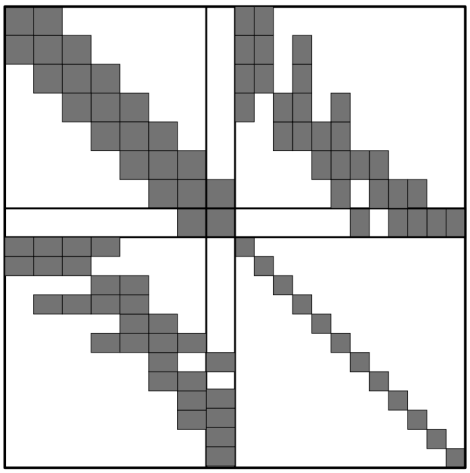
1 Introduction

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?**

   From the matrix structure, the top-left block corresponds to the Pose-Pose constraints (H_{pp}), as it exhibits the sequential band structure typical of odometry.

∵ There are 8 blocks along the diagonal of the H_{pp} matrix.

∴ There are 8 Robot Poses.

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

   From the bottom-right block, which corresponds to H_{ll}, the relationship of Landmark-Landmark.

   ∵ There are 12 non-zero blocks along the diagonal of this section

   ∴ There are 12 Landmarks.

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

   Focus on the last row of the top-right block (corresponding to Pose 8).

   This block represents the Pose-Landmark constraints (H_{pl}).

   ∵ There are 5 non-zero blocks in this row within the landmark section.

   ∴ The current pose has observed 5 landmarks.

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

   Check the rows in the top-right block (H_{pl}) to see how many landmarks each pose observes.

   ∵ The last row (corresponding to Pose 8) has the maximum number of non-zero blocks (5 blocks).

   ∴ Pose 8 has observed the most number of landmarks.

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

   Locate the column corresponding to the 2nd landmark in the top-right block (H_{pl}).

   This column is part of the first vertical group of non-zero blocks.

   Checking the rows corresponding to this group:

   - Row 1, 2, and 3 contain non-zero blocks.
   - Row 4 onwards is empty for this column.

   ∴ The 2nd landmark has been observed by Pose 1, Pose 2, and Pose 3.

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

   Conclusion — The mapped matrix should be like this:

$$
\left[\begin{array}{cccccccc|cccccccccc}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
\hline
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right]
$$

---

§ Proof:

After marginalizing out the 2nd feature, the row and column corresponding to the 2nd feature will be removed, and a fill-in will occur in the top-left block (H_{pp}). Since the 2nd feature connects Pose 1, Pose 2, and Pose 3, these three poses will become fully connected (forming a dense 3×3 block). Besides, in the top-right (H_{pl}) and bottom-left (H_{lp}) blocks, the column/row corresponding to the 2nd feature is deleted, and all subsequent columns/rows (Landmark 3 to 12) shift to the left/up to fill the gap.

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

   Conclusion — The sparsity pattern should be a full matrix of ones:

$$
\left[\begin{array}{c|cccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}\right]
$$

---

§ Proof:

Predict the sparsity pattern after marginalizing out past poses (Pose 1 to Pose 7), keeping only Pose 8 and all landmarks: **1)** The matrix reduces to size 13×13 (1 Pose + 12 Landmarks). **2)** Marginalizing the trajectory (the "backbone" connecting landmarks) creates direct correlations between all remaining variables. **3)** The information matrix becomes fully dense:

- The remaining pose (P_8) becomes correlated with all landmarks.
- All landmarks become correlated with each other (the $H_{\{ll\}}$ block becomes dense).

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

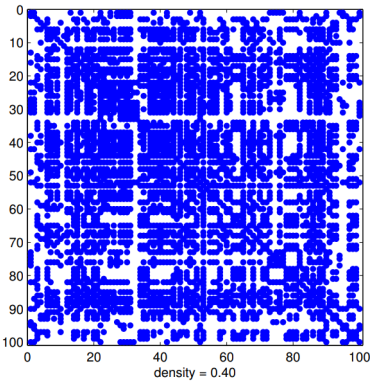
Conclusion — To preserve the sparsity pattern (avoid creating new non-zero blocks where there were zeros), we should chose a variable whose marginalization does not introduce new edges between previously unconnected nodes, and that is the **12th Landmark**.

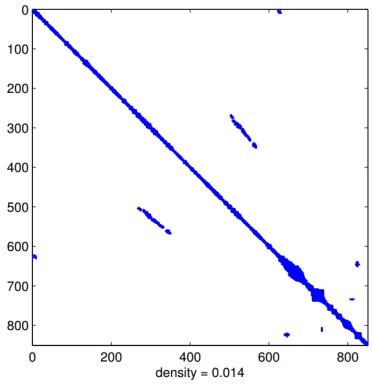
---

§ Proof:

1) Looking at the last column of the matrix, the 12th landmark is observed by only one pose (Pose 8). **2)** Marginalizing a variable connects all its neighbors to each other. Since the 12th landmark has only one neighbor (Pose 8), there are no distinct pairs of nodes to connect. **3)** The information from the landmark is simply folded into the diagonal block of Pose 8. No new fill-in entries are created in the matrix.

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?





Conclusion — The left figure corresponds to surveying a small room and the right figure corresponds to exploring a large building.

---

§ Proof:

The left matrix is very dense (density 0.40). Marginalizing landmarks creates connections between poses that observe the same features. A dense matrix indicates that nearly all poses share a high number of common observations (co-visibility). This happens when the robot is confined in a small space, looking at the same scene repeatedly from slightly different angles.

The right matrix is sparse (density 0.014) with a dominant diagonal band and specific off-diagonal blocks. The **diagonal band** represents the sequential motion (odometry), where the robot mostly sees new features as it moves forward. While the **off-diagonal blocks** represent loop closures. These occur when the robot revisits a previously mapped area, creating constraints between the current pose and distant past poses. This structure is typical of large-scale trajectories where the robot explores new areas and occasionally returns to old ones.

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.

Approach — Construct a **Shortest Path Spanning Tree** (e.g., using Breadth-First Search) rooted at the first pose, and initialize all other poses by propagating measurements along the edges of this tree.

Rationale — This minimizes the number of relative transformations (edges) composed to reach any given node (utilizing loop closures as "shortcuts"), thereby significantly reducing the accumulated drift compared to the long sequential odometry chain.

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).**
   - **Map Initialization:** This module computes the initial camera pose and map structure from two frames by automatically selecting between a homography model for planar scenes and a fundamental matrix for general scenes.
   - **Tracking:** This thread localizes the camera in every frame by matching features to the local map and decides when to insert a new keyframe into the system.
   - **Local Mapping:** This thread processes new keyframes to optimize the local reconstruction via bundle adjustment and performs culling of redundant map points and keyframes.
   - **Loop Closing:** This thread searches for loops with every new keyframe to detect drift and corrects the map by performing a pose graph optimization over the Essential Graph.
   - **Place Recognition:** This embedded module uses a bag-of-words visual vocabulary to efficiently perform loop detection and global relocalization when tracking is lost.
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?**

   ORB-SLAM employs a two-step mechanism to ensure the validity of the loop closure:
   - **Temporal/Geometric Consistency Check**: The system must **continuously detect three consistent closed-loop candidates** (that is, these three candidate keyframes are interconnected in the common view) before accepting the position of this closed-loop candidate, thereby eliminating accidental incorrect matches.
   - **Geometric Verification (`Sim(3)` Transformation)**: For the candidates that pass the initial screening, the system will use the RANSAC algorithm to calculate the **`Sim(3)` transformation** between the current keyframe and the closed-loop keyframe. Only when this transformation is supported by **sufficiently many inliers**, will this closed-loop be finally accepted.

The role of pose graph optimization is to **distribute** the accumulated errors from the closed loop throughout the entire graph. If an incorrect closed loop (i.e., an incorrect geometric constraint) is included:

- The optimizer will attempt to minimize the cost function that includes this erroneous constraint, forcing the actually irrelevant keyframes to be brought closer or aligned.
- This will result in **severe map corruption/distortion**, as the originally correct trajectory will be distorted to fit this erroneous constraint, thereby destroying the global consistency.

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.**
   - **Tracking:** This module estimates the rigid body pose `se(3)` of the new image relative to the current key frame by using the pose of the previous frame as the initial value and employing the Direct Image Alignment algorithm.
   - **Depth Map Estimation:** This module utilizes the tracked frames and conducts numerous pixel-level small-baseline stereo comparisons to refine the depth map of the current key frame, or create a new key frame when the camera has moved too far.
   - **Map Optimization:** This module continuously optimizes the pose graph composed of keyframes in the background. It detects loops and scale drift through direct `Sim(3)` image alignment and performs global consistency optimization.

   Similarities
   - **Real-time monocular system:** Both are SLAM systems based on monocular cameras and capable of running in real-time on the CPU.
   - **Keyframe architecture:** Both of them adopt a keyframe-based approach instead of filtering each frame individually.
   - **Pose graph optimization:** Both use pose graph optimization (such as `g2o`) to maintain the global consistency of the map.
   - **Handling scale drift:** For the scale drift problem in monocular SLAM, both methods utilize the `Sim(3)` transformation (including the scale factor) for loop closure correction and optimization.

   Differences

- **Methodology**: **LSD-SLAM** is a direct method that directly estimates the pose and geometry by minimizing the photometric error on the pixel intensities of the images. While **ORB-SLAM** is a feature-based method. It calculates by extracting and matching ORB feature points and minimizing the re-projection error.
- **Map Density: LSD-SLAM** generates semi-dense (densely populated in gradient areas) depth maps. While **ORB-SLAM** generates a sparse point cloud map.
- **Initialization: LSD-SLAM** can be initialized from a random depth map and relies on subsequent motion convergence. While **ORB-SLAM** has a dedicated automatic initialization step, which constructs the initial map by performing parallel computations of the homography matrix and the fundamental matrix.
- **Loop Detection**: **LSD-SLAM** relies on appearance-based algorithms to propose candidates and verifies the closed loop through "reciprocal tracking check". While **ORB-SLAM** employs the position recognition module based on the bag-of-words model (`DBoW2`) for loop closure detection and repositioning.

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

Feature-based approaches such as ORB-SLAM are generally considered to be more robust to lighting variations and occlusions.

Robustness to Illumination

- **Feature-based appr:** The feature-based approach uses feature descriptors (such as ORB), which are designed to have good invariance to illumination and viewpoint changes. This means that even if the overall brightness of the image changes, the feature points can still be correctly matched.
- **Direct appr:** The direct method (such as LSD-SLAM) is based on the "photometric consistency assumption" (i.e., the assumption that the pixel intensity of the same point in the scene remains unchanged between different frames). This makes them susceptible to automatic gain, automatic exposure adjustment, and roll shutter ghosting. Although it can be alleviated by an affine illumination model, it is usually less stable than the feature-based method under drastic illumination changes.

Robustness to Occlusions

- **Feature-based appr:** The feature-based method supports wide baseline matching. When occlusion occurs, even if some feature points are lost or wrongly matched, algorithms such as RANSAC can effectively eliminate outliers and use the remaining correct matching points to restore the pose.

- **Direct appr:** The direct method is usually limited by a narrower baseline. It has higher requirements for the continuity between images. Although robust kernel functions (such as Huber norm) are used to handle outliers , large-scale occlusion can significantly disrupt the optimization terrain of photometric errors and easily lead to tracking failure.

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 $\left(\overline{t}_{\mathrm{ij}}, \overline{R}_{\mathrm{ij}}\right)$ 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:

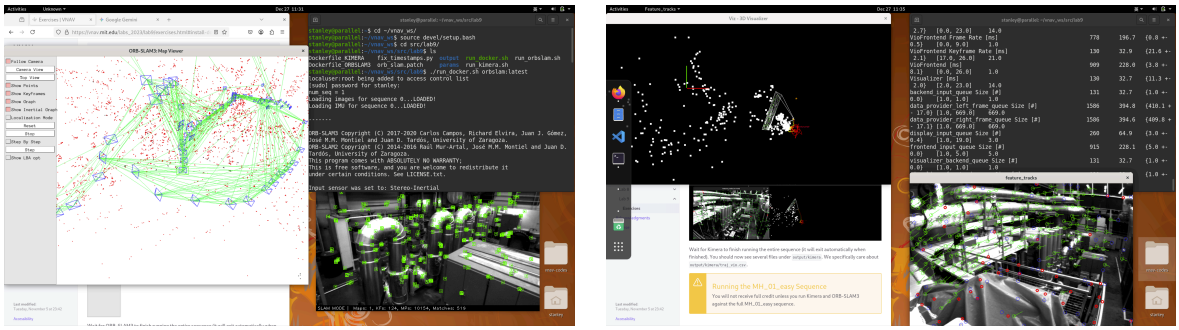


Figure 4: **orbslam** & **kimera** Running Snapshot

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

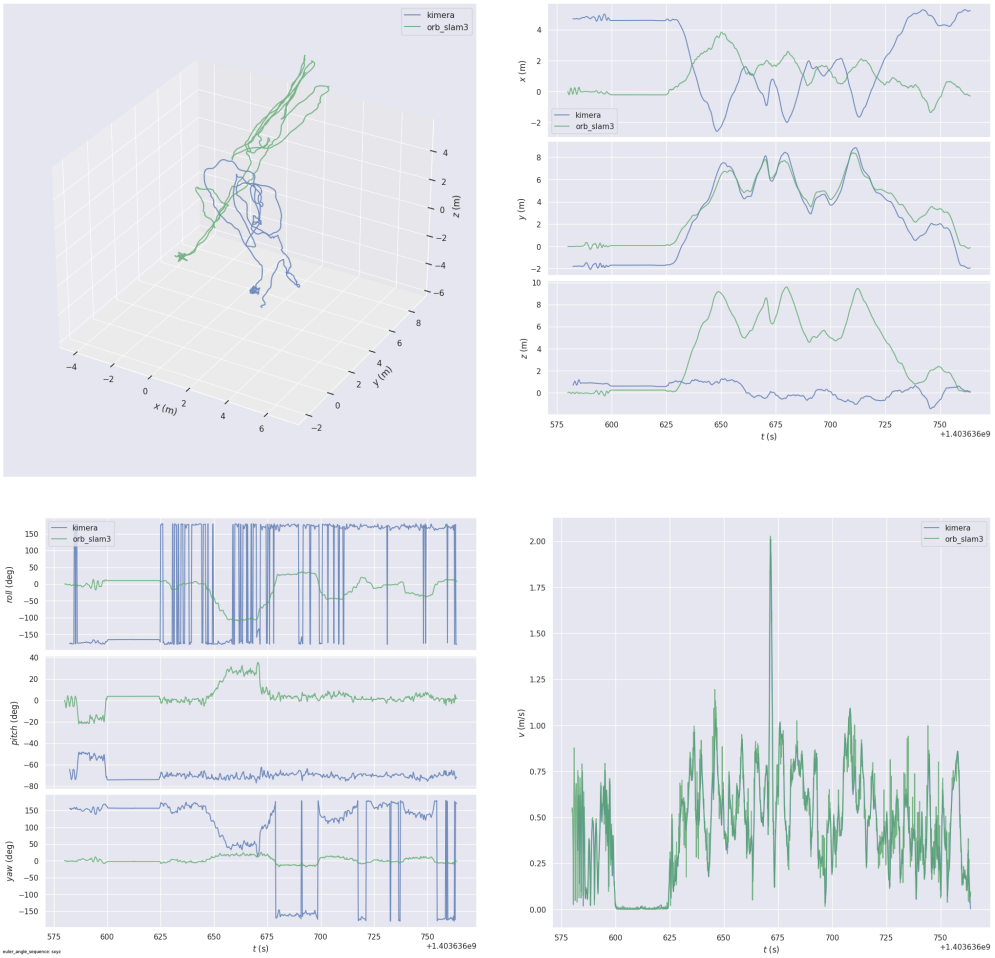


Figure 5: **OrbSlam** / **Kimera** (without aligning) Performance Comparison

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:

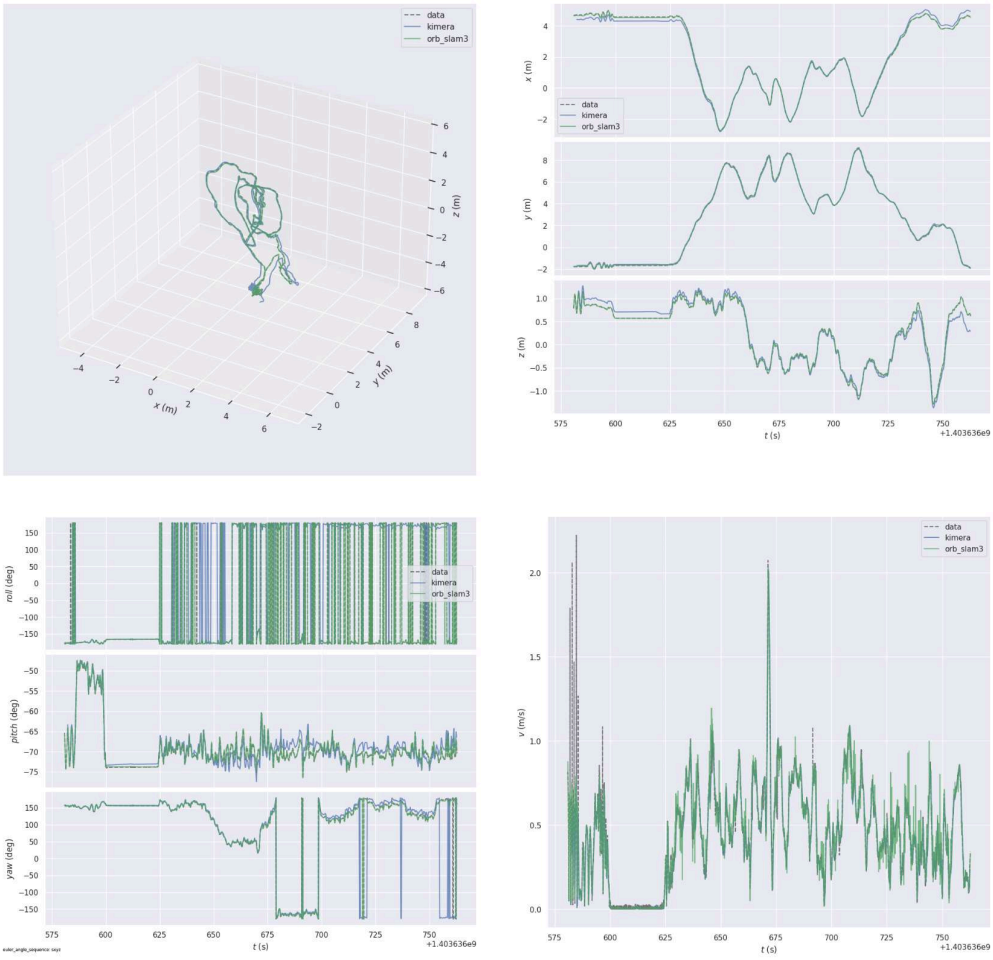


Figure 6: **OrbSlam / Kimera** Performance Comparison

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 atti-

tude 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.

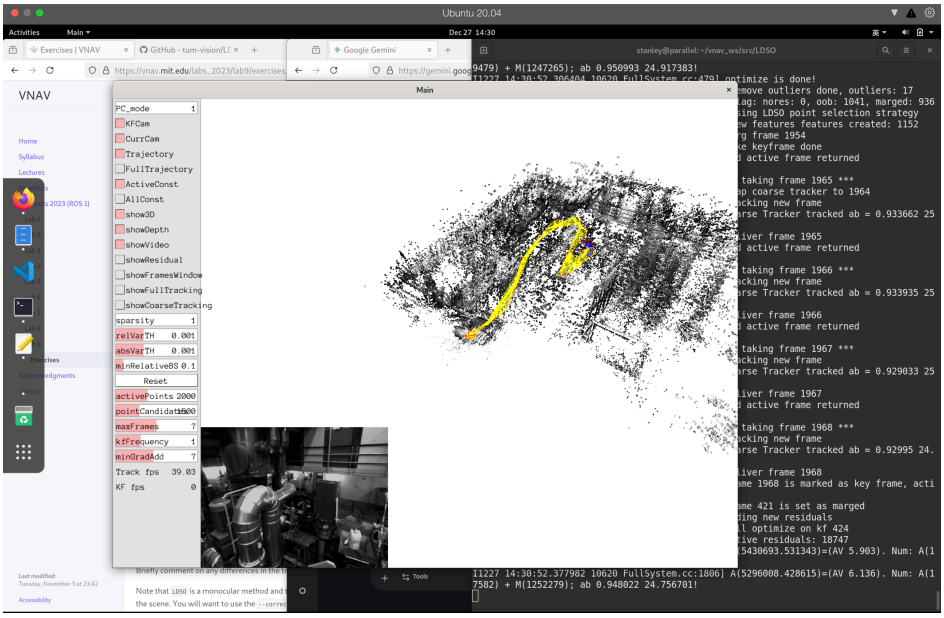


Figure 7: **LDSO** Running Snapshot

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`.

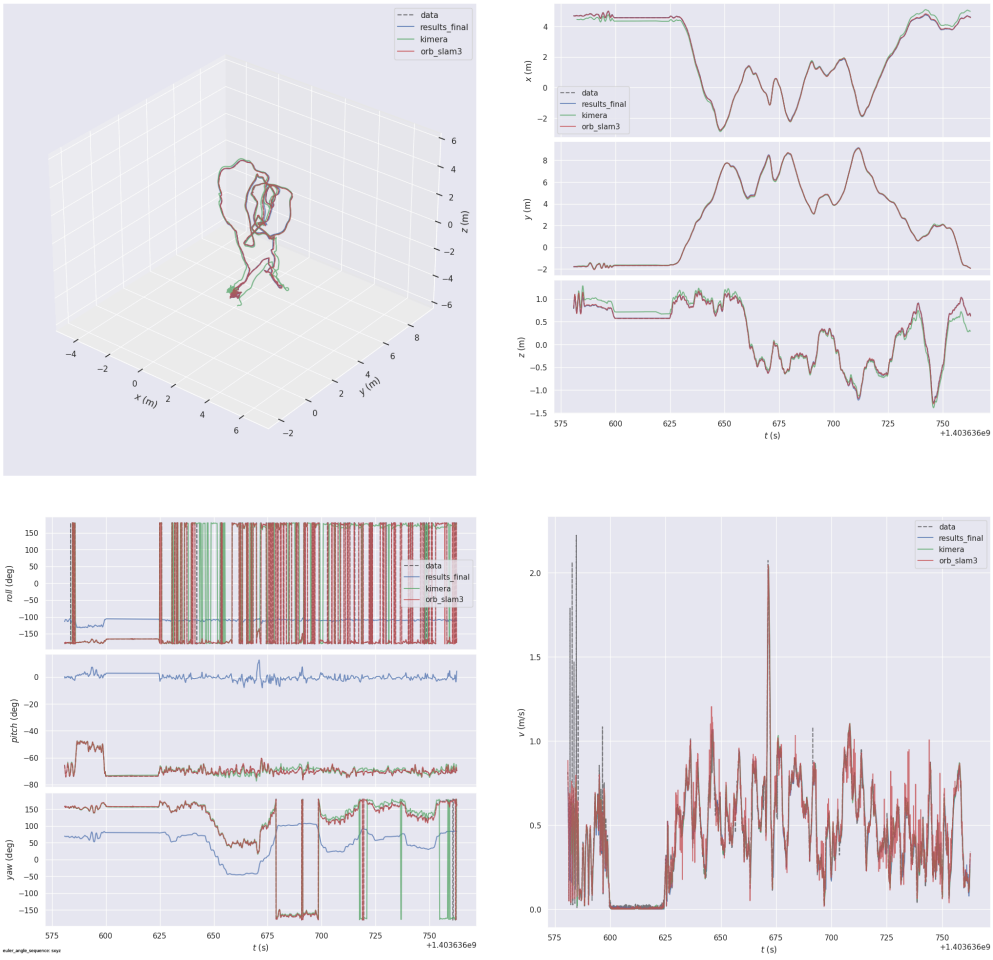


Figure 8: **OrbSlam / Kimera / LDSO** Performance Comparison

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

•
•