ME5413 Autonomous Mobile Robot

Homework 2 Localization

Group 4

Full Name	Student ID
LHW	1
MYC	1
Simon Kenneth	1

Master of Science in Robotics

COLLEGE OF DESIGN AND ENGINEERING

Mar 14th, 2024

1 Task1

1.1 Introduction of Cartographer

Cartographer is a Google-developed open-source library for 2D and 3D SLAM, ideal for mobile robots using LiDAR data. It offers real-time map updates and accuracy improvements through advanced techniques, and integrates well with ROS, enhancing navigation and path planning in dynamic environments. It uses graph optimization technology to correlate the collected sensor data in time and space, and continuously optimizes and updates the robot's location estimation and map construction by minimizing the error between all the observed data.

1.2 Running process

The Cartographer project has made available two distinct open-source repositories: *cartographer* and *cartographer_ros*. The *cartographer* repository is dedicated to implementing the core functionalities of the SLAM algorithm and generates library files essential for its operation. Building *cartographer* necessitates compiling several dependencies beforehand, including the ceres-solver. Given that the ceres-solver is currently under development while the Cartographer project has ceased maintenance, it is necessary to revert the ceres-solver to version 1.13.0 to circumvent compilation errors.

On the other hand, *cartographer_ros* builds upon the foundational *cartographer* library and is packaged as a collection of ROS packages. This encapsulation facilitates the integration of ROS communication capabilities, enabling the subscription to Lidar scan and odometry messages, among other features.

1.3 Improvement

The Cartographer_ROS package provides a convenient parameter tuning mechanism through a *lua* file, enabling easy adjustment of trajectory builder, pose graph, and other parameters. To enhance SLAM performance, we explored various parameter configurations. The statistical results are presented in Tab. 1.

Experimentation reveals that odometry prediction accuracy is significantly influenced by *TRAJECTORY_BUILDER_2D.submaps.num_range_data*, which dictates how range data (such as laser or radar scans) is aggregated to form a single submap in the 2D SLAM process. A lower value means that submaps will be created more frequently with less data, potentially leading to a higher resolution map but at the cost of increased computational load and potentially more noise in the map. Conversely, a higher value means less frequent submap creation with more data, reducing noise at the expense of possibly overlooking detailed environmental features. However, the benefits of increasing this parameter diminish over time, adversely affecting map quality.

The *optimize_every_n_nodes* parameter also significantly impacts odometry accuracy by determining the frequency of sparse pose adjustment optimization; a lower value indicates more frequent optimization. When the *num_range_data* is small, lower *optimize_every_n_nodes* will achieve better precision.

Parameters related to online correlative scan matching, such as linear/angular search windows and translation/rotation delta cost weights, have a minor effect on SLAM performance. Our findings suggest that emphasizing translation delta cost yields better outcomes, indicating a preference for translation

accuracy in matching, as straight-line terrain features predominate in the given environments. Increasing the search windows will significantly increase the computation payload while the pose estimation will even worsen.

Finally, our optimal parameter combination achieved 0.151 for APE Mean, and 0.16 for APE RMSE, as illustrated in Fig. 2. The corresponding mapping result in Rviz is shown in Fig. 1.

1.4 Bonus Task

We utilize the evo tool [1] for evaluating the estimated odometry against the ground truth data. The default configuration doesn't provide subscribable estimated odometry data, only with the *tf* from the map frame to odom frame, which poses challenges in comparing between the estimation value and the ground truth value.

To streamline the evaluation workflow, we configure the *publish_tracked_pose* parameter (exists in the source code but not mentioned in the tutorial documentation) to *true* within the *lua* configuration file. This adjustment facilitates the automatic publishing of Cartographer's estimated odometry topics with respect to the map frame. We recorded a ROS bag that encapsulates both the estimated odometry and ground truth data to make evo evaluation easier. Figure 2 depicts the outcomes of the evo evaluation.

2 Task2

2.1 Introduction to Employed SLAM Algorithms

SLAM (Simultaneous Localization and Mapping) remains a cornerstone in robotics research, continuously enriched by innovative methodologies. These algorithms are broadly categorized into: vision-based, LiDAR-based, and multi-sensor fusion approaches.

This assignment evaluates **10** SLAM algorithms across these classifications, utilizing the 'evo' tool [1] for assessing Absolute Pose Error (APE) and Relative Pose Error (RPE). Detailed execution methodologies for each algorithm are presented in Appendix B.

2.1.1 Vision-Based SLAM

1. **ORB-SLAM3**

   ORB-SLAM3 [2] presents a versatile SLAM framework supporting monocular, stereo, and RGB-D inputs, using both pin-hole and fisheye lenses. It introduces an advanced visual-inertial integration using Maximum-a-Posteriori estimation, significantly enhancing real-time performance and accuracy by 2 to 5 times compared to preceding versions. Its novel multi-map system improves place recognition and allows efficient data reutilization, including sophisticated bundle adjustment techniques. Empirical validations are provided in Figures 4 and 5.

2. **Basalt**

   Basalt [3] advances visual-inertial odometry (VIO) with non-linear factor recovery for precise, robust mapping. It merges camera and inertial measurements to achieve global map consistency,

addressing challenges such as inertial data degradation. By incorporating non-linear factors from VIO trajectories with loop-closing constraints, Basalt significantly enhances map accuracy. Refer to Figures 6 and 7 for empirical results.

2.1.2 LiDAR-Based SLAM

1. **LeGO-LOAM**

   LeGO-LOAM [4] introduces a real-time, ground-optimized lidar odometry and mapping method, offering six degree-of-freedom pose estimation for ground vehicles. It employs ground plane segmentation and feature extraction for real-time efficiency on low-power systems. The method's accuracy, compared favorably with state-of-the-art LOAM, is documented in Figures 10 and 11.

2. **A-LOAM**

   A-LOAM [5] enhances the original LiDAR Odometry and Mapping (LOAM) methodology [6], focusing on engineering efficiency and ease of further development. It addresses asynchronous range measurements and minimizes point cloud misregistration, achieving low drift and high computational efficiency. Results are showcased in Figures 12, 13, and 14.

3. **KISS-ICP**

   KISS-ICP [7], a streamlined odometry estimation system, diverges from the trend of complicating sensor-based odometry by paring down to essential components, achieving surprising efficacy. It supports diverse environmental conditions and LiDAR sensors without the need for extensive parameter tuning. KISS-ICP utilizes point-to-point ICP, adaptive thresholding for correspondence matching, a robust kernel, a straightforward motion compensation technique, and a point cloud subsampling strategy. Performance is detailed in Figures 15 and 16.

2.1.3 Multi-Sensor Fusion-Based SLAM

1. **V-LOAM**

   V-LOAM [8] achieves real-time odometry and mapping by leveraging 2-axis lidar in 6-DOF, addressing asynchronous data acquisition challenges. It operates without the need for high-precision inputs, achieving low drift and high computational efficiency. Documentation of its performance is available in Figures 17 and 18.

2. **FAST-LIO2**

   FAST-LIO2 [9] introduces a swift, robust, and versatile LiDAR-inertial odometry framework, enhancing navigation and mapping accuracy without feature extraction by directly registering raw points to the map. This approach leverages subtle environmental features, improving accuracy and adaptability across various LiDAR technologies. Its second innovation, the ikd-Tree data structure, allows for efficient map updates and dynamic re-balancing, outperforming traditional dynamic data structures in speed and downsampling capabilities. Its performance is illustrated in Figures 19 and 20.

2.1.4 Learning-Based SLAM

1. **DF-VO**

   Despite the traditional knowledge-based methods, we also tried an innovative learning-based SLAM method. DF-VO [10] innovatively combines multi-view geometry and deep learning to create a robust Visual Odometry (VO) system that addresses the challenges of dynamic, low-texture scenes and scale-drift issues inherent in monocular methods. This system leverages deep neural networks for self-supervised learning of scene depths and camera movements, achieving scale-consistent predictions across long videos. By sampling high-quality correspondences from deep optical flows and integrating them with geometrically triangulated depths, DF-VO accurately recovers camera poses and aligns depths to mitigate scale drift, even in dynamic environments. Figures 8 and 9 detail the empirical results.

2.2 Analysis

According to the experiments, Basalt [3] exhibits remarkable performance with the lowest RMSE of 2.94, highlighted in green, indicating superior consistency and reliability in odometry estimation. This algorithm, hence, demonstrates a commendable balance between accuracy and computational efficiency, underpinned by a relatively low standard deviation of 1.11, which signifies minor fluctuations in error magnitude across different scenarios.

Conversely, FAST-LIO2 [9] records the highest RMSE value of 13.19, denoted in red, reflecting significant challenges in maintaining consistent localization accuracy. This could be attributed to its vulnerability in handling dynamic environments or complex scenes, underscored by an exceptionally high Max error of 24.21 and a substantial Std of 3.88, pointing towards pronounced variability in performance.

Moreover, SDV-LOAM [9] and VINS-GPS [11] manifest considerable discrepancies in Max errors (36.51 and 11.85, respectively), suggesting potential limitations in dealing with outlier data points or abrupt environmental changes. Such observations may imply the necessity for further refinement in their respective outlier rejection mechanisms or adaptation strategies to enhance robustness.

In contrast, algorithms like ORB-SLAM3 [2] and A-LOAM [5], despite not achieving the lowest error metrics, offer a balanced performance spectrum, making them viable candidates for a range of applications necessitating a trade-off between precision and computational demand.

Surprisingly, LeGO-LOAM [4], although proposed in 2018, still exhibited excellent odometry estimation precision, even exceeding some updated multi-sensor fusion-based algorithms. This fact inspires us that newer algorithms are not necessarily better. Therefore, in practical applications, we cannot blindly follow the state-of-the-art and need to choose the most suitable algorithm based on specific circumstances.

In summation, the comparative analysis accentuates the importance of algorithmic design choices in SLAM, emphasizing that an optimal balance between accuracy, computational efficiency, and robustness is pivotal for real-world applicability. Future advancements should aim at minimizing RMSE and Std while ensuring scalability and adaptability to diverse environmental dynamics.

2.3 Drawbacks and Solutions

Moreover, the sparse feature extraction of vision-based SLAM algorithms, while capable of providing accurate odometry, may result in maps lacking the necessary detail for intricate navigation tasks. This limitation underscores the necessity for refined mapping capabilities capable of capturing the complexities of diverse environments.

The expected superiority of learning-based methods over traditional techniques in SLAM has yet to materialize in practical scenarios. The adaptability of these models to a wide range of real-world environments remains a significant challenge, possibly due to the difficulty in generalizing from training datasets to unseen conditions.

Despite these challenges, the future of multi-sensor fusion SLAM shows promise, contingent upon the development of more effective fusion strategies. In this regard, the integration of deep learning and multi-sensor fusion is a potential strategy. For instance, novel neural networks like Transformer [12], renowned for their success in processing data from different modalities through multi-head attention mechanisms, could provide a new approach for enhancing multi-sensor fusion SLAM. Transformers may potentially address the critical issues of integrating disparate sensor data by focusing on the most relevant features from each sensor modality, thereby improving environmental modeling and movement tracking within the SLAM framework. However, such methods require the creation of diverse and representative training datasets, optimization of the computational efficiency of these models for real-time applications, and development of SLAM-specific architectures that incorporate the spatial and temporal dynamics unique to this domain.

Simultaneously, richer context awareness can be incorporated into the SLAM architecture, such as in DA-VO [13], which utilizes pre-trained flow prediction and semantic segmentation models as prerequisites for the algorithm, providing more robust visual localization in complex urban scenarios.

In summary, while current SLAM approaches encounter limitations, particularly in multi-sensor fusion, there exists a clear pathway to overcoming these challenges through innovative approaches such as Transformer models. By leveraging the unique strengths of various sensor modalities and addressing integration issues directly, the next generation of SLAM systems promises enhanced robustness, accuracy, and adaptability. This evolution is crucial for advancing autonomous navigation and robotic exploration, paving the way for a future where robots can seamlessly navigate and comprehend complex and changing environments.

References

[1] M. Grupp, "evo: Python package for the evaluation of odometry and slam." https://github.com/MichaelGrupp/evo, 2017.

[2] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[3] V. Usenko, N. Demmel, D. Schubert, J. Stückler, and D. Cremers, "Visual-inertial mapping with non-linear factor recovery," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 422–429, 2019.

[4] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.

[5] HKUST Aerial Robotics, "A-loam: Advanced implementation of loam (lidar odometry and mapping)," https://github.com/HKUST-Aerial-Robotics/A-LOAM, 2019, gitHub repository.

[6] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and systems*, vol. 2, no. 9. Berkeley, CA, 2014, pp. 1–9.

[7] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, "Kiss-icp: In defense of point-to-point icp–simple, accurate, and robust registration if done the right way," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1029–1036, 2023.

[8] J. Zhang and S. Singh, "Laser–visual–inertial odometry and mapping with high robustness and low drift," *Journal of field robotics*, vol. 35, no. 8, pp. 1242–1264, 2018.

[9] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

[10] H. Zhan, C. S. Weerasekera, J.-W. Bian, R. Garg, and I. Reid, "Df-vo: What should be learnt for visual odometry?" *arXiv preprint arXiv:2103.00933*, 2021.

[11] T. Qin, S. Cao, J. Pan, P. Li, and S. Shen, "Vins-fusion: An optimization-based multi-sensor state estimator," 2019.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[13] X.-Y. Kuo, C. Liu, K.-C. Lin, and C.-Y. Lee, "Dynamic attention-based visual odometry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 36–37.

[14] Z. Yuan, Q. Wang, K. Cheng, T. Hao, and X. Yang, “Sdv-loam: Semi-direct visual-lidar odometry and mapping,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

a’c

A Task1

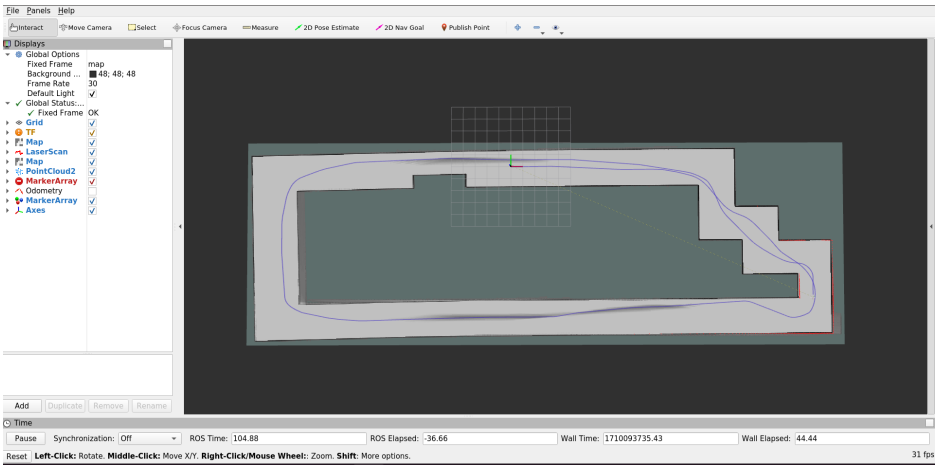

Figure 1: Cartographer running effect in Rviz.

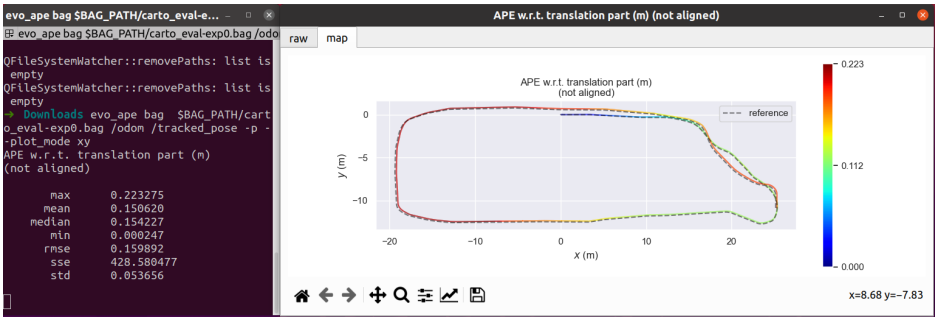

Figure 2: Cartographer APE resultes.

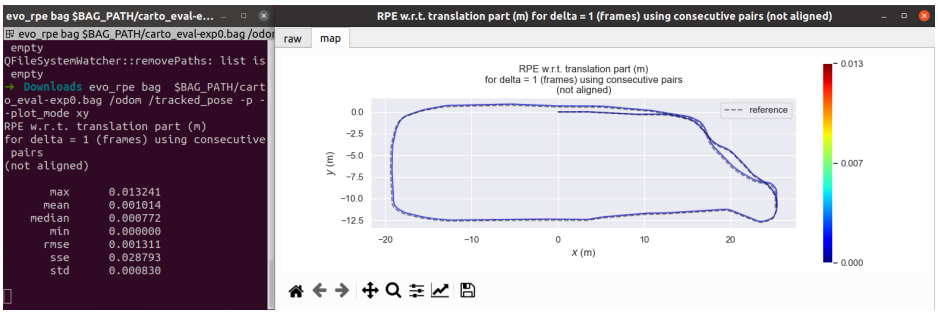

Figure 3: Cartographer RPE resultes.

Table 1: Cartographer parameter tuning experiments. NRD: *num_range_data*, OENN: *optimize_every_n_nodes*, MS: *constraint_builder.min_score*, LSW: *linear_search_window*, ASW: *angular_search_window*, TDCW: *translation_delta_cost_weight*, RDCW: *rotation_delta_cost_weight*

Index	APE Max	APE Mean	APE RMSE	NRD	OENN	MS	LSW	ASW	TDCW	RDCW
1	2.635	0.358	0.668	35	35	0.65	-	-	-	-
2	3.098	0.452	0.860	35	10	0.65	-	-	-	-
3	1.627	0.197	0.373	140	35	0.65	-	-	-	-
4	1.512	0.198	0.353	140	35	0.65	0.1	35	10	0.1
5	0.479	0.154	0.178	280	35	0.65	0.1	35	10	0.1
6	0.223	0.151	0.160	1000	35	0.65	0.1	35	10	0.1
7	0.224	0.151	0.160	2000	35	0.65	0.1	35	10	0.1
8	0.223	0.151	0.160	1000	5	0.65	0.1	35	10	0.1
9	3.997	0.467	0.963	35	5	0.65	0.1	35	10	0.1
10	3.990	0.488	0.998	35	1	0.65	0.1	35	10	0.1
11	3.302	0.937	1.139	10	5	0.65	0.1	35	10	0.1
12	2.314	0.429	0.721	70	5	0.65	0.1	35	10	0.1
13	1.077	0.175	0.257	140	5	0.65	0.1	35	10	0.1
14	4.751	1.331	1.968	140	5	0.1	0.1	35	10	0.1
15	1.405	0.192	0.301	140	5	1.2	0.1	35	10	0.1
16	4.725	1.314	1.956	140	5	0.8	0.1	35	10	0.1
17	4.756	1.278	1.911	140	5	0.65	0.5	35	10	0.1
18	4.756	1.304	1.941	140	5	0.65	0.2	35	10	0.1
19	4.750	1.337	1.972	140	5	0.65	0.025	35	10	0.1
20	4.753	1.319	1.957	140	5	0.65	0.1	70	10	0.1
21	4.748	1.339	1.976	140	5	0.65	0.1	15	10	0.1
22	1.079	0.175	0.258	140	5	0.65	0.1	35	20	0.1
23	1.077	0.174	0.255	140	5	0.65	0.1	35	1000	0.1
24	1.925	0.242	0.441	140	5	0.65	0.1	35	20	1.0
25	0.223	0.151	0.160	1000	35	0.65	0.1	35	20	0.1
26	0.223	0.151	0.160	1000	5	0.65	0.1	35	20	0.1
27	**0.223**	**0.151**	**0.160**	1000	5	0.5	0.1	35	20	0.1

B Task2

Table 2: APE of all employed algorithms.

Methods	Max	Mean	Median	Min	RMSE	Std
ORB-SLAM3 [2]	9.25	4.39	3.87	0.32	5.09	2.58
Basalt [3]	7.53	2.72	2.53	1.07	2.94	1.11
DF-VO [10]	10.90	4.50	3.79	0.13	5.31	2.83
Lego-LOAM [4]	6.61	3.20	3.25	1.15	3.28	0.73
A-LOAM [5]	11.84	3.28	2.51	0.61	3.83	1.98
KISS-ICP [7]	8.49	3.57	3.25	1.03	3.82	1.33
V-LOAM [8]	10.62	2.86	2.34	1.08	3.31	1.68
FAST-LIO2 [9]	24.21	12.61	12.91	0.77	13.19	3.88
SDV-LOAM [9]	36.51	10.39	9.50	0.60	11.96	5.93
VINS-GPS [11]	11.85	7.81	8.55	2.09	8.25	2.66

Table 3: RPE of all employed algorithms.

Methods	Max	Mean	Median	Min	RMSE	Std
ORB-SLAM3 [2]	1.66	0.030	0.030	0.00080	0.050	0.033
Basalt [3]	0.11	0.014	0.011	0.0006	0.018	0.011
DF-VO [10]	0.12	0.019	0.016	0.00050	0.012	0.012
Lego-LOAM [4]	3.34	1.04	1.04	0.0025	1.30	0.77
A-LOAM [5]	0.14	0.019	0.017	0.0010	0.022	0.011
KISS-ICP [7]	3.19	0.90	1.01	0.0015	1.10	0.63
V-LOAM [8]	0.11	0.014	0.012	0.001	0.018	0.010
FAST-LIO2 [9]	3.76	1.10	1.03	0.001	1.29	0.68
SDV-LOAM [14]	0.13	0.014	0.012	0.00049	0.017	0.010
VINS-GPS [11]	1.7	0.60	0.27	0.00083	0.81	0.54

B.1 Vision based SLAM

1. ORB-SLAM3

Table 4: ORB-SLAM3 details

Code Repertory: ORB-SLAM3
Running Command

`cd /home/simon/nus_ws/src/ORB_SLAM3/Examples_old/ROS/ORB_SLAM3`
`source ./build/devel/setup.zsh`
`rosrun ORB_SLAM3 Stereo /home/simon/nus_ws/src/ORB_SLAM3/Vocabulary/ORBvoc.txt /home/simon/nus_ws/src/ORB_SLAM3/Examples/Stereo/kitti.yaml ONLINE_RECTIFICATION`
`cd ~/LocalDiskExt/Datasets/HW2_SLAM/Task_2`
`rosbag play ./have_fun.bag /kitti/camera_color_left/image_raw:=/camera/left/image_raw /kitti/camera_color_right/image_raw:=/camera/right/image_raw`

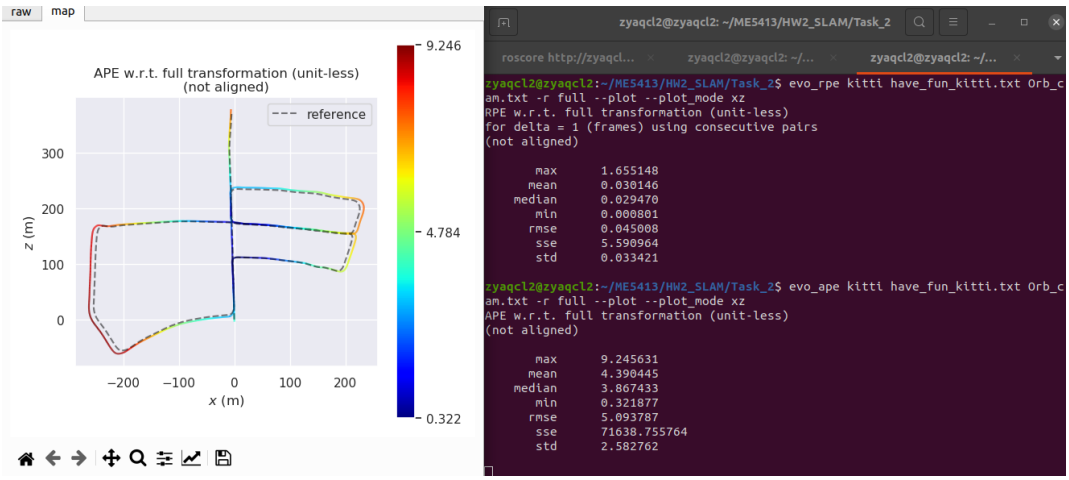

Figure 4: ORB-SLAM3-ape-rpe

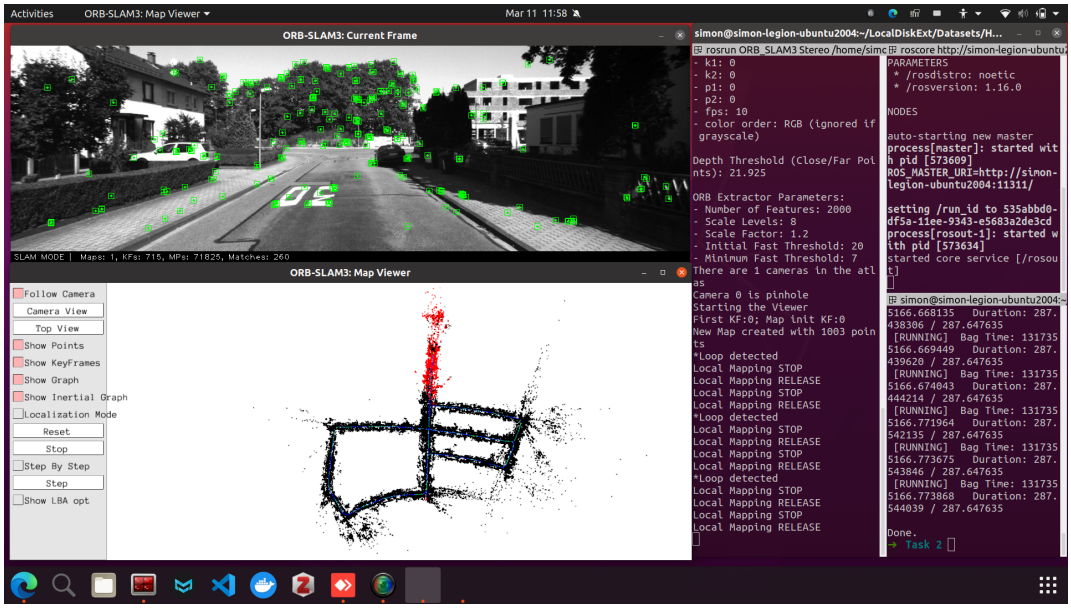

Figure 5: ORB-SLAM3 running effect

2. Basalt

Table 5: Basalt details

Code Repertory: Basalt
Running Command

`# Pre-process`
`python /home/simon/nus_ws/src/basalt/scripts/basalt_convert_kitti_calib.py`
`-d /home/simon/LocalDiskExt/Datasets/HW2_SLAM/Task2/KITTI/sequences/05`
`cd /home/simon/nus_ws/src/basalt/build`
`./basalt_vio --dataset-path /home/simon/LocalDiskExt/Datasets/HW2_SLAM/Task2/KITTI/sequences/05`
`--cam-calib /home/simon/LocalDiskExt/Datasets/HW2_SLAM/Task2/KITTI/sequences/05/basalt_calib.json`
`--dataset-type kitti --config-path /home/simon/nus_ws/src/basalt/data/kitti_config.json`
`--save-groundtruth 1 --show-gui 1 --use-imu 0 --marg-data ./mapping_cache --save-trajectory kitti`

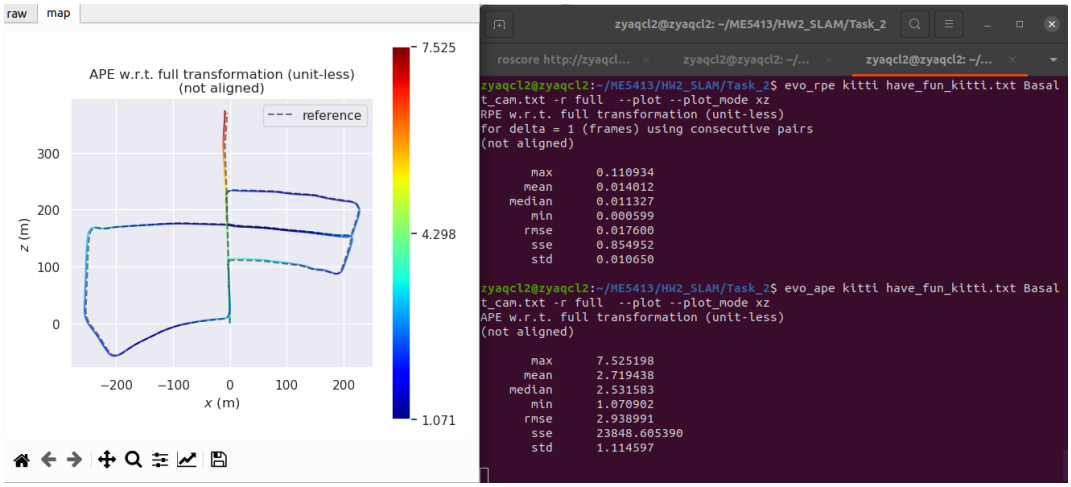

Figure 6: Basalt-ape-rpe

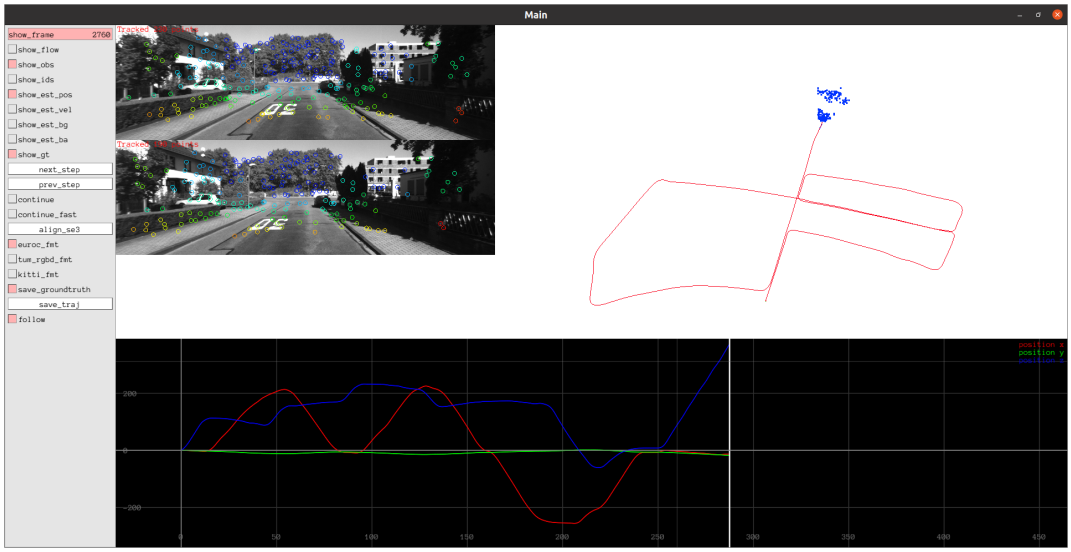

Figure 7: Basalt running effect

B.2 learning based SLAM

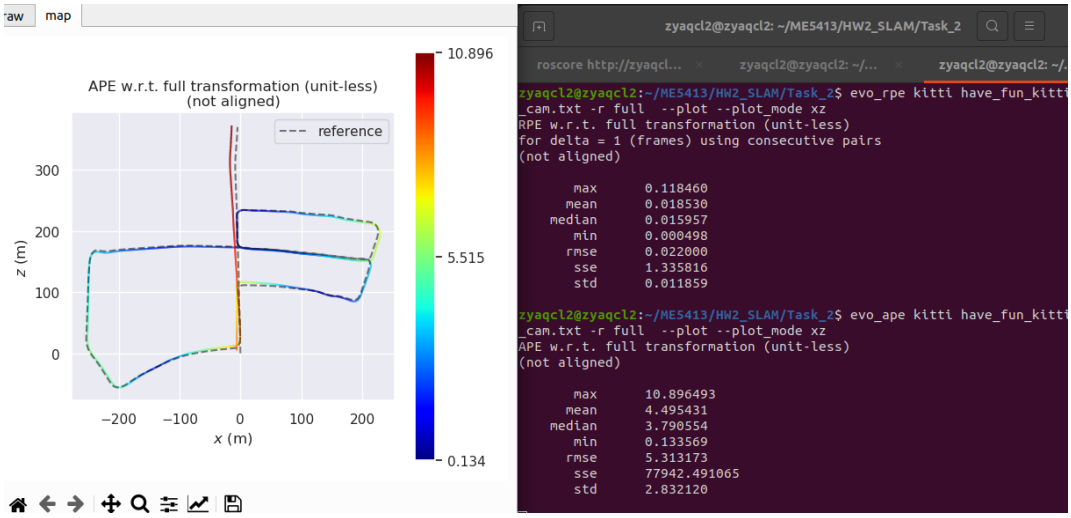

Figure 8: DF-VO-ape-rpe

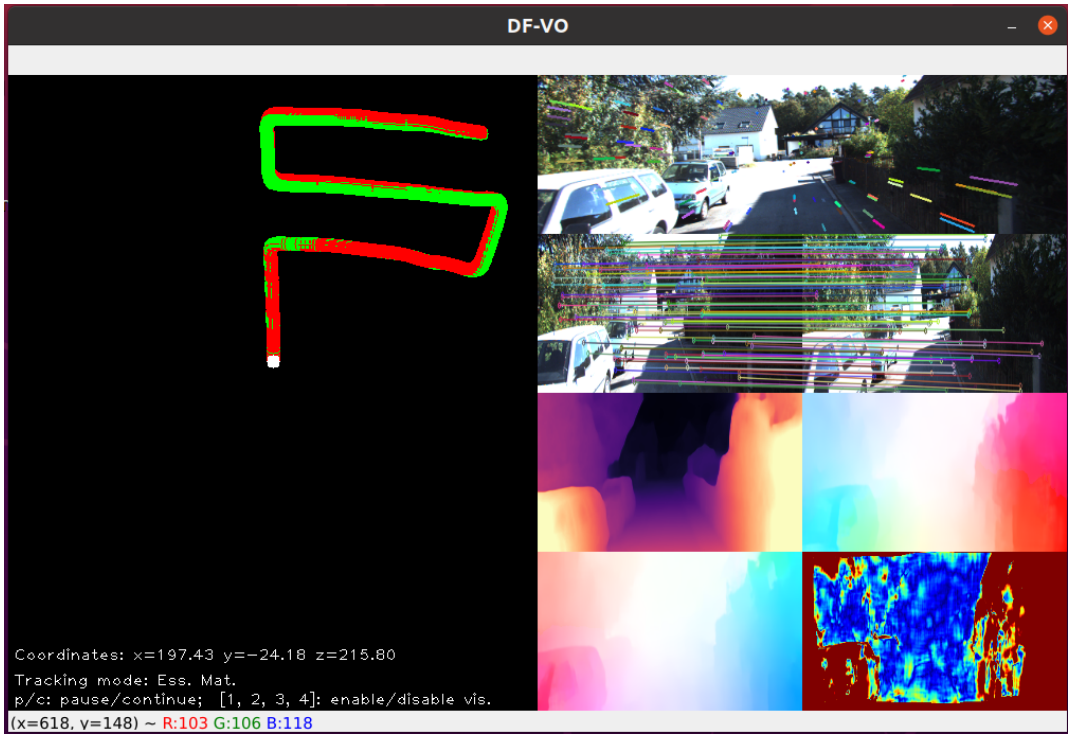

Figure 9: DF-VO running effect

B.3 Pure Lidar SLAM

1. Lego-LOAM

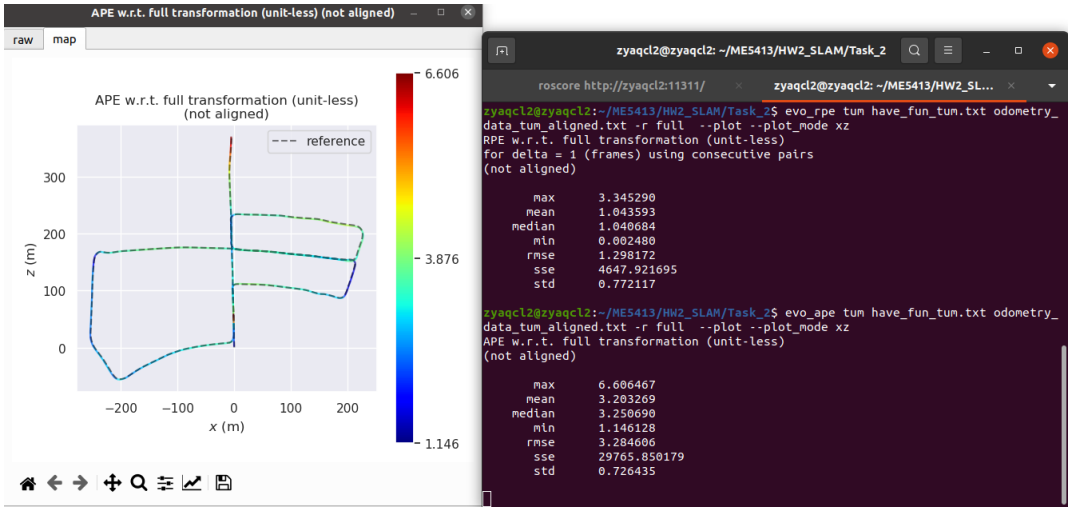

Figure 10: Lego-LOAM-ape-rpe

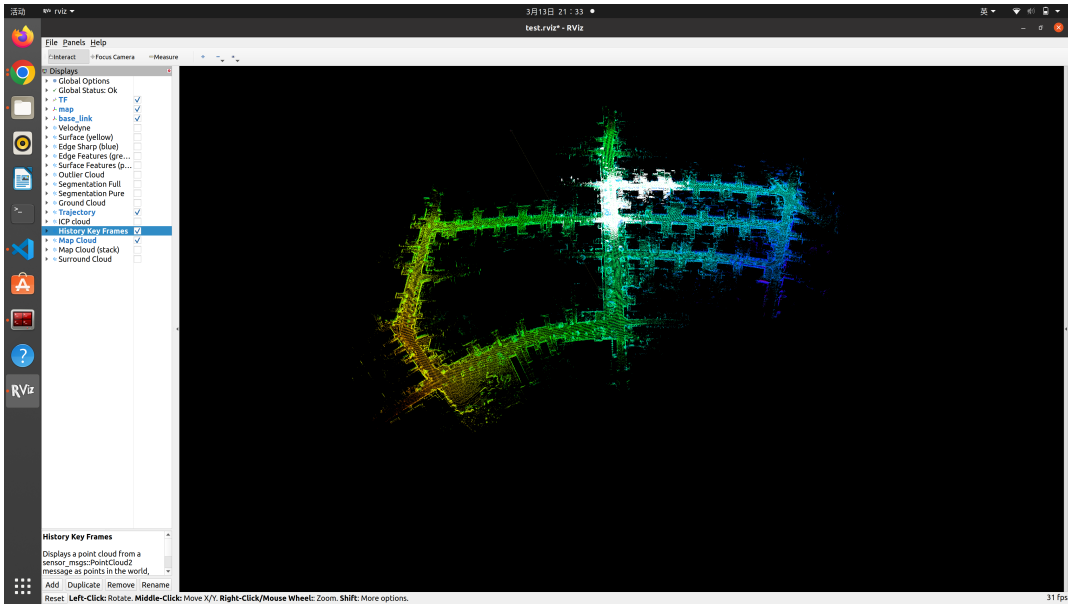

Figure 11: Lego-LOAM running effect

Table 6: Lego-LOAM details

Github Repo
`https://github.com/RobustFieldAutonomyLab/LeGO-LOAM.git`
Dependencies
`gtsam`
Run Command
`roslaunch lego_loam run.launch` `rosbag play have_fun.bag --clock`
Modifications
include/utility.h

`extern const string pointCloudTopic = "/kitti/velo/pointcloud";`
`extern const string imuTopic = "/kitti/oxts/imu";`
`extern const int N_SCAN = 64;`
`extern const int Horizon_SCAN = 1800;`
`extern const float ang_res_x = 0.2;`
`extern const float ang_res_y = 0.427;`
`extern const float ang_bottom = 24.9;`
`extern const int groundScanInd = 50;`
`extern const bool useCloudRing = false;`

2. A-LOAM

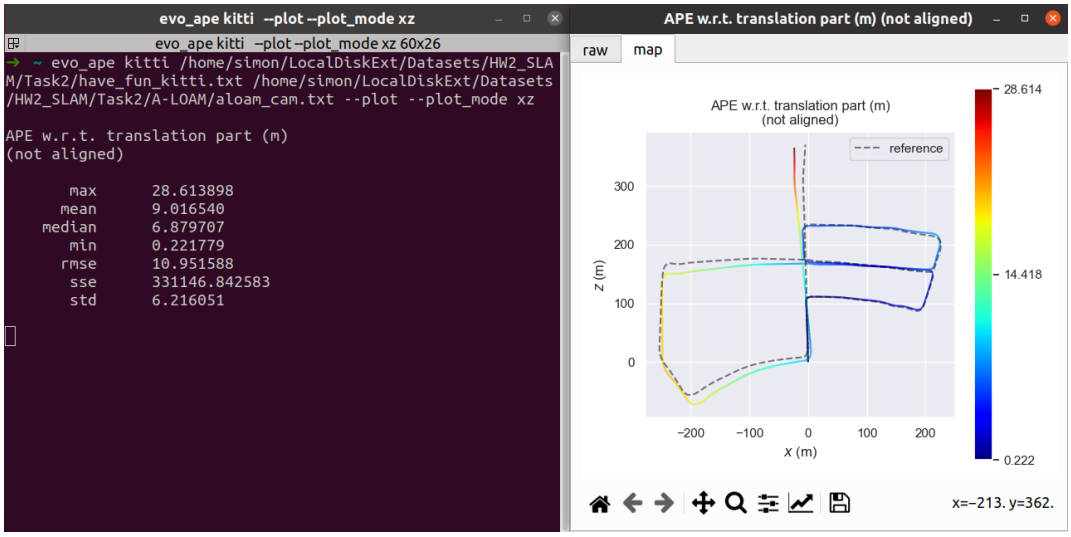

Figure 12: A-LOAM APE

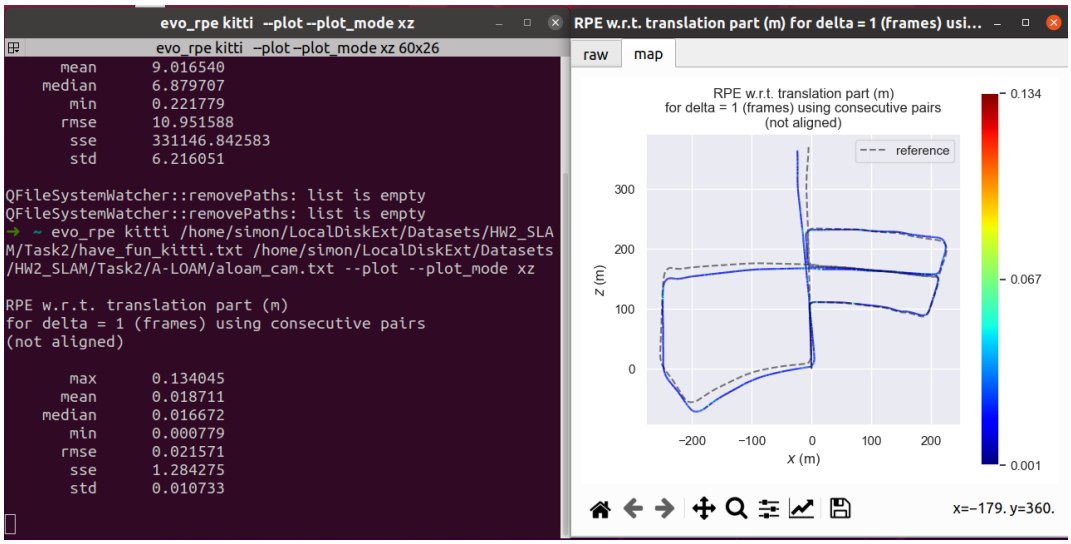

Figure 13: A-LOAM-rpe

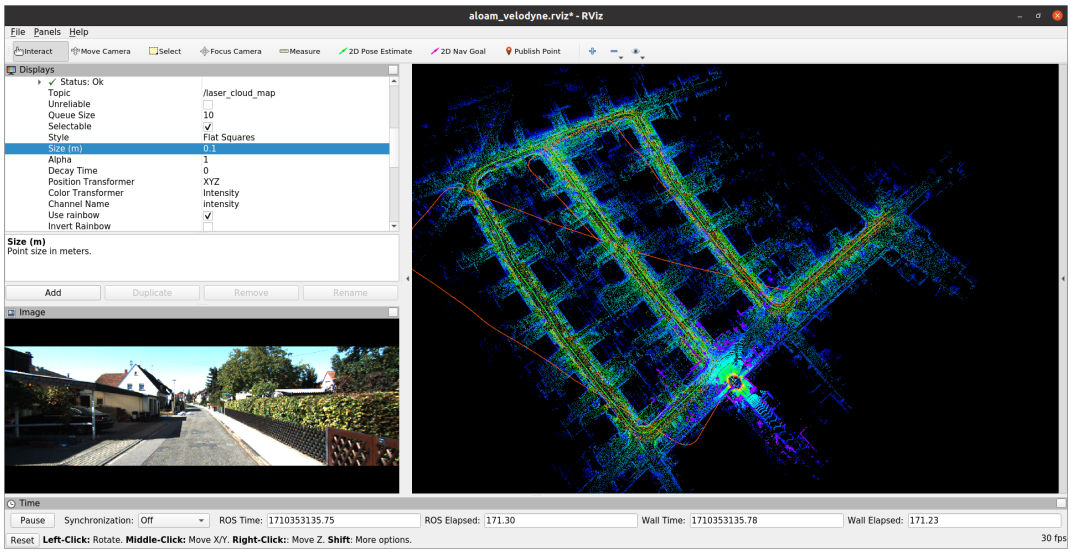

Figure 14: A-LOAM running effect

3. KISS-ICP

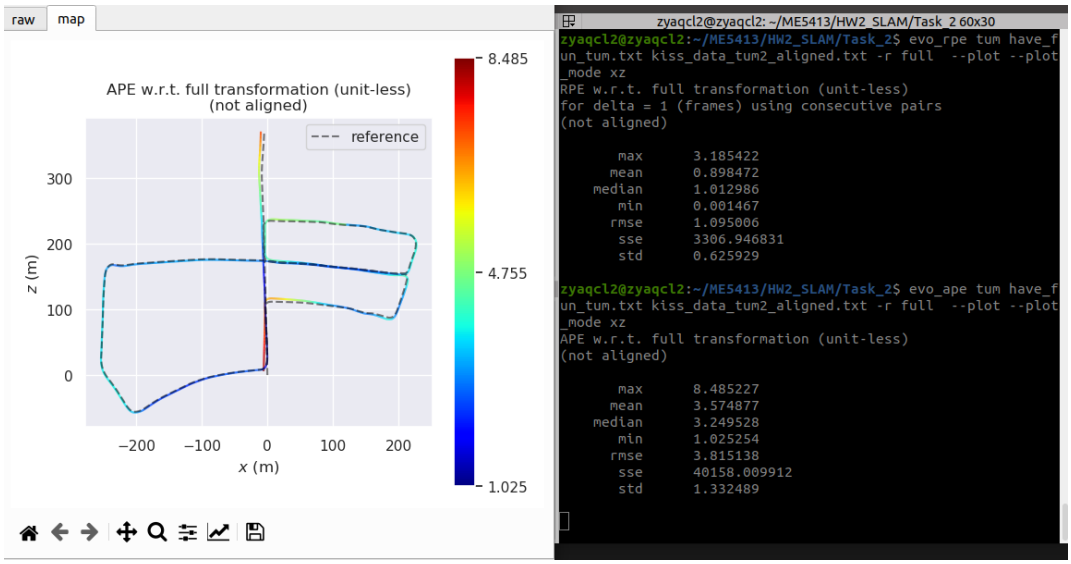

Figure 15: KISS-ICP-ape-rpe

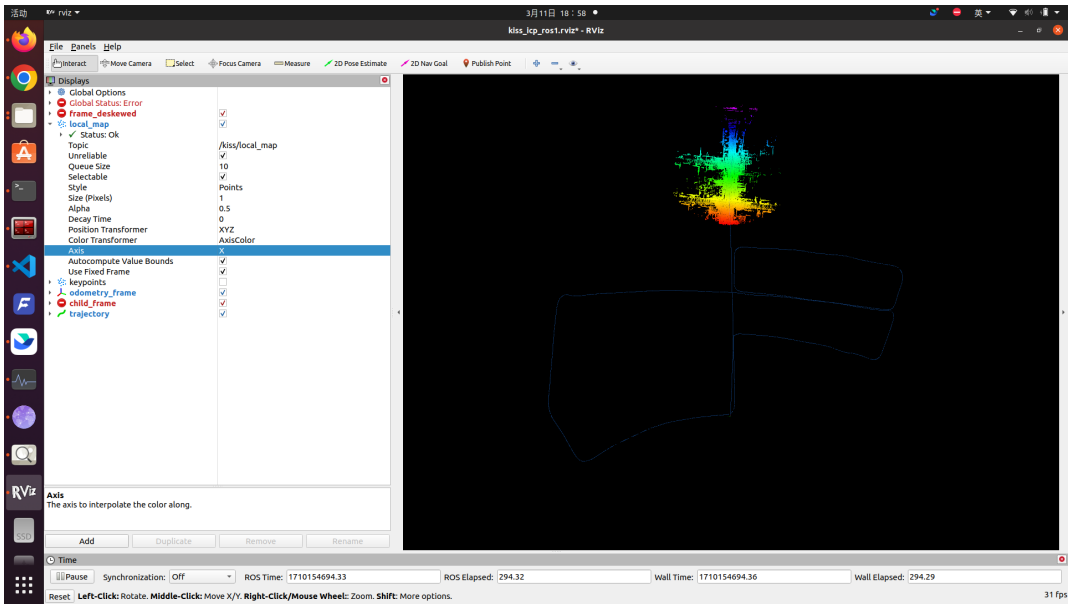

Figure 16: KISS-ICP running effect

Table 7: KISS-ICP details

Github Repo
`https://github.com/PRBonn/kiss-icp.git`
Dependencies
None
Run Command
`ros2 launch kiss_icp odometry.launch.py topic:=/kitti/velo/pointcloud` `rosbag play have_fun.bag`
Modifications
None

B.4 Multi-sensor fusion SLAM

1. V-LOAM

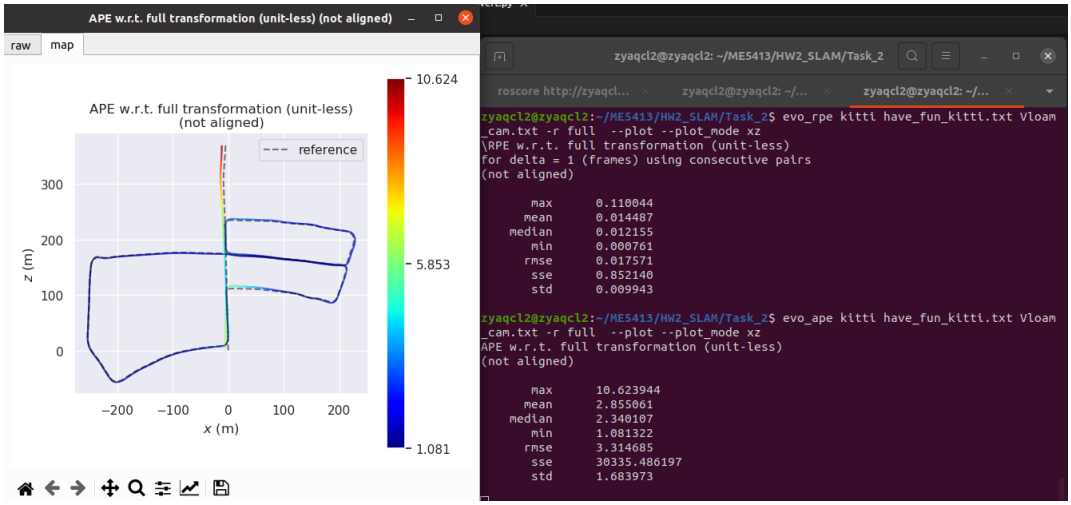

Figure 17: V-LOAM-ape-rpe

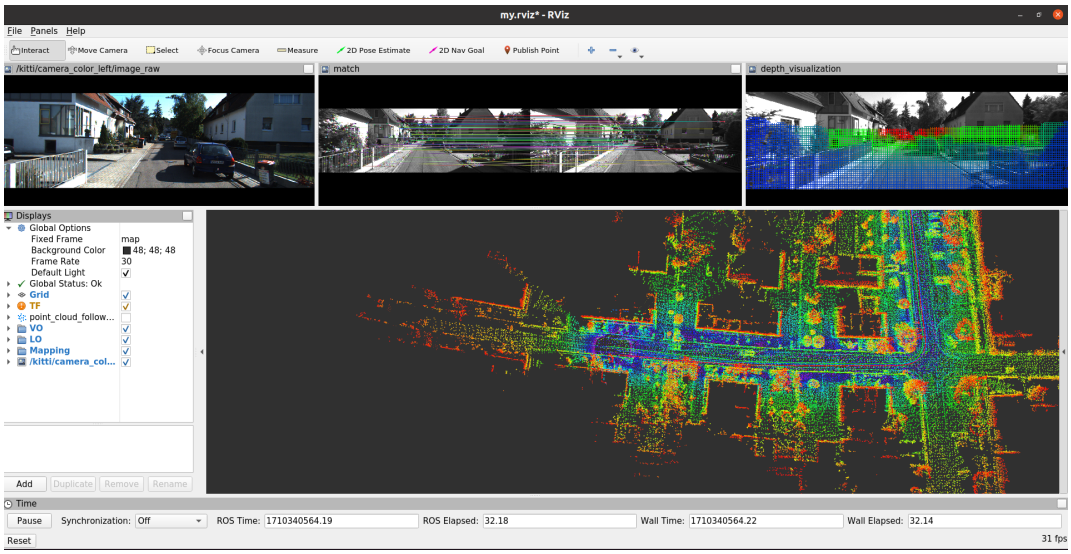

Figure 18: V-LOAM running effect

2. FAST-LIO2

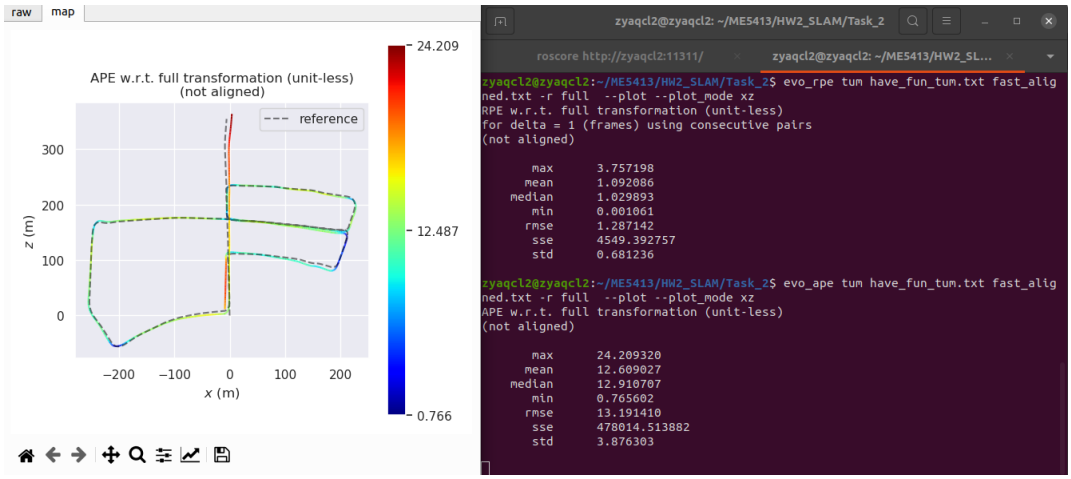

Figure 19: FAST-LIO2-ape-rpe

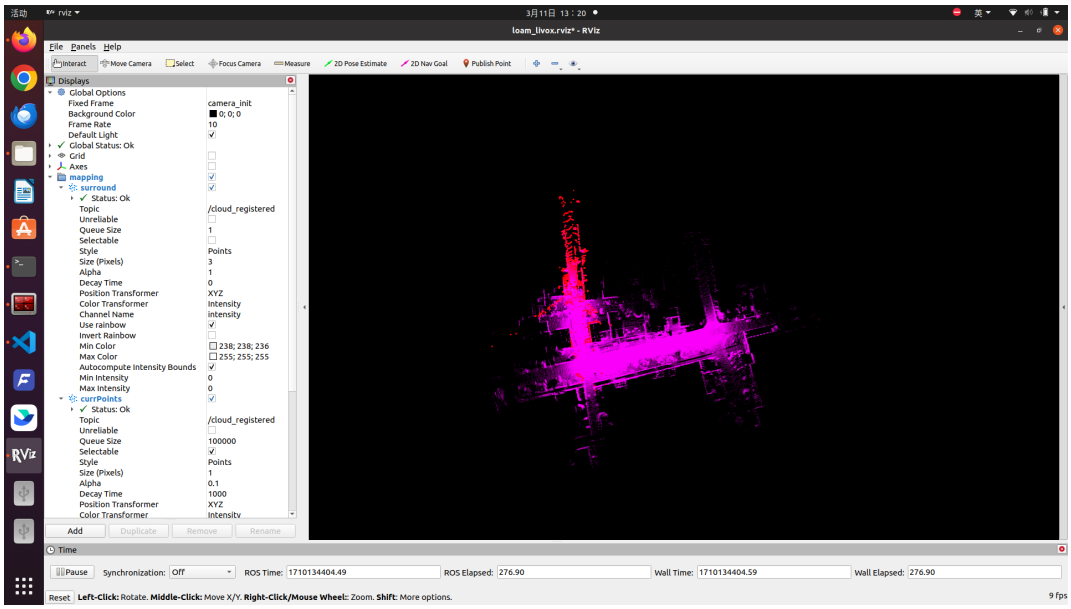

Figure 20: FAST-LIO2 running effect

Table 8: FAST-LIO 2 details

Github Repo
`https://github.com/hku-mars/FAST_LIO.git`
Dependencies
`PCL, Eigen, Livox driver`
Run Command
`roslaunch fast_lio mapping_velodyne.launch` `rosbag play have_fun.bag`
Modifications
config/velodyne.yaml

`lid_topic: "/kitti/velo/pointcloud"`
`imu_topic: "/kitti/oxts/imu"`
`scan_line: 64`
`extrinsic_T: [ 8.10543972e-01, -3.07054372e-01, 8.02723995e-01 ]`
`extrinsic_R: [9.99997685e-01, -7.85402788e-04, 2.02440581e-03,`
`7.55307105e-04, 9.99889850e-01, 1.48245438e-02,`
`-2.03582620e-03, -1.48229758e-02, 9.99888022e-01]`

3. SDV-LOAM

Table 9: SDV-LOAM details

Code Repertory: SDV-LOAM
Running Command

`sudo docker run -it -d -p 5901:5901 -p 6080:6080 -p 4040:4040 -v /home/simon/LocalDiskExt/Datasets/HW2_SLAM/Task2:/dataset --privileged --name kinetic 18a7e522d0da`
`sudo docker container start kinetic`
`sudo docker exec -it kinetic /bin/bash`
`su ubuntu`
`cd catkin_ws && source ./devel/setup.zsh`
`roslaunch sdv_loam run.launch`

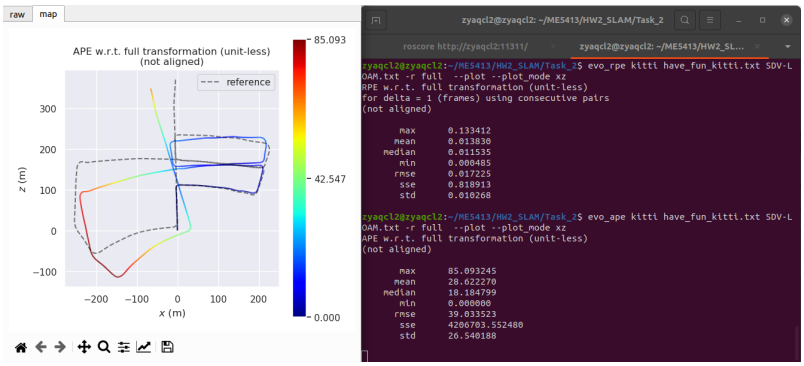

Figure 21: SDV-LOAM-ape-rpe

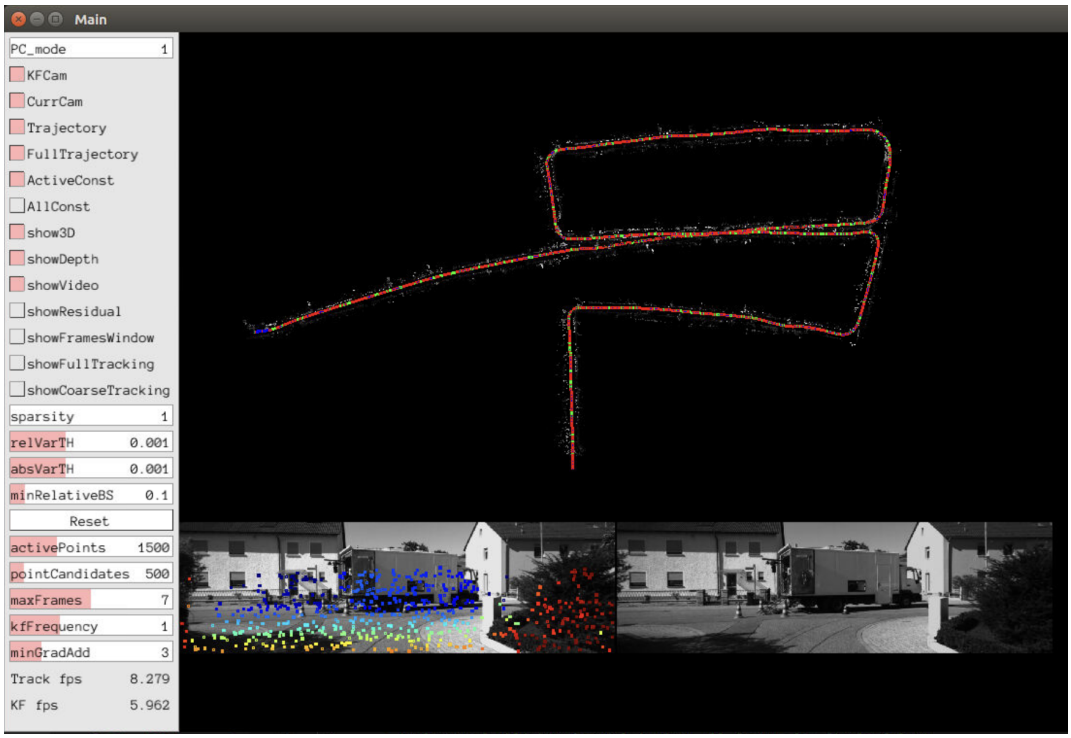

Figure 22: SDV-LOAM running effect

4. VINS-Fusion

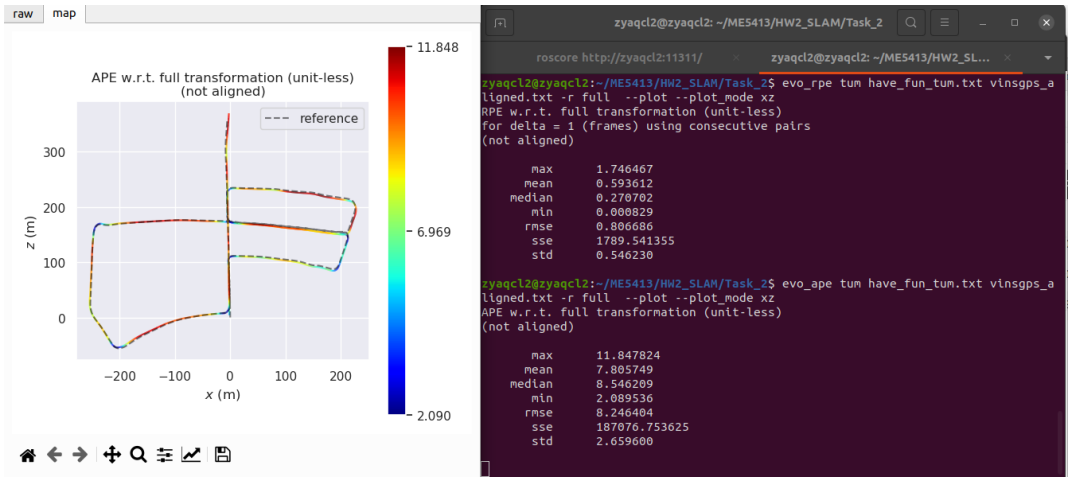

Figure 23: VINS-Fusion-ape-rpe

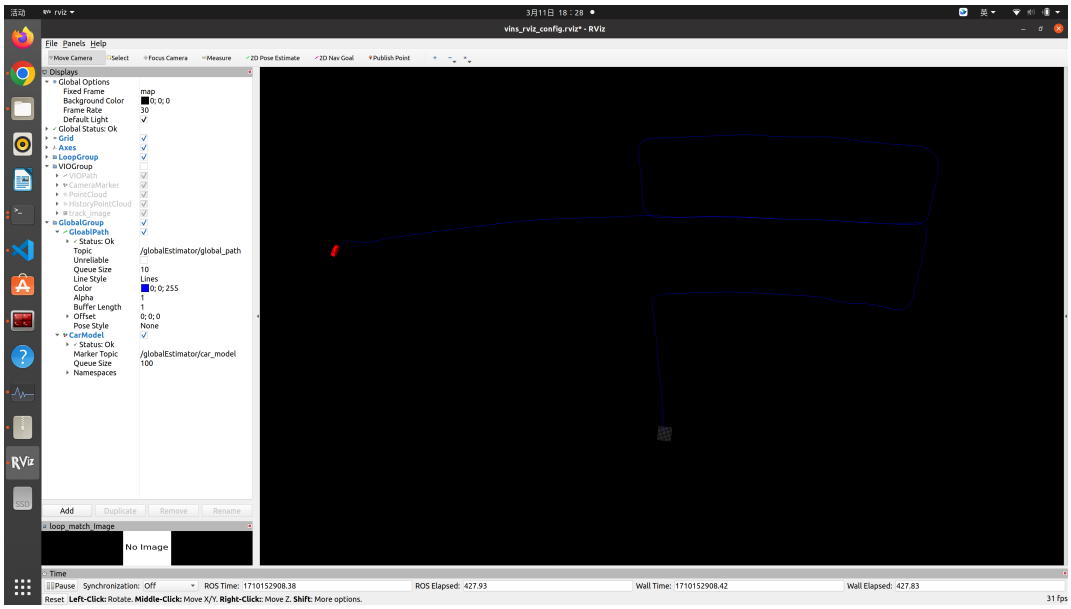

Figure 24: VINS-GPS running effect

Table 10: VINS-GPS details

Github Repo
`https://github.com/guisoares9/VINS-Fusion.git`
Dependencies
`Ceres`
Run Command

`roslaunch vins vins_rviz.launch`
`rosrun vins vins_node ~/catkin_ws/src/VINS-Fusion/config/kitti_odom/kitti_09_30_config.yaml`
`rosrun global_fusion global_fusion_node`
`rosbag play have_fun.bag`

Modifications
config/kitti_raw/kitti_09_30_config.yaml
`image0_topic: "/kitti/camera_gray_left/image_raw"` `image1_topic: "/kitti/camera_gray_right/image_raw"`
global_fusion/src/globalOpt.cpp
`pose_stamped.header.frame_id = "map";`
global_fusion/src/globalOptNode.cpp

`car_mesh.header.frame_id = "map";`
`odometry.header.frame_id = "map";`
`odometry.child_frame_id = "map";`