Lab 9 Report: EXERCISES

Yilin Zhang [1]. Lab Group: 31.

I. Individual

Deliverable 1 - Spy Game

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

a) How many robot poses exist in this problem?: Looking at the top-left quadrant, the Pose-Pose block, there are 4 distinct blocks along the main diagonal. This corresponds to the robot's trajectory states P_1, P_2, P_3, P_4.

b) How many landmarks exist in the map?: Looking at the bottom-right quadrant, the Landmark-Landmark block, there are 12 distinct square blocks along the diagonal. This corresponds to landmarks L_1 through L_12.

c) How many landmarks have been observed by the current (last) pose?: The last pose is *Pose 4*, corresponding to the 4^{th} row in the block structure. Then, look at the *Top-Right quadrant*, the Pose-Landmark block, specifically *Row 4*. The non-zero blocks in this row align with the last 4 columns of the matrix. These correspond to landmarks L_9, L_10, L_11, L_12. Therefore, the last pose observes 4 landmarks.

d) Which pose has observed the most number of landmark?: By examining the width of the observation blocks in the Top-Right quadrant for each pose row:

- Pose 1 observes L_1, L_2.
- Pose 2 observes L_2, L_3, L_4.
- Pose 3 observes $L_4, L_5, L_6, L_7, L_8, L_9$.
- Pose 4 observes L_9, L_10, L_11, L_12.
- Pose 3 has the widest block, meaning it observed the most features.

e) What poses have observed the 2nd landmark?: Locate the column corresponding to the 2nd landmark in the Top-Right quadrant, which is the 2nd column of that block. There are non-zero entries in the rows corresponding to *Pose 1* and *Pose 2*. This creates the overlap that links and through a shared feature observation.

f) Predict the sparsity pattern of the information matrix after marginalizing out the 2nd feature.: Marginalizing out a variable creates a "fill-in" between all variables that were connected to the marginalized variable.

The 2nd feature is observed by *Pose 1* and *Pose 2*. Eliminating L_2 would create an edge between P_1 and P_2. However, P_1 and P_2 are *already connected* via odometry, evident from the off-diagonal entries in the top-left Pose-Pose block. The result is the sparsity pattern of the remaining blocks generally stays the same, no new non-zero blocks are created, but the values within the existing block will become denser. The row and column for will be removed.

g) Predict the sparsity pattern of the information matrix after marginalizing out past poses (i.e., only retaining the last pose).: Marginalizing out the trajectory P_1, P_2, P_3 essentially bakes all past information into the remaining variables.

Poses are connected to the landmarks they observed. Marginalizing a pose creates a clique among all landmarks seen by that pose and its neighbors. Because the poses form a chain and observe overlapping sets of landmarks, this process will propagate correlations through the entire map. Therefore, the resulting matrix (containing P_4 and all Landmarks) will have a **fully dense block** for all landmarks observed by the marginalized poses (L_1 to L_9). These landmarks will essentially become fully correlated with each other and with P_4. The sparsity is destroyed for the landmark section.

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

As seen in Q7, Mmarginalizing *Poses* creates dense fill-in among landmarks, destroying sparsity. Marginalizing *Landmarks* only creates connections between the poses that observed that specific landmark. Since landmarks in this example are observed by consecutive poses, and those poses are already connected by odometry, marginalizing landmarks adds information to existing blocks without creating new blocks far from the diagonal. This operation results in the "Reduced Camera Matrix" or Pose Graph, which retains the sparse, block-tridiagonal structure of the original pose block.

i) The figures in appendix A 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 1?: These images show the Pose-Pose information matrix after marginalizing out all landmarks. The density of off-diagonal elements indicates how many loop closures or shared observations exist between poses at different times.

The left image depicts a "small room" and a "high overlap". The matrix is very dense with many off-diagonal entries far from the main diagonal. The robot frequently observes landmarks it has seen before, even from much earlier in the trajectory. This implies the robot is moving in a confined space (like a small room) or constantly looping back, creating strong correlations between current and past poses.

While the right image illustrates a "large building" and "exploration". The matrix is sparse and band-diagonal (tridiagonal). The robot mostly only shares information with its immediate neighbors P_{t-1}, P_{t+1}. It rarely re-observes old features. This implies an exploration trajectory in a large environment, like a long corridor or a city street, where the robot moves forward and does not return to previous locations.

[1]Yilin zhang, OUC id: 23020036094, is with Faculty of Robotics and Computer Science, Ocean University of China and Heriot-Watt University, China Mainland zyl8820@stu.ouc.edu.cn

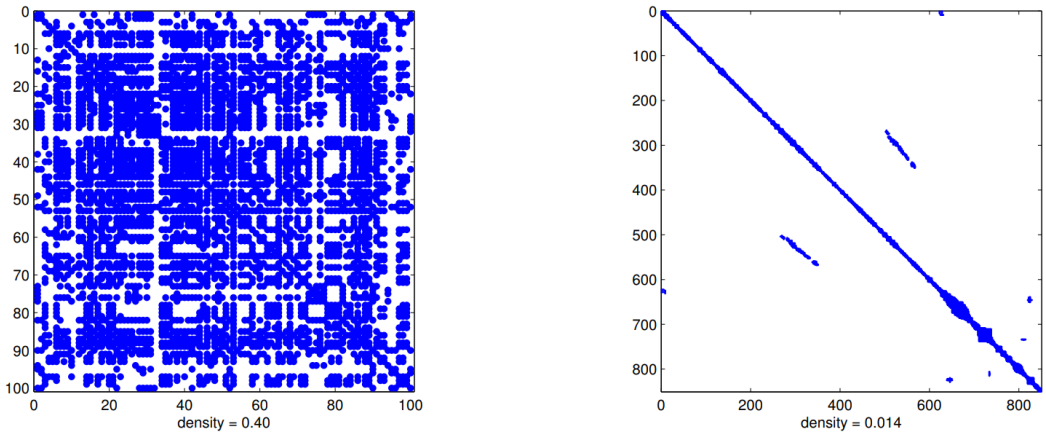


Fig. 1: Spy game images of Deliverable 1.9.

Deliverable 2 - Well-begun is Half Done

We propose to initialize each node by chaining the relative pose measurements along the shortest path from the root to that node in the pose graph, where the length of an edge is defined as the trace of its inverse covariance matrix, because this minimizes the accumulated uncertainty along the path and hence provides a more accurate initial guess than simply using the odometric chain.

Deliverable 3 - Feature-based methods for SLAM

Read the ORB-SLAM paper (available here) and answer the following questions: [1]

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

- *Tracking*: Estimates the camera pose for each frame, decides when to insert a new keyframe, and performs global relocalization if tracking is lost.
- *Local Mapping*: Processes new keyframes, triangulates new map points, performs local bundle adjustment, and culls redundant keyframes and map points.
- *Loop Closing*: Detects large loops using bag-of-words place recognition, computes a similarity transformation to correct drift, and optimizes a pose graph over the Essential Graph for global consistency.
- *Place Recognition*: Employs a bag-of-words approach (DBoW2) to recognize previously visited places for loop detection and camera relocalization.
- *Map*: Stores keyframes and map points, maintains a covisibility graph and an Essential Graph, and supports local and global optimization.

k) 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?: When the place recognition module proposes a loop candidate, ORB-SLAM performs *geometric verification* before accepting it. Using RANSAC and Horn's method, a 7-DoF similarity transformation $\mathbf{S}_{il}$ is estimated between the current keyframe and the loop candidate. The transformation is refined, and additional matches are searched. Only if enough inliers support the transformation is the loop accepted. The system requires *three consecutive consistent loop detections*, keyframes connected in the covisibility graph to confirm a loop.

If a false loop closure is mistakenly accepted and incorporated into the *Essential Graph* optimization. The pose-graph optimization would propagate the incorrect constraint throughout the graph, causing significant drift and misalignment in the reconstructed map and trajectory. Bundle adjustment failure**: Subsequent global bundle adjustment (BA) may fail to converge or could worsen the error, as noted in Section VIII-E. In practice, ORB-SLAM's stringent geometric and temporal consistency checks make such false positives rare. If one occurs, manual reset or mapping from scratch may be needed.

Deliverable 4 [Optional] - 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: [2]

l) Provide a 1 sentence description of each module used by LSD-SLAM and outline similarities and differences with respect to ORB-SLAM.: LSD-SLAM consists of three modules: *tracking*: direct SE(3) image alignment for camera pose estimation, *depth map estimation*: filter-based semi-dense depth map creation and refinement, and *map optimization*: pose-graph optimization with Sim(3) constraints for loop closure and scale drift correction. Compared to ORB-SLAM, both systems are keyframe-based and use pose graph optimization for loop closure; however, LSD-SLAM uses direct photometric alignment and semi-dense depth maps, while ORB-SLAM relies on feature-based matching and sparse bundle adjustment.

m) Which approach (between feature-based or direct) is expected to be more robust to changes in illumination or

occlusions? Motivate your answer.: Feature-based methods are generally more robust to changes in illumination or occlusions because they use descriptors designed for invariance to lighting and can handle occlusions through robust matching e.g., RANSAC, while direct methods rely on the brightness constancy assumption and are sensitive to illumination changes and occlusions. [3]

Deliverable 5 [Optional] - From landmark-based SLAM to rotation estimation

The landmark-based SLAM problem can be reduced to a rotation-only optimization by eliminating the translation and landmark position variables. Given the cost function 1, we first rewrite the first two terms using rotation invariance of the Euclidean norm:

$$\sum_{(i,k)\in\epsilon_l} \|p_k - t_i - R_i\overline{p}_{ik}\|_2^2 + \sum_{(i,j)\in\epsilon_o} \|t_j - t_i - R_i\bar{t}_{ij}\|_2^2.$$

Stacking all translation and landmark variables into a vector $x = [t_1, \ldots, t_N, p_1, \ldots, p_M]^\top$, these terms can be expressed as $\|Ax - d(R)\|^2$, where A is a constant sparse matrix determined by the graph structure, and $d(R)$ is a vector depending on rotations:

$$d(R) = [\ldots, R_i\overline{p}_{ik}, \ldots, \ldots, R_i\bar{t}_{ij}, \ldots]^\top.$$

For fixed rotations, the optimal x^* minimizing $\|Ax - d(R)\|^2$ is given by $x^* = A^\dagger d(R)$, where $A^\dagger$ is the pseudoinverse of A. Substituting back, the first two terms reduce to $\|(I - AA^\dagger)d(R)\|^2$, which depends only on rotations. Thus, the original problem becomes:

$$\min_{R_i\in SO(3)} \|(I - AA^\dagger)d(R)\|^2 + \sum_{(i,j)\in\epsilon_o} \|R_j - R_i\overline{R}_{ij}\|_F^2.$$

This is a rotation-only optimization problem with $3N$ variables.

Despite the reduction in variables, the rotation-only problem can be more computationally demanding. The cost function now involves a projection term $\|(I - AA^\dagger)d(R)\|^2$, which requires solving a linear least-squares problem for each evaluation of $d(R)$ (i.e., for each rotation configuration). Although A is sparse, solving this inner loop repeatedly can be expensive. Moreover, the optimization over $SO(3)$ is non-convex, and the Hessian with respect to rotations becomes dense, losing the sparsity that makes standard SLAM solvers efficient. In contrast, the original problem exploits sparsity through linearization and iterative solvers (e.g., Gauss-Newton), where variables can be efficiently marginalized (e.g., via Schur complement). Thus, the rotation-only formulation may not yield computational gains in practice.

II. Team

The datasets were downloaded and executed using the following commands:

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

Fig. 3 shows the runtime output of both systems.

Deliverable 6 - Performance Comparison

Before comparing trajectories, we ran `fix_timestamps.py` to correct the timestamp files. We then used the `evo_traj` tool to generate the initial comparison plot:

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

The results are shown in Fig. 4. Without alignment, the trajectories from blue Kimera and green ORB-SLAM3 exist in separate spatial coordinates. This is expected, as each algorithm initializes its own world frame upon startup. The 3D Trajectory and XYZ plots clearly show different starting points and orientations. The RPY plot indicates a large, constant offset in yaw (around $150°$), while roll and pitch estimates are closer, likely due to gravity alignment. Interestingly, the Speed plot shows that both algorithms produce very similar velocity estimates, suggesting they are both tracking the drone's motion dynamics accurately, despite the coordinate mismatch.

Next, we aligned the trajectories to the ground truth data. First, the ground truth file was converted to TUM format:

```
evo_traj euroc ~/datasets/vnav/MH_01_easy/mav
↪  0/state_groundtruth_estimate0/data.csv
↪  --save_as_tum
```

Then, the comparison was run with alignment enabled:

```
evo_traj tum output/kimera/kimera.txt
↪  output/orbslam/orb_slam3.txt --ref
↪  data.tum --plot --align
```

The aligned results are in Fig. 5. After alignment, both estimated trajectories closely follow the ground truth reference. The 3D and XYZ plots show minimal drift, even during the MAV's complex maneuvers. The RPY plots confirm that both algorithms accurately capture rapid changes in attitude. The Speed plot reveals that while both estimates contain some high-frequency noise—common in IMU-based predictions—they reliably track the overall velocity profile of the ground truth. These results demonstrate that both ORB-SLAM3 and Kimera provide reliable state estimation on the EuRoC dataset.

[Optional] Deliverable 7 - LDSO

We also ran LDSO (LiDAR-Direct-Sparse-Odometry) on the EuRoC dataset. Fig. 2 shows a snapshot from the viewer, displaying the reconstructed sparse 3D point cloud, active keyframes, and the current camera pose. The semi-dense map effectively captures the structural details of the Machine Hall environment.

LDSO is a monocular, direct visual odometry method. A known limitation of monocular systems is scale ambiguity—they cannot recover the absolute metric scale of the world. Therefore, to fairly compare its trajectory against the ground truth and the stereo/VIO systems (Kimera and ORB-SLAM3), we must perform a Sim(3) alignment. This corrects for differences in rotation, translation, and scale. We used the `--correct_scale` flag in `evo` for this purpose.

Fig. 6 presents the comparison results. Several key points stand out:

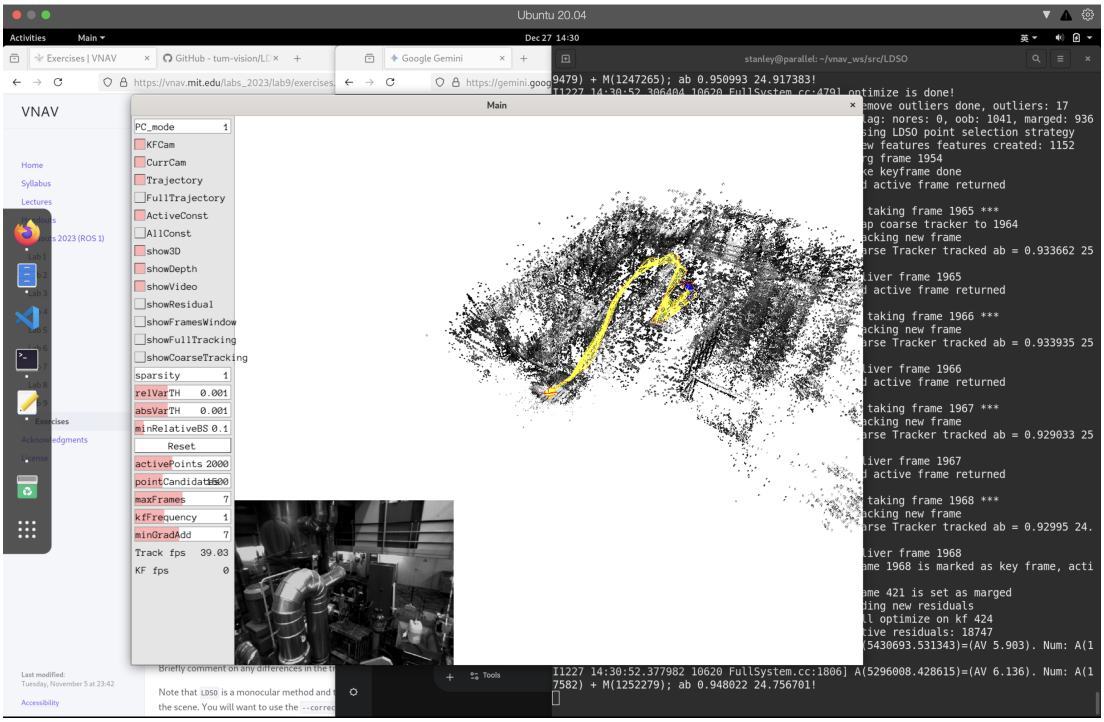


Fig. 2: Runtime snapshot of the LDSO pipeline.

a) Trajectories: After scale correction, the LDSO trajectory (blue, labeled 'results_final') aligns very well with the ground truth, Kimera, and ORB-SLAM3. This shows that even without IMU or stereo data, LDSO's direct photometric optimization can accurately reconstruct the camera's path in a feature-rich environment like EuRoC.

b) XYZ: The LDSO position estimates appear smooth. Direct methods utilize intensity gradients from most pixels, which can lead to robust tracking, though they are also more sensitive to lighting changes and photometric calibration.

c) RPY: LDSO's roll, pitch, and yaw estimates track the ground truth well. Unlike VIO systems which have direct accelerometer measurements to constrain roll and pitch, monocular VO can drift in these axes over time. For this sequence, however, LDSO maintains stable orientation estimates.

d) Speeds: The velocity profile derived from the scaled LDSO trajectory matches those from the VIO pipelines. This indicates good temporal consistency in its pose estimates—the scaled monocular velocities correspond well to real-world motion.

Summary of Team Deliverables

e) Comparison plots of Kimera and ORB-SLAM3 on `MH_01_easy` with comments on differences in trajectories: The "Spy Game" assignment mathematically demonstrated that the full SLAM problem could theoretically be reduced to a rotation-only optimization problem, decreasing the number of variables from $6N + 3M$ to $3N$. However, this simplification comes with a significant computational drawback: the loss of sparsity in the system. Visually, marginalizing landmarks creates dense blocks (fill-in) in the information matrix. While solving a large sparse system typically scales as $O(N)$, solving the reduced but dense system becomes $O(N^3)$. This explains why modern backends (like the g^2o library used in ORB-SLAM and LSD-SLAM) do not naively eliminate all landmarks. Instead, they preserve and exploit the inherent sparse block structure of the Bundle Adjustment problem to maintain real-time efficiency.

f) Plot of Kimera and ORB-SLAM3 trajectories on `MH_01_easy` aligned with ground-truth with comments on any differences in the trajectories: Comparing ORB-SLAM/Kimera (feature-based) with LDSO (direct) illustrates a fundamental design trade-off. Feature-based methods rely on detecting and matching distinctive keypoints (e.g., ORB descriptors), which provides robustness against illumination changes and viewpoint variations. In contrast, direct methods like LDSO optimize for photometric consistency across pixels, which can yield smoother trajectories and denser reconstructions but assumes consistent brightness and is more vulnerable to auto-exposure or lighting variations. While LDSO produced a semi-dense map useful for environmental understanding, ORB-SLAM's sparse map is more computationally efficient for long-term tracking.

g) [Optional] Plot of Kimera + ORB-SLAM3 + LDSO (aligned and scale corrected with ground-truth) with comments on any differences in trajectories: The experiments clearly highlight the limitation of monocular visual odometry: scale ambiguity. While the trajectory shape estimated by monocular LDSO was correct, its absolute scale was arbitrary, requiring Sim(3) alignment (scale correction) for meaningful comparison with ground truth and metric systems

like stereo ORB-SLAM and VIO Kimera. This underscores the critical advantage of sensor fusion (visual-inertial) or stereo vision in robotics applications. For monocular systems, scale drift remains inevitable over long trajectories unless corrected through loop closures with Sim(3) optimization, as implemented in systems like LSD-SLAM.

III. Some words

This would be the last lab of this semaster. We didn't need to compliment any code, only a environment on computer is totally enough. This week is the deadline of three labs: 5, 6, and 9. We also have a signal lab report and a homework deadline this weekend, a presentation of academic English and a presentation of signal two days later, a lab report of computer network and a final exam three days later. It seems like always, things fill up in a few single days. Every months around, it happens over again and again. It somehow reminds me of: "History is a good teacher. If you have not learn anything from him, he will teach you again and again, till you learn something."

That's it. I need to do labs from other courses, now...

References

[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[2] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 834–849.

[3] X. Gao, R. Wang, N. Demmel, and D. Cremers, "LDSO: Direct sparse odometry with loop closure," *arXiv preprint arXiv:1808.01111*, 2018. [Online]. Available: https://arxiv.org/abs/1808.01111

[4] M. I. of Technology. (2023) Mit16.485 - visual navigation for autonomous vehicles. [Online]. Available: https://vnav.mit.edu/labs_2023/lab9/exercises.html

Appendix A
Individual

Deliverable 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: [4]

Deliverable 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.

Hint: consider a graph with an arbitrary topology. Also, think about the problem as a graph, where you fix the "root" node and initialize each node by chaining the edges from the root to that node..

Deliverable 4 [Optional] - Direct methods for SLAM

LSD-SLAM is a direct method for SLAM, as opposed to feature-based methods we have worked with previously. As you know, feature-based methods detect and match features (keypoints) in each image and then use 2-view geometry (and possibly bundle adjustment) to estimate the motion of the robot. Direct methods are different in the fact that they do not extract features, but can be easily understood using the material presented in class.

In particular, the main difference is the way the 2-view geometry is solved. In feature-based approaches one uses RANSAC and a minimal solver (e.g., the 5-point method) to infer the motion from feature correspondences. In direct methods, instead, one tries to estimate the relative pose between consecutive frames by minimizing directly the mismatch of the pixel intensities between two images:

$$E(\xi) = \sum_i (I_{ref}(p_i) - I(\omega(p_i, D_{ref}(p_i), \xi)))^2 = r_i(\xi)^2$$

Where the objective looks for a pose ξ (between the last frame I_{ref} and the current frame I that minimizes the mismatch between the intensity $I_{ref}(pi)$ at pixel p_i for each pixel in the image, and intensity of the corresponding pixel in the current image I. How do we retrieve the pixel corresponding to p_i in the current image I? In other words, what is this term?:

$$I(\omega(p_i, D_{ref}(p_i), \xi))$$

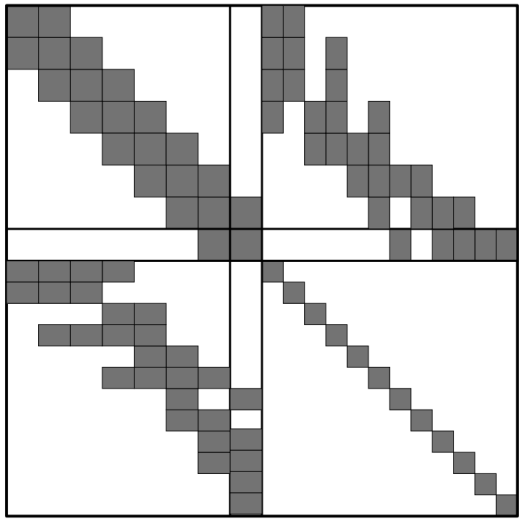
It seems mysterious, but it's nothing new: this simply represents a perspective projection. More specifically, given a pixel p_i, if we know the corresponding depth $D_{ref}(p_i)$ we can get a 3D point that we can then project to the current camera as a function of the relative pose. The ω is typically called a **warp function** since it "warps" a pixel in the previous frame into a pixel at the current frame I_{ref}. What is the catch? Well.. the depth D_{ref} is unknown in practice, so you can only optimize $E(\xi)$ if at a previous step you have some triangulation of the points in the image. In direct methods, therefore these is typically an "initialization step": you use feature-based methods to estimate the poses of the first few frames and triangulate the corresponding points, and then you can use the optimization of $E(\xi)$ to estimate later poses. The objective function $E(\xi)$ is called the photometric error, which quantifies the difference in the pixel appearance in consecutive frames.

NOTE: LDSO (Direct Sparse Odometry with Loop Closures) which you are going to use in Part 2 of this handout is simply an evolution of LSD-SLAM (by the same authors). We are suggesting you to read the LSD-SLAM paper since it provides a simpler introduction to direct methods, while we decided to use LDSO for the experimental part since it comes with a more recent implementation.

Deliverable 5 [Optional] - From landmark-based SLAM to rotation estimation

Consider the landmark-based SLAM problem 1:

$$\min_{t_i \in \mathbb{R}, R_i \in SO(3), p_i \in \mathbb{R}^3} \sum_{(i,k) \in \epsilon_l} \|R_i^\top (p_k - t_i) - \overline{p}_{ik}\|_2^2 + \sum_{(i,j) \in \epsilon_o} \|R_i^\top (t_j - t_i) - \overline{t}_{ij}\|_2^2 + \|R_j - R_i \overline{R}_{ij}\|_F^2 \quad (1)$$



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

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

  Hint: i) Euclidean norm is invariant under rotations, and (ii) translations/positions variables appear . . . ! Consider also using a compact matrix notation to rewrite the sums in the cost function otherwise it will be tough to get an expression of the rotation-only problem.
- 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.

  Hint: What property makes optimization-based SLAM algorithms fast to solve?

APPENDIX B
FIGURES

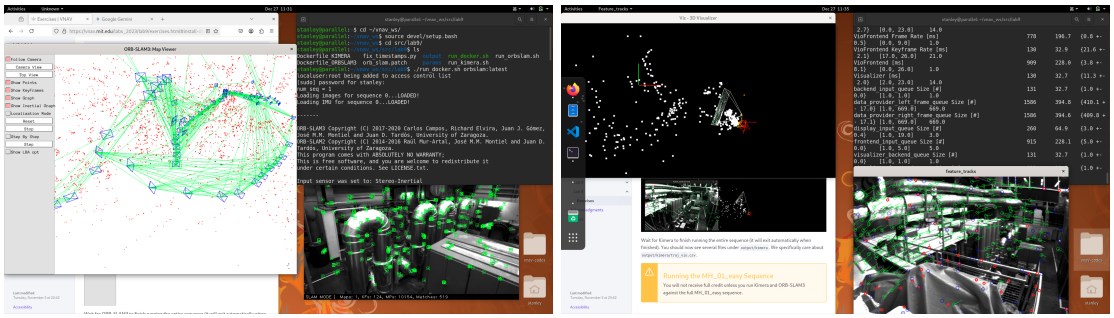


Fig. 3: Runtime snapshots of ORB-SLAM and Kimera.

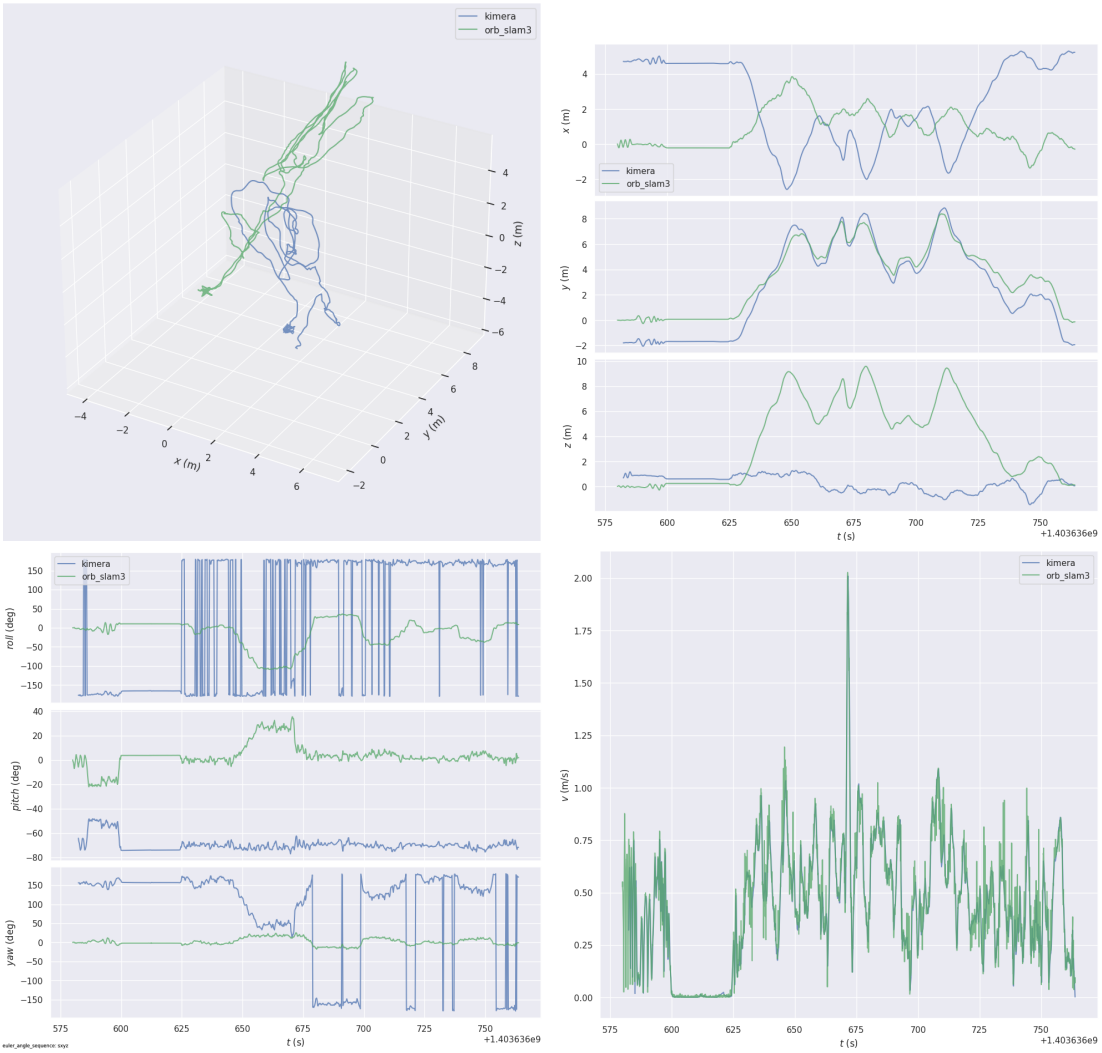


Fig. 4: Comparison of ORB-SLAM3 and Kimera trajectories without alignment.

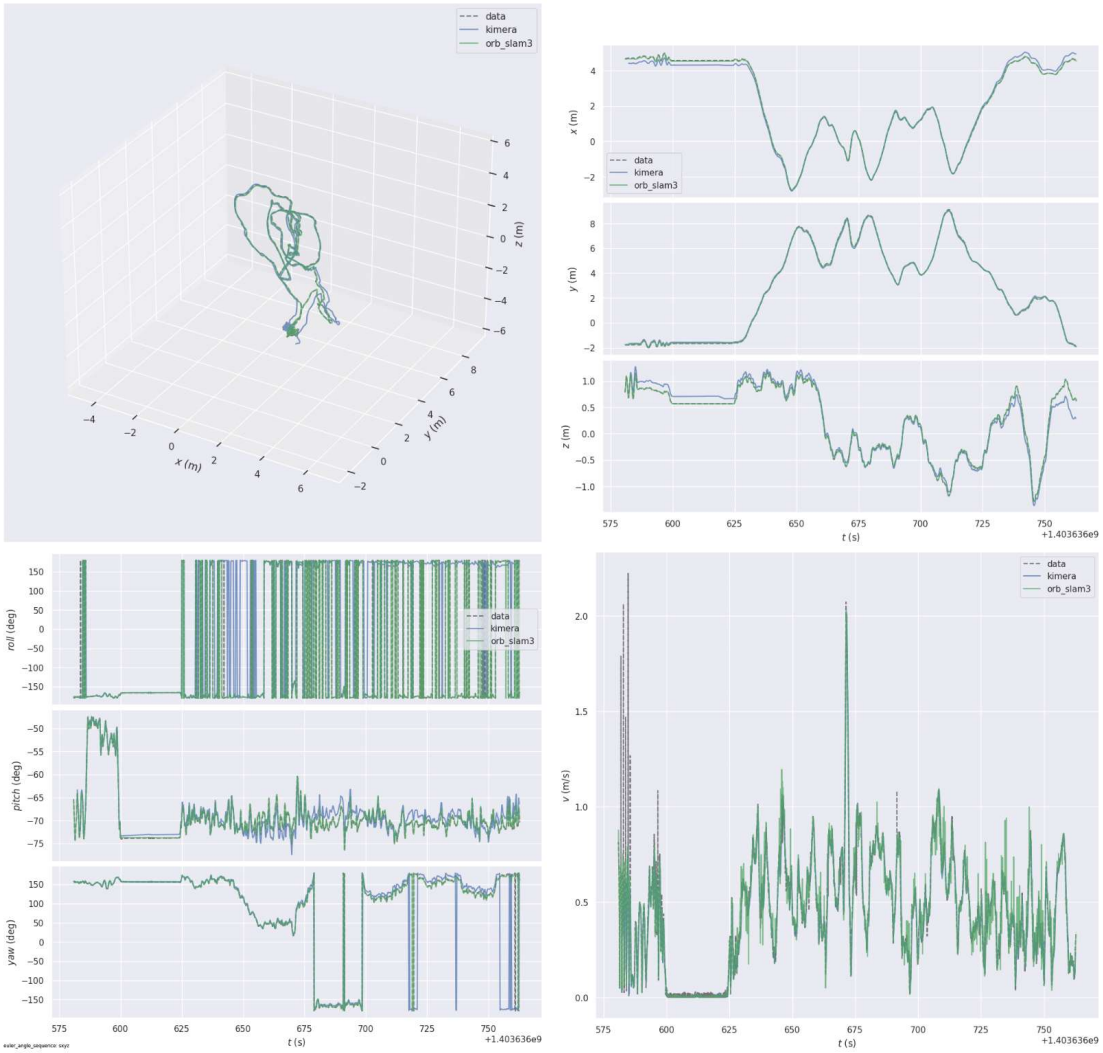


Fig. 5: Comparison of ORB-SLAM3 and Kimera trajectories after alignment to ground truth.

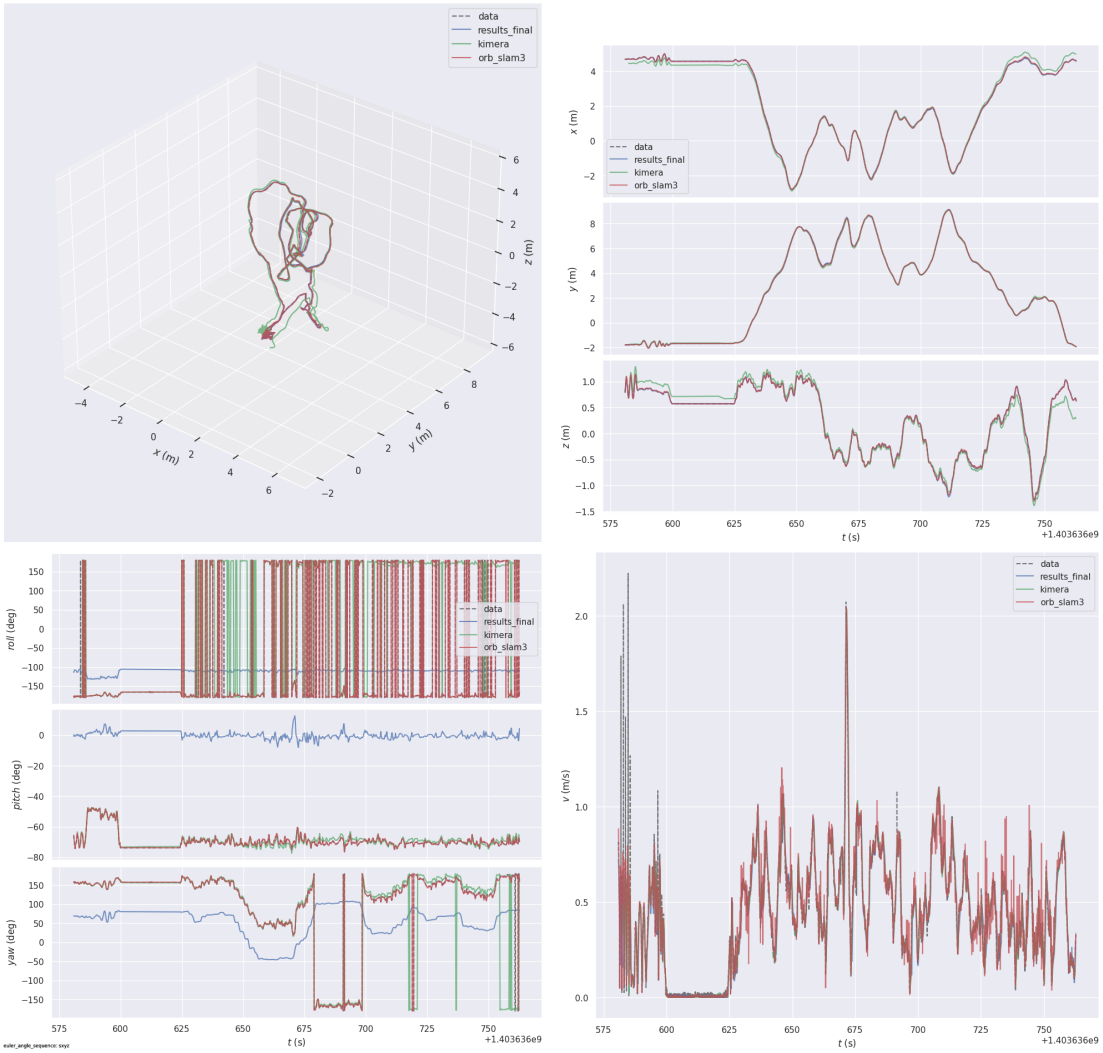


Fig. 6: Trajectory comparison of ORB-SLAM3, Kimera, and LDSO (with scale correction).