Assessing ORB-SLAM3 for VO/VIO and Autonomous Navigation: Freiburg RGB-D and RealSense-Based Robot Experiments

Abu Hanif Muhammad Syarubany (20258259)
Korea Advanced Institute of Science & Technology
Daejeon, South Korea
hanif.syarubany@kaist.ac.kr

***Abstract*—We present an empirical evaluation of ORB-SLAM3 on both a standard VO/VIO benchmark and a simulated mobile robot equipped with an RGB-D sensor. On the Freiburg 2 RGB-D dataset, we systematically vary the number of extracted ORB features and quantify the resulting trajectories using the *evo* toolbox with SE(3) Umeyama alignment and standard APE/RPE metrics. We then transfer the same evaluation pipeline to a RealSense D435i–based robot in Gazebo, where the ORB-SLAM3 configuration is matched to the simulated camera via a dedicated YAML file, ROS camera info topics, and depth-color alignment using nodelets in a visually textured environment. Our results show that ORB-SLAM3 maintains robust performance in a range of feature settings and can provide sufficiently accurate pose estimates for autonomous navigation when camera parameters and sensor alignment are carefully configured.**

***Index Terms*—visual odometry, visual–inertial SLAM, ORB-SLAM3, autonomous navigation.**

I. Introduction

Visual odometry (VO) and visual inertial odometry (VIO) have become key enabling technologies for autonomous navigation in robotics, AR/VR, and mobile computing. Modern feature-based SLAM systems such as ORB-SLAM2 and ORB-SLAM3 deliver real-time, globally consistent trajectories from monocular, stereo, RGB-D, and visual–inertial sensors by combining sparse feature tracking, keyframe-based bundle adjustment, and loop closing [1], [2]. Despite their success, the practical performance of these systems still depends strongly on configuration choices (e.g., number of features, camera calibration, and sensor alignment) as well as on the visual characteristics of the environment. Understanding how these factors affect accuracy and robustness is essential before deploying such systems on real robots.

In this work we experimentally evaluate ORB-SLAM3 on both a standard VO/VIO benchmark and on an autonomous robot navigation scenario. Using the Freiburg 2 RGB-D dataset [3], we study the impact of varying the number of ORB features on trajectory accuracy. We then transfer the same pipeline to a RealSense D435i, equipped mobile robot in simulation, where the camera parameters are carefully matched to the robot model and depth–color alignment is handled via a ROS nodelet chain [4]. In both cases, trajectories are assessed using the *evo* toolbox [5] with SE(3) Umeyama alignment and standard APE/RPE metrics, allowing a consistent comparison between dataset-based evaluation and a realistic navigation setting.

II. Related Work

Feature-based visual SLAM has been extensively studied over the past decade. ORB-SLAM2 popularized a keyframe-based formulation with ORB features, loop closing, and global bundle adjustment that works with monocular, stereo, and RGB-D cameras in real time [1]. Building on this, ORB-SLAM3 extends the framework to visual-inertial and multi-map SLAM, supporting monocular, stereo, RGB-D, and fisheye cameras, and introducing tightly coupled IMU initialization and multi-map fusion [2]. These systems remain among the most widely used baselines in VO/VIO research and are often deployed as “off-the-shelf” solutions in robotics applications.

A number of benchmarks have been introduced to systematically evaluate VO/VIO and SLAM methods. The TUM RGB-D dataset provides synchronized RGB-D sequences with accurate motion capture ground truth and standardized evaluation metrics for APE and RPE [3]. The EuRoC MAV dataset focuses on visual inertial SLAM for micro aerial vehicles with stereo cameras and IMU measurements in challenging indoor environments [6]. The KITTI odometry benchmark targets outdoor driving scenarios with long trajectories and emphasizes translational and rotational drift over subsequences [7]. These datasets have become de facto standards for comparing SLAM systems across a wide range of trajectories and sensor configurations.

Trajectory evaluation toolchains have also matured. The *evo* package [8] provides a generic framework to parse trajectories in common formats (TUM, KITTI, ROS bags), perform SE(3) Umeyama alignment to remove global frame differences, and compute APE/RPE statistics and visualizations [5]. Many recent works adopt a similar evaluation methodology for aligning estimated trajectories to ground truth, reporting summary error metrics, and analyzing temporal error profiles, which we follow in this paper for both the Freiburg 2 dataset and our RealSense-based robot simulation.

III. Methodology

This section describes the experimental pipeline used to evaluate ORB-SLAM3 on the Freiburg 2 dataset [9] and on the robot simulation. We first explain the trajectory evaluation with the *evo* toolbox, then detail the SLAM configurations for the TUM and RealSense datasets, and finally describe the simulation setup and sensor alignment.

A. Trajectory Evaluation with evo

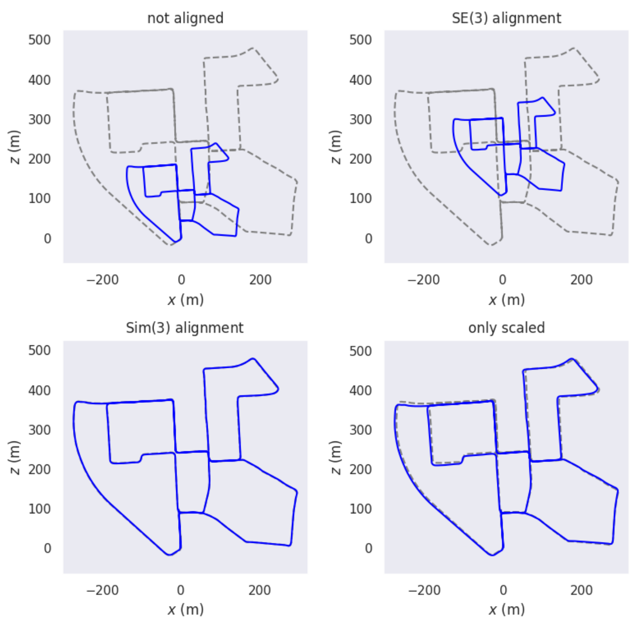


Fig. 1: SE(3) Umeyama Alignment Demonstration.

For all experiments, both the ground-truth and ORB-SLAM3 trajectories were exported in TUM format. The trajectories were processed with the *evo* toolbox, which first associates poses by timestamp and then performs an SE(3) Umeyama alignment (as illustrated in Figure 1). The Umeyama algorithm computes the rigid transform (rotation, translation, and uniform scale if needed) that best aligns the estimated trajectory to the ground-truth in the least-squares sense. After alignment, the remaining discrepancies between the two trajectories correspond to drift and local estimation errors rather than arbitrary coordinate-frame differences.

On the aligned trajectories we computed the Absolute Pose Error (APE) and Relative Pose Error (RPE) for translation and rotation. APE captures global consistency over the full sequence, while RPE measures local motion errors between consecutive poses. For each metric we recorded the full time series and summary statistics (max, mean, median, min, RMSE, SSE, and standard deviation), and visualized them in the plots shown in Figs. 4 and 5.

B. TUM2 Configuration and ORB Feature Tuning

For the Freiburg 2 experiments we used the standard ORB-SLAM3 configuration file `TUM2.yaml` as provided by the authors. All camera intrinsics, distortion coefficients, depth scale, and image resolution were kept unchanged, i.e., exactly as specified for the TUM RGB-D sequences. The only parameter we modified was the number of ORB features extracted per frame (`ORBextractor.nFeatures`).

By varying this value between 600 and 1500 we studied how the density of features affects the robustness and accuracy of the estimated trajectory. For each setting we ran ORB-SLAM3 on the same Freiburg 2 sequence, exported the resulting trajectory, and evaluated it with *evo* as described above. The corresponding error statistics are reported in Table I.

C. RealSense_D435i Configuration for the Robot

In the robot experiments ORB-SLAM3 was configured using a dedicated `RealSense_D435i.yaml` file. The goal was to make the SLAM configuration match the simulated RealSense sensor specified in the robot xacro description. To obtain the correct camera parameters we launched the robot and RealSense drivers in ROS and inspected the camera calibration reported on the `/camera/{color,depth}/camera_info` topics using `rostopic echo`.

The intrinsic matrix (f_x, f_y, c_x, c_y), distortion coefficients, image resolution, and frame rate from the `camera_info` messages were then copied into `RealSense_D435i.yaml`. Depth scale and clipping distances were also set to match the RealSense model. This ensured that ORB-SLAM3 received images with geometry consistent with both the simulated sensor and the configuration file.

To guarantee spatial alignment between RGB and depth images, we used the nodelet package to rectify the depth

TABLE I: Trajectory and Relative Pose Errors from FR2 Dataset vs Number of ORB Features.

Metrics	**max**	**mean**	**median**	**min**	**rmse**	**sse**	**std**
nFeatures = 600							
Average Precision Error (APE)	2.71016	0.53537	0.49290	0.01766	0.60159	334.03829	0.27438
Relative Pose Error - Translation	2.72403	0.04782	0.01599	0.00089	0.23786	35.87004	0.23300
Relative Pose Error - Rotation	2.82786	0.06587	0.02643	0.00234	0.28662	52.08219	0.27894
nFeatures = 800							
Average Precision Error (APE)	2.89121	0.35472	0.26786	0.00675	0.47844	211.50504	0.32105
Relative Pose Error - Translation	2.38177	0.03502	0.01605	0.00081	0.15315	13.81467	0.14909
Relative Pose Error - Rotation	2.82809	0.04590	0.02650	0.00163	0.19210	21.73543	0.18653
nFeatures = 1000							
Average Precision Error (APE)	3.11202	0.23661	0.21326	0.00754	0.30364	87.49728	0.19031
Relative Pose Error - Translation	2.60779	0.04174	0.01690	0.00170	0.20054	26.38115	0.19615
Relative Pose Error - Rotation	2.82700	0.05441	0.02585	0.00264	0.24458	39.24284	0.23845
nFeatures = 1200							
Average Precision Error (APE)	3.90487	0.36727	0.28930	0.00779	0.46780	207.45385	0.28974
Relative Pose Error - Translation	4.34524	0.05659	0.01768	0.00162	0.30760	62.06937	0.30235
Relative Pose Error - Rotation	2.82832	0.06247	0.02562	0.00339	0.28801	54.41561	0.28116
nFeatures = 1500							
Average Precision Error (APE)	3.22448	0.35848	0.30401	0.00470	0.44330	186.68489	0.26078
Relative Pose Error - Translation	2.94542	0.03879	0.01771	0.00180	0.18244	21.36919	0.17827
Relative Pose Error - Rotation	2.82843	0.04840	0.02728	0.00263	0.20463	26.88182	0.19882

APE: Absolute trajectory error (m). **Relative Pose Error**: translation (m), rotation (rad).

TABLE II: Trajectory and Relative Pose Errors from robot simulation vs. Number of ORB Features.

Metrics	**max**	**mean**	**median**	**min**	**rmse**	**sse**	**std**
nFeatures = 600							
Average Precision Error (APE)	0.51012	0.29864	0.32522	0.00237	0.31947	1244.65145	0.11349
Relative Pose Error - Translation	0.10666	0.00878	0.00561	0.00032	0.01275	0.20203	0.00924
Relative Pose Error - Rotation	0.08300	0.00544	0.00351	0.00018	0.00810	0.08149	0.00600
nFeatures = 800							
Average Precision Error (APE)	0.43626	0.29399	0.32949	0.00049	0.31343	1180.43479	0.10868
Relative Pose Error - Translation	0.05611	0.00584	0.00472	0.00031	0.00708	0.05115	0.00400
Relative Pose Error - Rotation	0.05397	0.00517	0.00301	0.00012	0.00774	0.06121	0.00577
nFeatures = 1000							
Average Precision Error (APE)	0.67380	0.37282	0.42955	0.00021	0.41252	2051.92484	0.17657
Relative Pose Error - Translation	0.02853	0.00660	0.00550	0.00021	0.00769	0.05475	0.00395
Relative Pose Error - Rotation	0.07912	0.00651	0.00387	0.00007	0.00977	0.08837	0.00729
nFeatures = 1200							
Average Precision Error (APE)	0.79343	0.37257	0.26463	0.00051	0.43673	2070.82160	0.22787
Relative Pose Error - Translation	0.02983	0.00832	0.00642	0.00071	0.01010	0.09018	0.00572
Relative Pose Error - Rotation	0.09103	0.00820	0.00469	0.00014	0.01244	0.13704	0.00936
nFeatures = 1500							
Average Precision Error (APE)	0.41563	0.19260	0.18361	0.00019	0.21376	429.09058	0.09273
Relative Pose Error - Translation	0.04901	0.00859	0.00652	0.00050	0.01072	0.10370	0.00641
Relative Pose Error - Rotation	0.09643	0.00886	0.00471	0.00013	0.01393	0.17495	0.01075

APE: Absolute trajectory error (m). **Relative Pose Error**: translation (m), rotation (rad).

image to be aligned with the color frame. In this setup, the nodelet chain produces a depth image already aligned with the color image before it is forwarded to ORB-SLAM3, so that extracted ORB features always have a consistent depth measurement.

D. *Simulation Environment and Visual Texture*

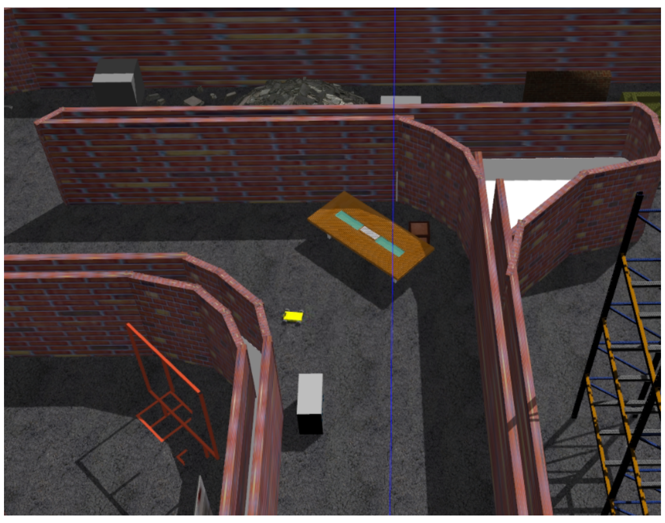


Fig. 2: Gazebo world modification.

The robot experiments were carried out in a Gazebo-based simulation. The virtual RGB-D camera in the simulator used the same intrinsics, distortion model, resolution, and frame rate as specified in `RealSense_D435i.yaml` and observed via the RealSense ROS drivers. This tight coupling between the xacro description, the simulated camera, and the SLAM configuration reduces systematic calibration errors and makes the evaluation more representative of a real deployment.

To support reliable feature extraction, we designed the Gazebo world with visually rich textures (as illustrated in Figure 2). Walls, floors, and objects were assigned non-uniform, high-frequency textures instead of flat colors. This increases the number and distinctiveness of ORB features detected in the images, which in turn improves tracking robustness and reduces pose drift. The trajectories generated in this simulated environment were logged, aligned with the robot's ground-truth pose using *evo*, and evaluated with the same APE and RPE metrics as in the Freiburg 2 experiments.

IV. Experiments and Results

A. ***Assignment 1:*** *Run ORB-SLAM3 with VIO Dataset*

In this experiment we ran ORB-SLAM3 on the VIO sequence of the Freiburg 2 dataset and inspected the result in RViz. The RGB-D frames were fed to the system with the default configuration, and we visualized the reconstructed sparse map points together with the ORB features tracked over time. The estimated camera/robot trajectory from ORB-SLAM3 (blue) was overlaid on top of the ground-truth path (orange) provided with the dataset, as shown in the Figure 3.

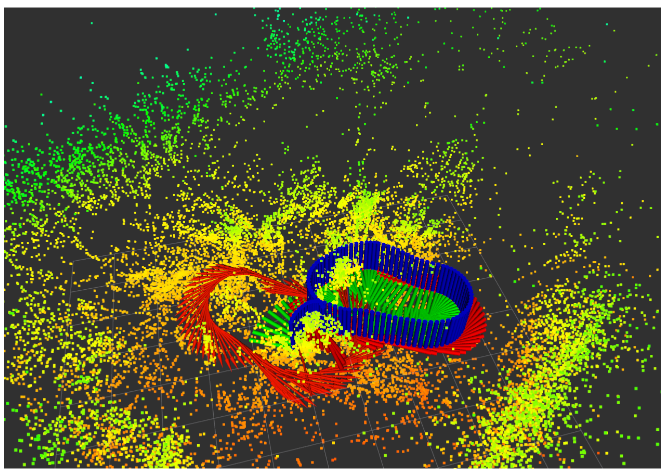
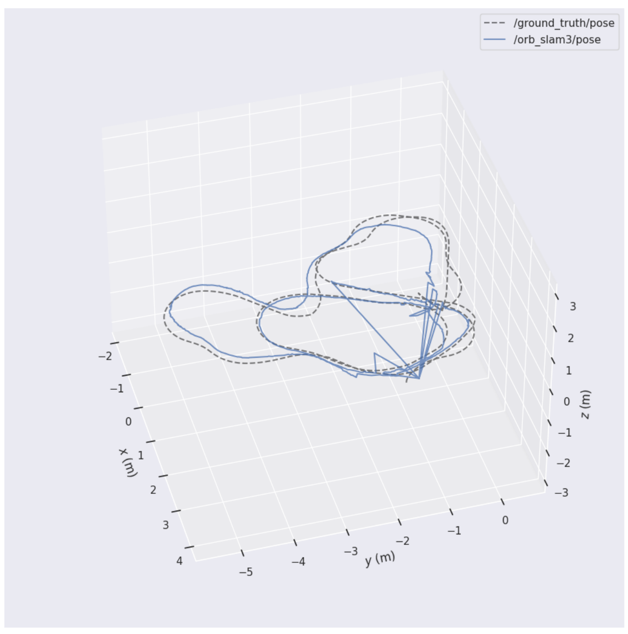


Fig. 3: RViz visualization of the ORB SLAM robot trajectory and its groundtruth pose in Freinburg 2 dataset.

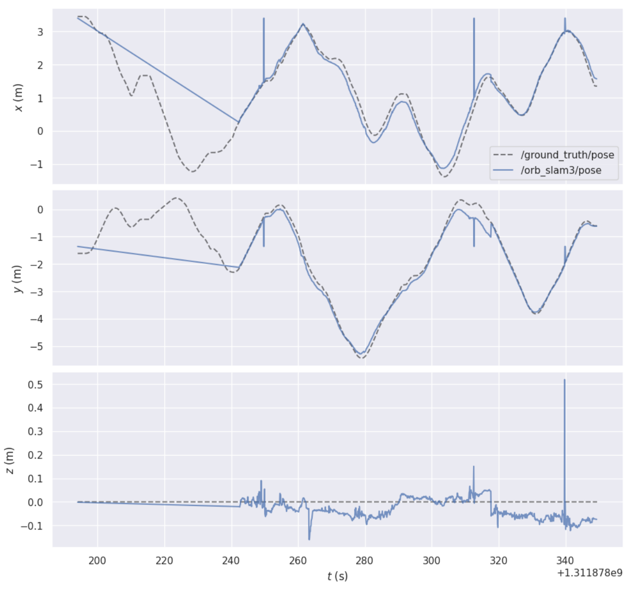
Qualitatively, the visualization reveals that the SLAM trajectory does not coincide with the ground truth: the estimated path drifts away and exhibits a different shape, even though it roughly follows the same loop. This indicates that, under our current settings, ORB-SLAM3 fails to maintain a globally consistent pose estimate for this sequence. In the following subsections we further analyze this discrepancy using quantitative error metrics (APE and RPE) and investigate parameter settings and conditions under which the tracking performance can be improved.

B. ***Assignment 2:*** *VO/VIO Dataset Evaluation*

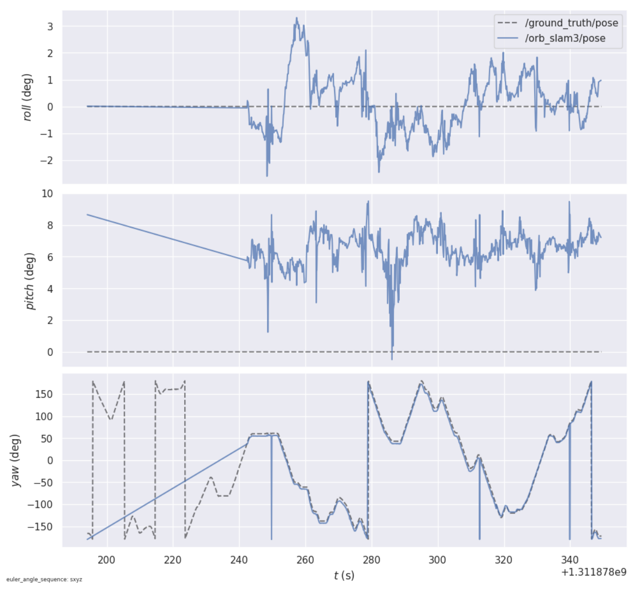
We quantitatively evaluate ORB-SLAM3 on the Freiburg 2 VO/VIO sequence by comparing the estimated camera pose against the provided ground-truth trajectory. After time alignment and SE(3) alignment using the evo toolbox, we compute the Absolute Pose Error (APE) as well as the Relative Pose Error (RPE) in translation and rotation. Table I reports summary statistics when varying the number of extracted



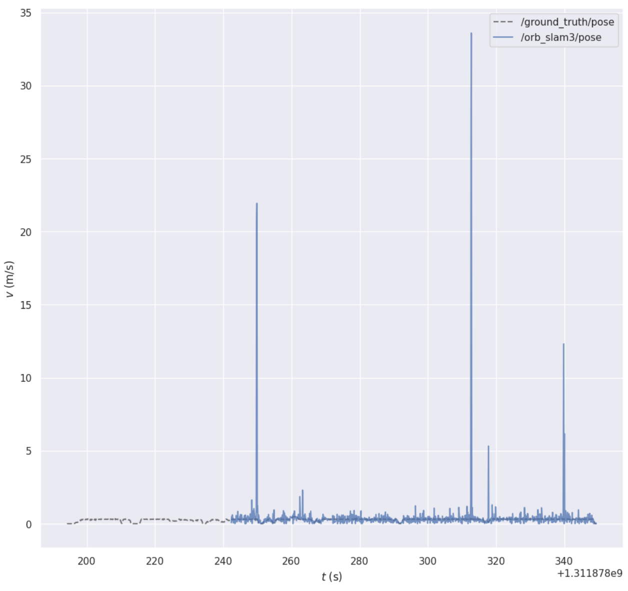
(a) Trajectories



(b) Position (X–Y–Z)

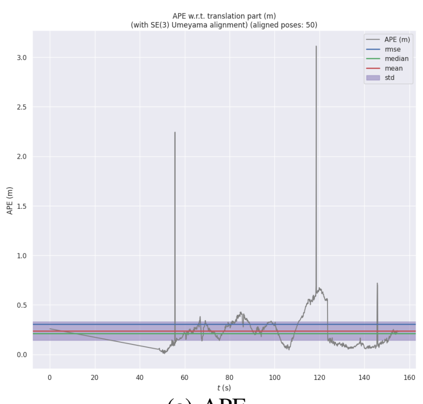


(c) Orientation (R–P–Y)

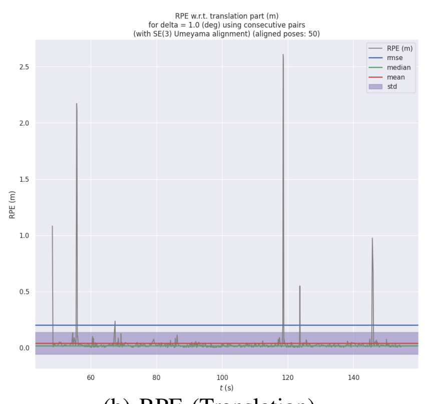


(d) Speeds

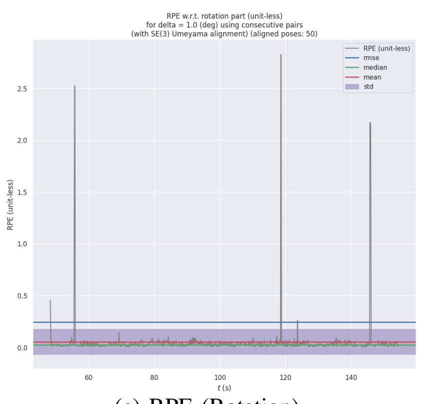
Fig. 4: Trajectory, position, orientation, and speed profiles for the evaluated FR2 sequence (nFeatures = 1000).



(a) APE

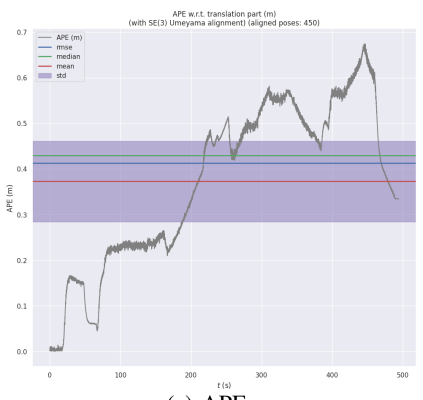


(b) RPE (Translation)

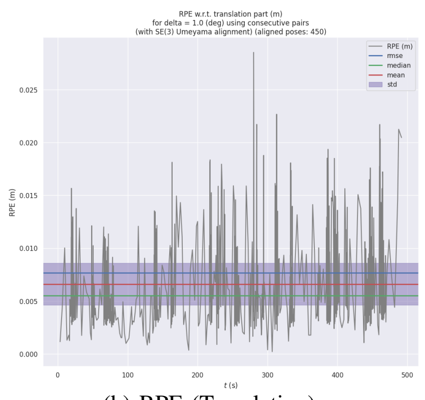


(c) RPE (Rotation)

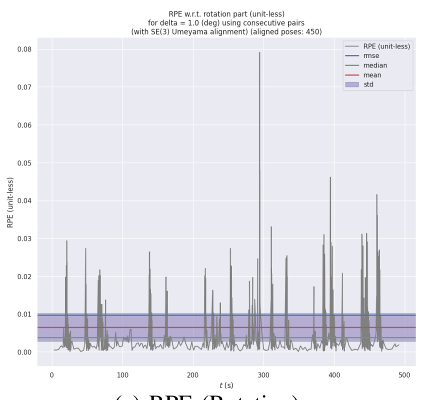
Fig. 5: Absolute and relative pose error statistics for the FR2 sequence (nFeatures = 1000).



(a) APE



(b) RPE (Translation)



(c) RPE (Rotation)

Fig. 6: Absolute and relative pose error statistics for the robot simulation sequence (nFeatures = 1000).

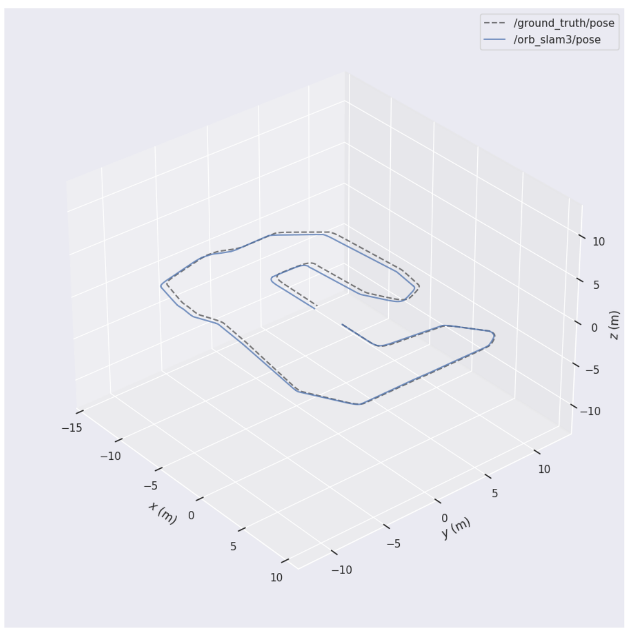
ORB features from 600 to 1500. Overall, the method achieves sub-metre absolute accuracy on this challenging handheld sequence: the mean APE ranges from 0.19–0.37 m, while the mean translational RPE remains on the order of a few millimetres per frame and the mean rotational RPE stays below 0.01 rad. Increasing the number of features generally improves global consistency (lowest APE and RMSE at $n = 1500$), whereas $n = 800$ yields the smallest local RPE, suggesting a trade-off between dense feature tracking and susceptibility to outliers.

The qualitative behavior is illustrated in Figure 4. The 3D trajectories show that ORB-SLAM3 follows the overall loop of the ground-truth path but still exhibits noticeable drift, particularly in height and at sharp turns. The per-axis position and orientation plots reveal slow accumulation of bias as well as occasional spikes, which correlate with peaks in the estimated linear speed; these correspond to rapid motions and texture-poor regions where tracking becomes less reliable. Figure 5 further visualizes the temporal evolution of APE and RPE. While most of the sequence lies within a narrow error band around the mean, several pronounced peaks appear, indicating short intervals of degraded tracking that dominate the maximum and SSE statistics. Taken together, these results show that ORB-SLAM3 can robustly track the FR2 sequence with reasonable accuracy, but its pose estimate is still imperfect and sensitive to fast motions and challenging viewpoints.

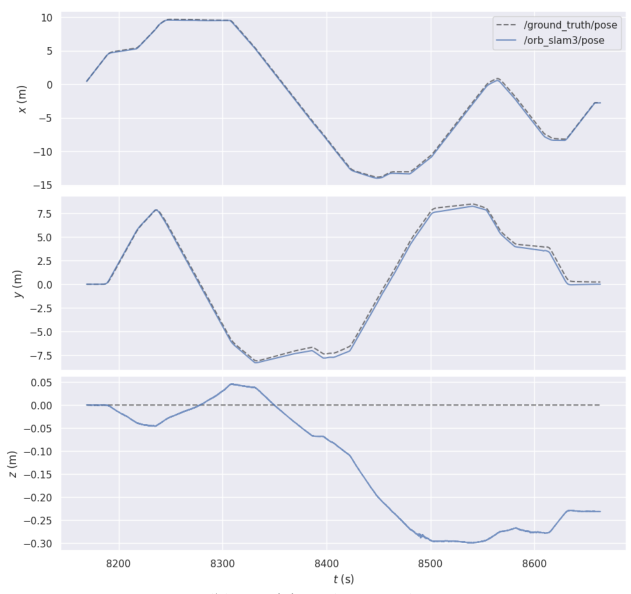
C. *Assignment 3: Autonomous Navigation with ORB-SLAM3*

To assess the suitability of ORB-SLAM3 for autonomous navigation, we integrated it into a differential-drive robot simulation and compared the estimated camera pose against the robot ground-truth trajectory. The robot executed a closed-loop path while ORB-SLAM3 ran in real time using different numbers of ORB features. After time and SE(3) alignment, we computed APE and translational/rotational RPE for the entire run. The quantitative results are summarized in Table II. Across all configurations, the mean APE remains below 0.38 m, with the lowest values obtained for $n = 800$ and $n = 1500$. The mean translational RPE is on the order of 10^{-3}–10^{-2} m, while the mean rotational RPE stays below 0.01 rad, indicating that local motion estimates between consecutive frames are very accurate. Increasing the number of features generally reduces the APE and RPE slightly, but the differences are modest, suggesting that ORB-SLAM3 is relatively robust to this parameter in the tested range.

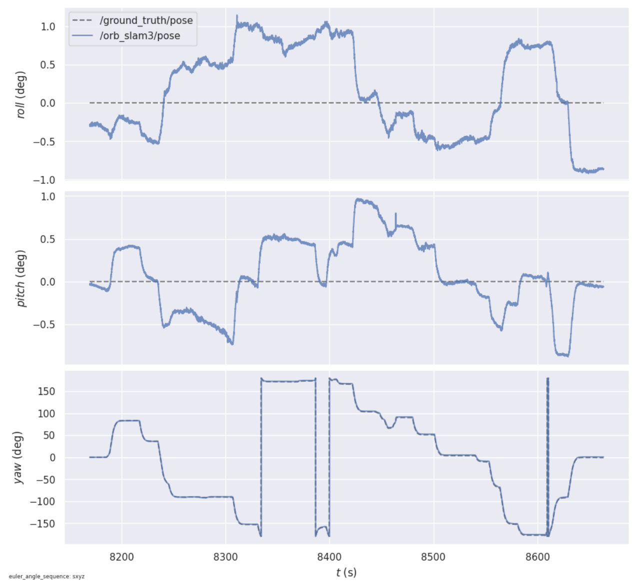
The spatial and temporal behavior of the estimator for the nominal configuration ($n = 1000$ features) is illustrated in



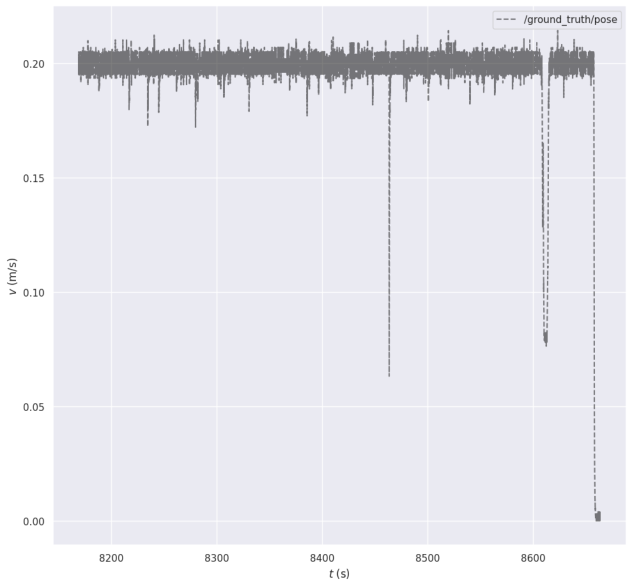
(a) Trajectories



(b) Position (X–Y–Z)



(c) Orientation (R–P–Y)



(d) Speeds

Fig. 7: Trajectory, position, orientation, and speed profiles for the evaluated robot simulation sequence (nFeatures = 1000).

Figure 7. The 3D trajectory plot shows that the estimated path closely follows the ground-truth loop with only small deviations. The per-axis position and orientation plots reveal that most of the run exhibits small biases and smooth evolution, while noticeable discrepancies appear mainly during rapid turns and segments with higher linear velocity, as seen in the speed profile. Figure 6 further visualizes the evolution of APE and RPE over time. The errors remain within a narrow band for the majority of the trajectory, with a few pronounced peaks corresponding to the aforementioned dynamic manoeuvres. Overall, these results indicate that ORB-SLAM3 is capable of providing sufficiently accurate and stable pose estimates for autonomous navigation in this robotic scenario, while highlighting the periods of aggressive motion where additional sensor fusion or motion constraints could further improve robustness.

V. Conclusion

In this work we presented an empirical study of ORB-SLAM3 on both a standard VO/VIO benchmark and a robot navigation scenario. Using the Freiburg 2 RGB-D dataset, we systematically varied the number of ORB features and quantified the resulting trajectories with APE and RPE metrics after SE(3) Umeyama alignment using the *evo* toolbox. The results show that ORB-SLAM3 maintains sub-metre global accuracy across a wide range of feature counts, with a clear trade-off between global consistency and local motion smoothness as the number of extracted features is changed.

We then transferred the same evaluation pipeline to a mobile robot equipped with RealSense D435i in simulation, where the SLAM configuration was matched to the simulated sensor through the RealSense_D435i YAML file, ROS camera_info messages, and depth–color alignment using nodelets. In a visually textured Gazebo environment, ORB-SLAM3 produced trajectories that closely followed the robot ground truth and remained accurate even during relatively aggressive manoeuvres, with error peaks localized to a few challenging segments. Overall, our findings indicate that ORB-SLAM3 is a viable backbone for autonomous navigation provided that sensor parameters and alignment are carefully configured, and they highlight how small design choices, such as feature density and scene texture, can significantly influence the final pose accuracy.

References

[1] R. Mur-Artal and J. D. Tardós, “ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras,” *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[2] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, “ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM,” *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[3] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of RGB-D SLAM systems,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 573–580.

[4] W. Garage and R. Community, “nodelet: Ros nodelet package,” https://wiki.ros.org/nodelet, accessed: 2025-11-09.

[5] M. Grupp, “evo: Python package for the evaluation of odometry and SLAM,” https://github.com/MichaelGrupp/evo, 2017, accessed: 2025-11-09.

[6] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, “The EuRoC micro aerial vehicle datasets,” *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[7] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the KITTI vision benchmark suite,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3354–3361.

[8] M. Grupp, “evo_traj: Trajectory evaluation tool in the *evo* package,” https://github.com/MichaelGrupp/evo/wiki/evo_traj, 2017, accessed: 2025-11-09.

[9] “Tum rgb-d dataset,” https://cvg.cit.tum.de/data/datasets/rgbd-dataset/download, accessed: 2025-11-09.