Comparative Analysis of Visual and Lidar Based SLAM Algorithms in ROS

Ashwin Nehete*, Harsh Dhruva†, Xikai Dai‡ and Vimalesh Vasu§
Mechanical Engineering Department, Carnegie Mellon University
Pittsburgh, PA, USA
Email: *anehete@andrew.cmu.edu, †hdhruva@andrew.cmu.edu, ‡xikai@andrew.cmu.edu, §vvasu@andrew.cmu.edu

***Abstract*—We present a report on implementation of different visual and lidar based SLAM algorithms in Robotic Operating System (ROS). SLAM is considered to be a complex problem because to localize itself a robot needs a consistent map and for acquiring the map the robot requires a good estimate of its location. This mutual dependency among the pose and the map estimates makes the SLAM problem hard and requires searching for a solution in a high-dimensional space.**

***Index Terms*—ROS, RGB-D, Lidar, simultaneous localization and mapping (SLAM), CARLA simulator**

I. Introduction

Simultaneous Localization and Mapping (SLAM) is a robotics problem that aims to give robotic systems the ability to produce maps of their immediate environments and be able to localize their own positions at the same time. It is a complex problem that involves a combination of large data processing and algorithmic tuning methodologies to produce useful outputs. Given the complexity of the problem, there is a wide variety of solution implementations that each harbor their own advantages and disadvantages. For example, some implementations may only perform well in indoor environments, and others may only perform well in outdoor environments. This project's aim is to explore different lidar and visual based SLAM algorithms in ROS. An in-depth investigation and comparative analysis will provide a great insight into the workings of different algorithms, and create some groundwork for our team's research in the future.

To produce reliable comparative results, the chosen SLAM methods are run using the same data-sets as a control. In addition, each set of implementations is also compared using the same data-sets of varying environment and robot conditions, which allows for a more comprehensive study on the advantages and disadvantages of each implementation in relation to the other implementations.

II. Data Collection

A. Benchmark Datasets

In the community, many research teams have already done extensive work on developing reliable data-sets to be used for testing SLAM implementations. These works have been refined and optimized to produce the best results, and made public so as to mitigate the tedious process of collecting reliable data for independent and organization researchers. Before collecting our own data, we utilized these benchmark data-sets for our chosen SLAM implementation.

The purpose of running the benchmark data-sets is to get an initial idea of the algorithms using the best performing data sources available. We use these data-sets to understand key algorithmic aspects like the sparsity, loop-closure, and robustness of each algorithm, so that we have control data to compare with when we collect our own data and do our own analyses.

The benchmark data-set we used is the TUM RGB-D data-set developed by the Computer Vision Group of the Technical University of Munich [8]. The TUM data-set, recorded from a Microsoft Kinect, provides RGB-D and ground truth data for use in visual SLAM implementations [8].

B. CARLA Autonomous driving simulator

An open urban driving simulator, *CARLA* is an open source simulator for autonomous driving research. Democratization of autonomous driving is one of the main goals of CARLA, which is easily accessible and customizable by the users. It simulates a dynamic world implemented over Unreal Engine 4 (UE4) providing a simple interface between the agent interacting with the world [3]. The flexibility in configuration of several Autonomous driving sensor suite in CARLA along with the integration with other learning environments as ROS attracted our attention towards this simulator. Various types of sensors ranging from camera to radars, lidar, etc. Moreover, *ROS-Bridge* establishes a connection between ROS and CARLA. The messages exchanged between the ROS nodes can be translated to commands to instruct the CARLA agent. Similarly, it provides the sensor data from CARLA to be translated into ROS topics which are then fed into our SLAM system implementations.

We tested data in several different scenarios, four of which we will present as results. These include four drives in different Town and environment settings. The Scenarios are described as follows:

- Bag1: Long drive with no loops
- Bag2: Loop three times around a block
- Bag3: Loop around block in a dynamic environment with cars and pedestrians
- Bag4: Quarter Circle with constant turning

Our simulated car was attached with a front RGB Camera, a front Depth Camera, an IMU and a Lidar.

C. Metrics

In order to choose a SLAM algorithm for a given problem statement, it is necessary to compare the algorithms based on certain metrics. We obtain estimated camera trajectory and the estimated map as an output from any SLAM system. Although it is noticeable that a good trajectory does not ensure a good map. It is in principle possible to evaluate the quality of resulting estimated map, but obtaining accurate ground truth maps is tough. Hence, we analyse the estimated trajectory obtained from the algorithm considering the robot poses during data acquisition.

For evaluation, we consider the ground truth poses as $Q_1, ..., Q_n \in SE$ and sequence of poses from the estimated trajectory as $P_1, ..., P_n \in SE$. We use the absolute pose error and relative pose error for comparison [9]. Absolute pose error accounts for the global consistency of the estimated trajectory. It is evaluated by comparing the absolute distances between the ground truth and estimated trajectories. On the contrary, Relative pose error measures the local accuracy of the estimated trajectory. It accounts for the drift of the robot's estimated trajectory.

III. Algorithms

A. ORB-SLAM2

ORB-SLAM2 is a complete SLAM system based on monocular, stereo, and RGB-D sensors which focuses on building globally consistent maps in a wide range of environments. The slam system has three major parallel threads:

- Tracking for camera localization by recognition of ORB features with every frame followed by minimizing reprojection error.
- Local mapping and optimization by performing local bundle-adjustment.
- Loop closing to detect loops and perform pose-graph optimization to correct for the accumulated drift.

ORB are binary features invariant to rotation and scale (in a certain range), resulting in a very fast recognizer with good invariance to viewpoint. The system also maintains a covisibility graph that links two keyframes based on observed common points. It also uses DBoW2 for relocalization in cases where tracking fails and there is a need to reinitialize the same map. ORB-SLAM2 is able to achieve zero-drift localization in areas that have already been mapped. Also, this slam system depicts that bundle adjustment performs better than direct methods or ICP in cases where accurate camera localization is essential [1].

B. RTAB-MAP

RTAB-Map with RGB-D uses visual and depth images for feature extraction and matching for incremental visual based loop detection. Compared to other SLAM algorithms, RTAB-Map uses LTM (Long-Term Memory) and WM (Working Memory) to separately store nodes that contain important information such as odometry poses. Nodes will be classified based on their weights. If the process time for loop closure exceeds a certain threshold, the location with the lowest weight will be transferred into the LTM to free up the memory for the process. The weights of the feature locations is determined by how often it is visited, Nodes with lower weights will be stored into the Long-Term memory section. At the same time, WM is used to determine the similarity between the scanned features, which further determine the weight of each node. At the same time, the WM is also used for close loop detection. To determine loop closure, RTAB-Map uses Bayesian filters to determine the possibility of having a close loop, and compare the newly scanned nodes with the nodes stored in the WM. If the comparison result returns a high chance of loop closure, the new and old nodes are connected. If the neighbor nodes are stored in the LTM, it will be retrieved and stored in the WM for the process.

LTM, WM and STM(Short Term Memory) combined allows RTAB-Map to maintain a certain number of nodes during the process, and further limits the memory usage for the computer or robot. This advantage ideally allows RTAB-Map to be very useful for large scale reconstruction or long-term scanning.

C. Google Cartographer

The Cartographer system performs real time SLAM in both 2D and 3D environments, across multiple sensor suites. Cartographer consists of two interrelated subsystems, namely local SLAM and global SLAM. The local SLAM successively builds sub-maps that are locally consistent, however could drift over time. The global SLAM runs background threads to detect loop closure constraints by matching scans against created sub-maps. The global SLAM does the job of tying the sub-maps together, incorporating other sensor data to provide the most consistent global solution [10].

Input Range finding sensor measurements are filtered through a band pass filter to restrict the range between a minimum and maximum value. A fixed-size voxel filter is applied to subsample input point clouds, to reduce computational weight of point handling and to address issues with filtering points in sparser areas. An adaptive voxel filter is then applied to determine the optimal voxel size and achieve the target number of points. The use of an Inertial Measurement Unit (IMU) is optional in 2D SLAM, but needed to perform 3D SLAM, used to get an initial guess for the scan orientation to reduce difficulty of scan matching [10].

1) Local SLAM: Local SLAM inserts a new scan into the current sub-map using scan matching, which can be done through two different strategies, CeresScanMatcher and RealTimeCorrelativeScanMatcher. Submaps must be small to minimize drift and be locally correct, and large enough for potential loop closure to work. Submaps are widely stored in probability grids, one per sub-map in 2D, and two hybrid probability grids in 3D.

2) Global SLAM: Global SLAM is a pose graph optimization framework, optimizing based on the build constraints between nodes and submaps. Global optimization has cost functions to take into account multiple sources of data including the global and non-global constraints, IMU information,

local SLAM poses and fixed frame odometry. A final global optimization with more iterations is run when the trajectory is finished.

D. Hardware and Software

Since this project's aim is to test different SLAM algorithms that could be useful for group members future research, the hardware platform has no specific requirement. Furthermore, some of the SLAM algorithms are designed to be efficient in processing on mobile platforms, so there is no need to regulate the hardware platform for the testing. For the software side, all testing for the algorithms were conducted in a ROS environment with a Linux system, since this is most likely the setup for the future SLAM related research.

ROS version: Melodic

Linux version: 18.04.5 LTS (Bionic Beaver)

The only system that requires heavy computing power is the CARLA simulation, and the hardware used for the simulation is described in Table I below.

TABLE I
HARDWARE PLATFORM FOR SIMULATION

CPU	Intel(R) Core(TM) i9-10980HK CPU @ 2.40GHz
RAM	32 GB
GPU	Nvidia RTX-2070 Super
ROS version	Melodic
Linux version	18.04.5 LTS (Bionic Beaver)

IV. Initial Testing Results

A. ORB-SLAM2

As a part of our initial testing to verify our implementation, we tested the TUM sequences on ORB-SLAM2. The following sequences were used to depict different situations and test the robustness of the slam system.

The freiburg2_pioneer_360 sequence was recorded by a Kinect mounted on top of ActiveMedia Pioneer 3 Robot. It tests the applicability of SLAM systems to wheeled robots. Several objects like office containers, boxes are placed inside a hall with large dimensions. The freiburg3_long_office_household sequence is recorded with a RGB-D sensor moved through a office scene with much texture and structure. The end of the trajectory overlaps with the beginning to allow a large loop closure. For the freiburg3_teddy Asus Xtion sensor was moved around a teddy bear in two rounds at different heights. The values of APE (Absolute Pose Error) and RPE (Relative Pose Error) are mentioned in Table II.

TABLE II
ABSOLUTE POSE ERROR AND RELATIVE POSE ERROR

Dataset	APE	RPE
Motor 2	0.090	0.393
long_office_household	0.022	0.470
freiburg_teddy	0.029	0.509

B. RTAB-MAP

In order to have a rough idea on the performance of the RGB-D based RTAB-MAP, a demo from RTAB-Map was performed based on the famous TUM freiburg dataset. As Figure 1 suggests, the Rtabmapviz plugin allows the visualization of both the mapping reconstruction and the camera trajectory. And by exporting the data of mapping results, the accuracy of the trajectory can be calculated. By observing the reconstruction process through Rtabmapviz, it appears RTAB-Map is very suitable for handheld mapping and is a relatively fast mapping process.

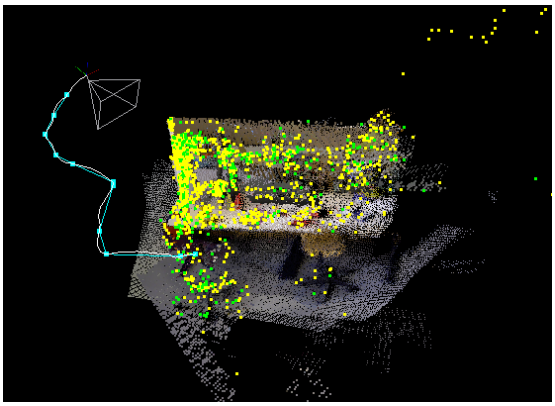

Fig. 1. Demo run

The next step is to import the CARLA-RGBD data. For this testing, the RTAB-Map will only feed in RGB-D images as input data, and the odometry information will be retrieved through the visual odometry process.

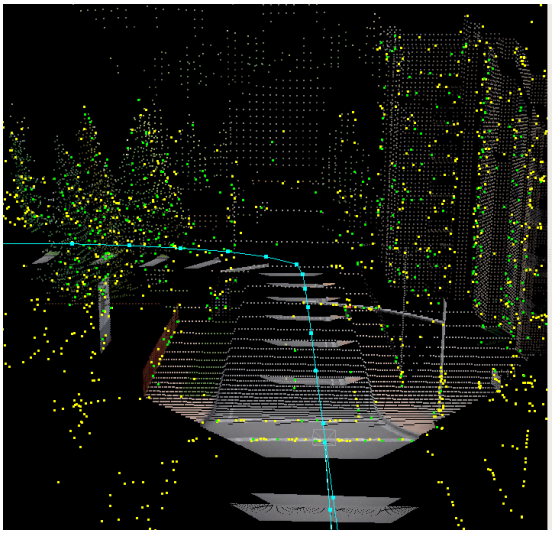

Fig. 2. Point cloud projection with grey area as road surfaces

As Figure 2 suggests, The RTAB-Map's performance on reconstruction, especially the road surfaces, is heavily affected by the simulated car speed. Generally, due to the fixed frame rate of the camera, as the car speed increases, the differences between the consecutive frames will increase. This increase

of differences will cause RTAB-Map to fail in predicting the same feature locations in different frames, and further cause problems such as loss in tracking.

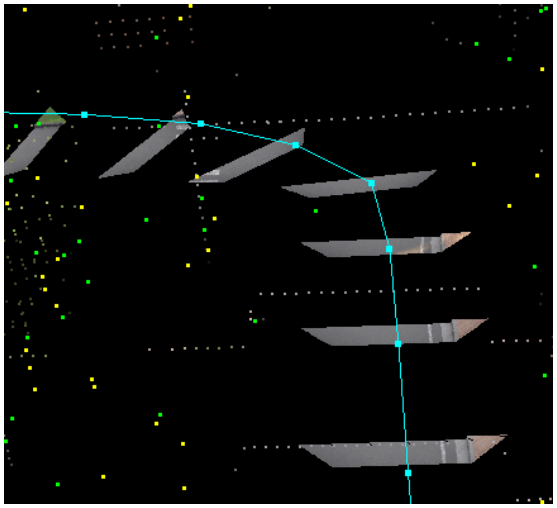

Fig. 3. Color points of road surfaces reconstruct

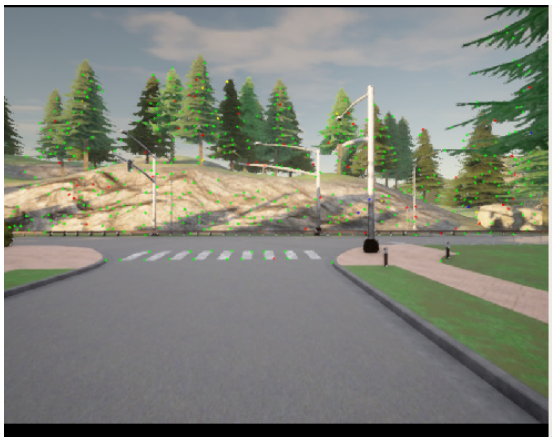

Fig. 4. Camera pov view

Another result from the testing is the mapping model. As Figure 3 suggests, the mapping reconstruction seems to model only the road surfaces and contains very little of the surroundings. The potential explanation of that could be explained based on Figure 4 which is the vehicle's RGB camera's angle view. As the figures indicate, most portions of the images are focused on the road surfaces. With the addition of an open driving environment and comparatively high speed, it became relatively difficult for RTAB-Map to find the feature locations and use them for map reconstruction purposes.

Also based on left portion of Figure 5, it failed in the loop detection, on two very similar pictures, and this small difference appears to be the reason why RTAB-Map rejected the loop hypothesis, which proves it is very sensitive to the small differences on images.

The CARLA simulation result, being purely based on the RGB-D images, didn't match its initial expectation, which

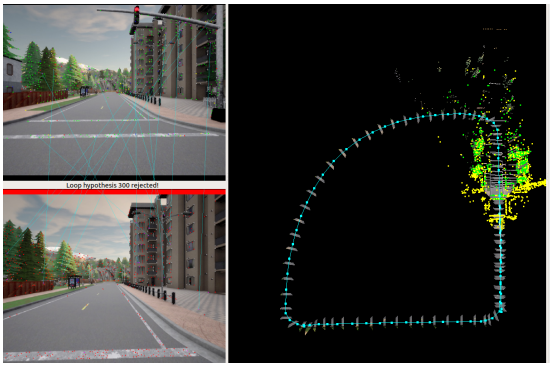

Fig. 5. Loop rejection in Bag4 Data

raised a question that extends beyond the original project plan. What is the real performance for RTAB-Map? To find that out, another experiment based on an Intel Realsense L515 camera is conducted. This time the input will be IMU reading and RGB-D data. This is a fairly simple experiment, with the specific instruction available on line [7], but the result is very different than the initial demo testing. RTAB-Map is still very sensitive to its environment. It responds very aggressively toward sudden movement of the camera or slow moving objects in front of the camera, and causes RTAB-Map to lose track.

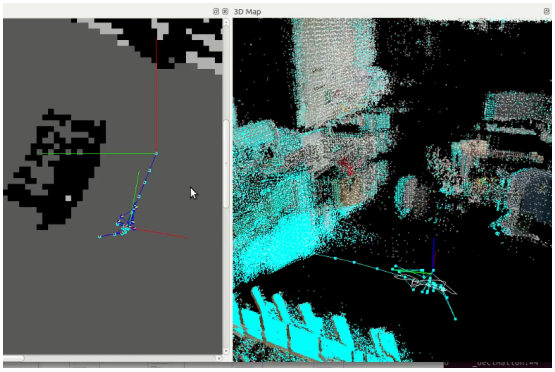

Fig. 6. RTAB + L515 Real-time Reconstruction

As Figure 6 indicates, compared to the demo run, the RTAB-Map is not performing nearly as well. One possibility for that is the potential hyper-parameter tuning. All the parameters, besides those used to subscribe published ROS topics, are set to default values in the launch files during the testing. So ideally, tuning hyper parameters could improve RTAB-Map's performances. Besides that, the RTAB-Map still shows its strength in the tracking as the loop in Figure 5 indicates. It is fairly impressive for a visual odometry to achieve such high accuracy.

C. Cartographer

Initial Testing on the Cartographer was performed on the Deutsches Museum demo bag files for 2D and 3D SLAM, and data from the CARLA Simulator. The output occupancy

map and factor graph for the Bag4 data is shown in Figure 7. The testing process for cartographer is relatively straight forward, and the testing result based on different input data is represented in the Appendix section. Based on the testing result, Cartographer is relatively good at detecting loop closure, but also shares the similar drawback with other SLAM algorithms: it is very sensitive to the dynamic environment in front of the scan.

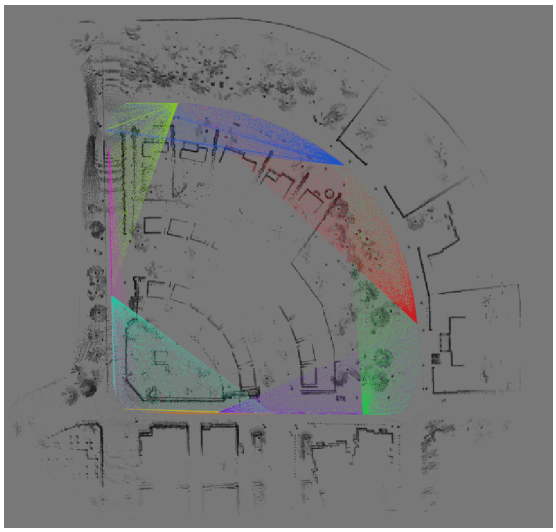

Fig. 7. Occupancy map and Factor Graph for Bag4 Data

V. Results on CARLA Bag Files

We present the calculation of Absolute Pose Error (APE) for ROSbag files collected by running different scenarios in CARLA. For specific plot, please refer to the appendix.

A. ORB-SLAM2 Monocular

Figure 8 depicts the APE for Monocular ORB-SLAM2. Since, monocular SLAM lacks scale, we observe that the estimated trajectory is scaled down as compared to the ground truth trajectory. The APE values for Monocular SLAM are observed to be the highest since it is most likely to experience drift. Figure 9 shows the sparsely reconstructed point cloud of the environment. Monocular ORB-SLAM2 is able to produce denser maps that RGBD based ORB-SLAM2. Loop closure is performed in Bag2 Data and ORB-SLAM2 is able to correct the map based on it. However as seen in 18, ORB-SLAM2 loses tracking in the dynamic environment due to the addition of cars and pedestrians. Some challenges observed through test runs include tracking failures at high speeds and in dynamic environments, building maps based on further static landmarks, and false re localization in repetitive environments.

B. ORB-SLAM2 RGBD

Figure 10 depicts the APE and estimated trajectory for RGBD ORB-SLAM2. RGBD based ORB-SLAM2 performs similar to Monoculor in Bag1, but also provides a scaled trajectory. The estimated trajectory is at a false start and drifts over time. For Bag2, the algorithm loses track of the map during a sharper turn, however Monocular ORB-SLAM2 does not lose track. This is likely due to a lower number of feature points detected using RGBD based ORB-SLAM2. In the Bag3 data, the algorithm also loses tracking due to high traffic during the turn. For Bag4 data the generated trajectory is mismatched with scale, but the algorithm does not lose track during the sharp turns and the constant turning radial curve as seen in Figure 23.

C. RTAB-MAP RGBD

Figure 11 depicts the APE and estimated trajectory for RGBD RTAB-MAP. The APE values for RTAB-MAP are the lowest. This strengthens the assumption that this SLAM system performs well on an outdoor dataset. For Bag1 data the RTAB-MAP provides the most accuracy trajectory, due to usage of odometry data. Loop closure is well achieved in Bag2, without losing track of the trajectory during higher speed maneuvers as seen in Figure VII-A. For the Bag3 data in the dynamic environments, RTAB-MAP also loses tracking in the higher traffic areas as seen in Figure 20. For Bag4 data RTAB-MAP performs extremely well in estimating the trajectory, with the least Absolute Pose Error as seen in Figure 24.

D. Cartographer

Figure 12 depicts the APE and estimated trajectory for Cartographer. The APE values for cartographer lie midway to those of the other SLAM systems. For Bag1 data, there is no loop closure taking place as seen in Figure 13. We speculate this to be a reason for the APE value to be higher than that for RTAB-MAP.

VI. Conclusion

The SLAM term project has served to overcome a great learning curve for the team, for a group of students who are new to the concept SLAM, serving as the perfect introduction into the different type of SLAM algorithms to apply, with various different input sensor configurations to aid in future research and work.

The conclusion based on the testing is that, monocular ORB-SLAM2 provides reliable tracking with relatively low errors. But similar to the other algorithms, the monocular system is very sensitive to sudden movement, and will lose tracking if there are moving objects in front of the camera.

As for RTAB-Map, it has a very strong visual odometry tracking algorithm. Even though the testing based on the CARLA simulations data set does not match the initial expectation in terms of generating 3D maps, it still proves its ability during the Realsense camera L515 testing. With some potential parameter exploration and tweaking, it is a powerful system to solve SLAM problems.

Google Cartographer is relatively good at loop closure detection, but suffers from the characteristic of laser data that in certain environments there exists reflections and defections, which further cause inaccurate localization and tracking. Also,

based on the testing result, it is relatively sensitive to the moving object in front of the laser scan, and lacks performance in highly dynamic environments.

Some of these 3D SLAM algorithms also support localization and tracking in 2D. With further research, these algorithms sometimes could perform just as well as other 2D based SLAM algorithms.

Overall, our team learnt how to set up different SLAM algorithms in ROS, for varied sensor suites through testing with Benchmark datasets and Simulated Data generated through CARLA simulation. Setting up multiple algorithms and porting data through the CARLA Simulator was an interesting challenge. This gave us a great insight into the inner structure of each of the algorithms, as we were directly debugging several important components of the SLAM systems. We hope to take this experience and further develop a SLAM system in ROS to solve the challenges faced within our Research.

VII. Future Work

The studies conducted in this project can be expanded to utilize more complex data-sets and additional SLAM implementations.

In the case of more complex data-sets, the CARLA simulator provides functionalities for simulating in a multitude of other conditions. To expand our study, we would generate data-sets for additional simulation conditions such as varying weather conditions that may increase slip or drag, or applying dynamic lighting conditions.

In addition to using data collected in a simulated environment, a useful expansion would be to set up a real-world mobile robot fitted with lidar and RGB-D cameras, and use this system to collect real-world data. Data-sets that could be collected with such a robot include, indoor data-sets under dynamic lighting conditions, outdoor data-sets under the real-world effects of changing wind or sunlight conditions, and an many others. The addition of this type of data collection would enable us to study more real-world accurate comparative analyses for application into our real-world based research work.

Acknowledgment

We thank Prof. Michael Kaess for teaching 16833-Robot Localization and Mapping course and motivating us throughout the project. We also extend our gratitude towards the TA's of the course for their guidance.

References

[1] Mur-Artal, Raul, and Juan D. Tardós. "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras." IEEE Transactions on Robotics 33.5 (2017): 1255-1262.

[2] Labbé, Mathieu, and François Michaud. "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation." Journal of Field Robotics 36.2 (2019): 416-446.

[3] Dosovitskiy, Alexey, et al. "CARLA: An open urban driving simulator." Conference on robot learning. PMLR, 2017.

[4] Team, CARLA. "CARLA." CARLA Simulator, carla.org/.

[5] A Benchmark for the Evaluation of RGB-D SLAM Systems (J. Sturm, N. Engelhard, F. Endres, W. Burgard and D. Cremers), In Proc. of the International Conference on Intelligent Robot Systems (IROS), 2012.

[6] Kümmerle, Rainer, et al. "On measuring the accuracy of SLAM algorithms." Autonomous Robots 27.4 (2009): 387-407.

[7] ROS,wiki.'Handheld Mapping', wiki.ros.org/HandHeldMapping/.

[8] "Computer Vision Group - Datasets - RGB-D SLAM Dataset and Benchmark." 2015. November 10, 2015. https://vision.in.tum.de/data/datasets/rgbd-dataset.

[9] Kasar, Amey. "Benchmarking and comparing popular visual SLAM algorithms." arXiv preprint arXiv:1811.09895 (2018).

[10] "Cartographer ROS Integration — Cartographer ROS Documentation." n.d. Accessed May 12, 2021. https://google-cartographer-ros.readthedocs.io/en/latest/.

A. Appendix

Bag1 Data

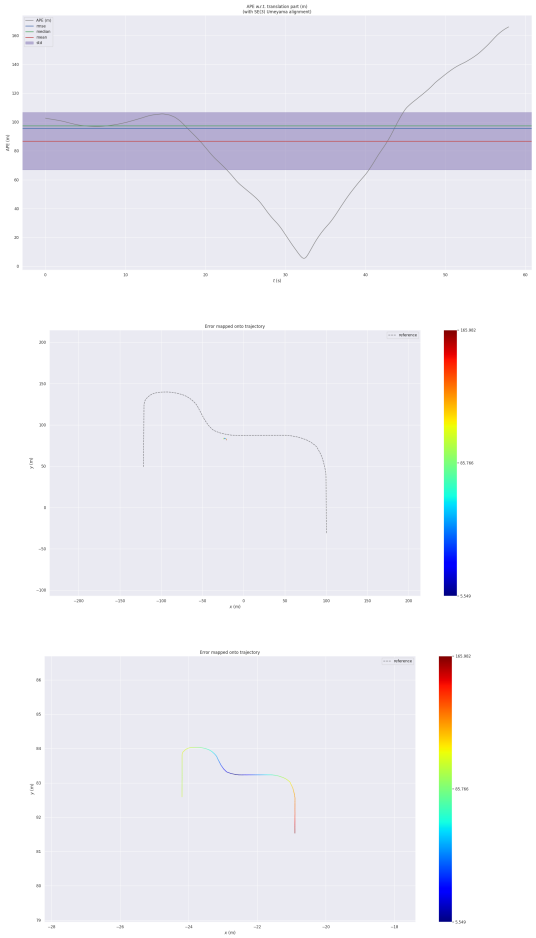

Fig. 8. APE and estimated trajectory for ORBS-SLAM2 Monocular Bag1

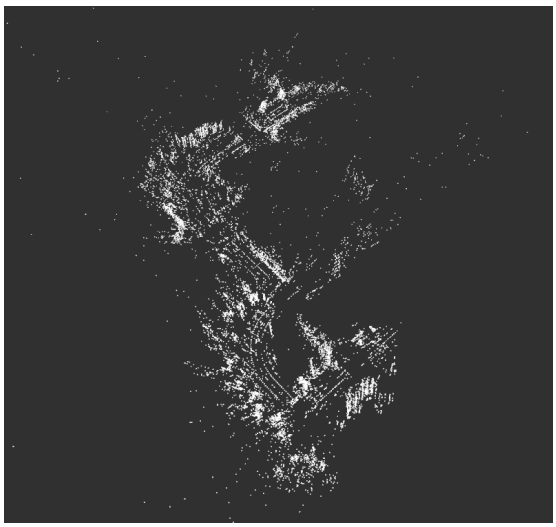

Fig. 9. Point Cloud Map for Monocular ORB-SLAM2 on Bag 1 Data

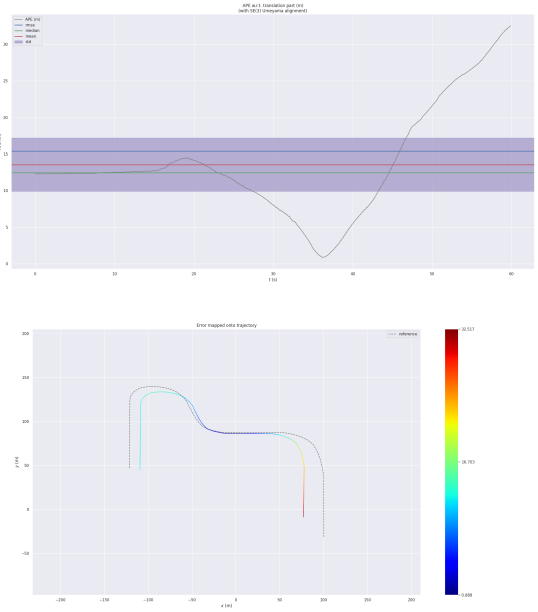

Fig. 10. APE and estimated trajectory for ORBS-SLAM2 RGBD Bag1

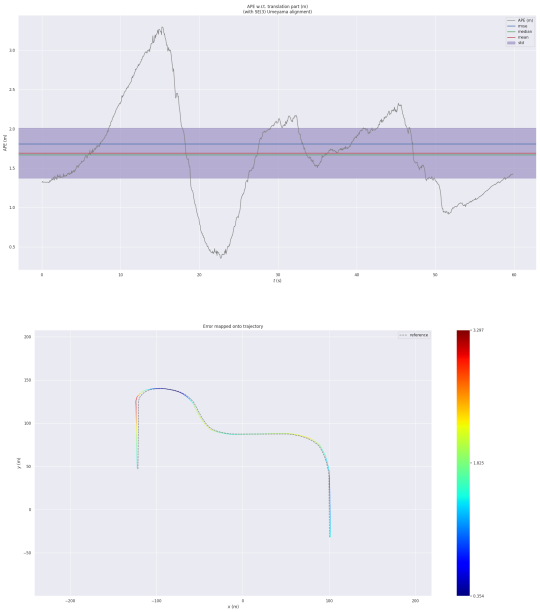

Fig. 11. APE and estimated trajectory for RTAB-MAP RGBD Bag1

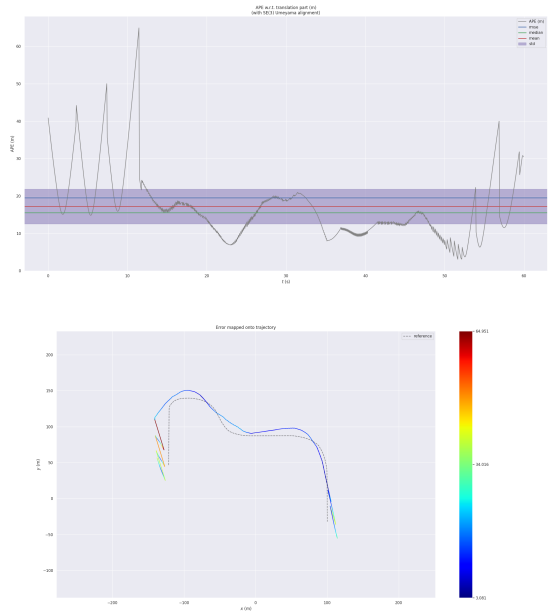

Fig. 12. APE and estimated trajectory for Cartographer Bag1

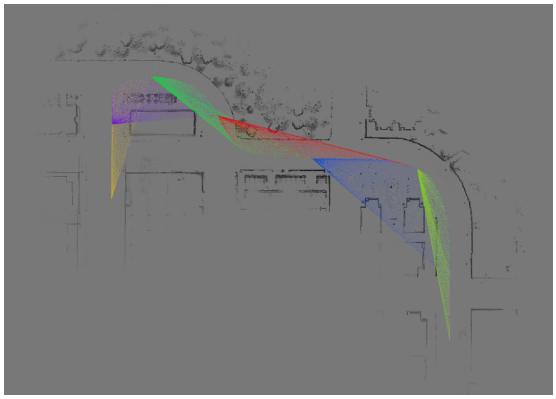

Fig. 13. Occupancy Grid Map and Pose Graph for Cartographer on Bag1 Data

Bag2 Data

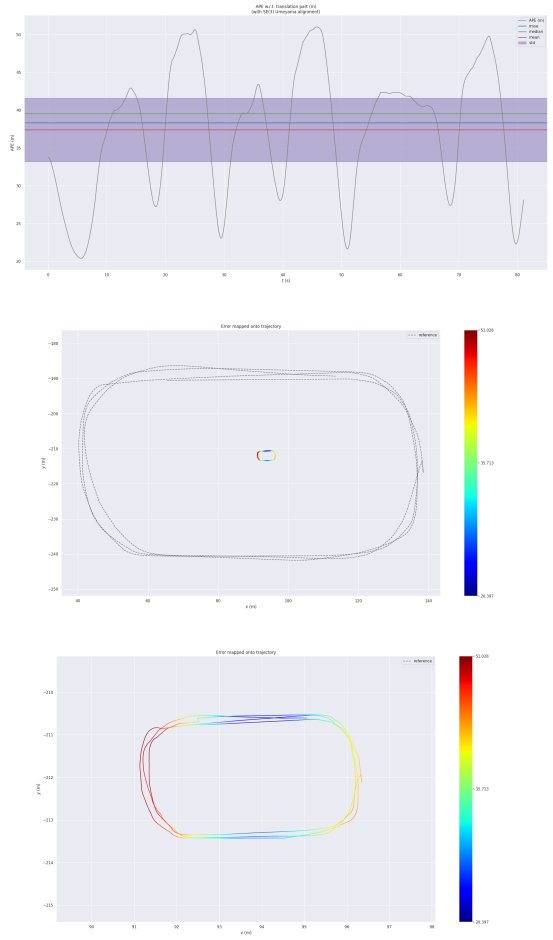

Fig. 14. APE and estimated trajectory for ORBS-SLAM2 Monocular Bag2

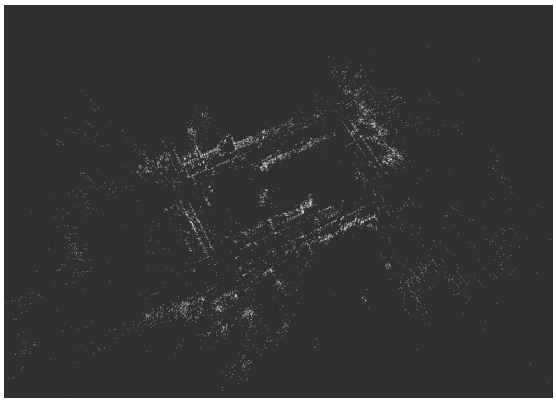

Fig. 15. Point Cloud Map for Monocular ORB-SLAM2 on Bag2 Data

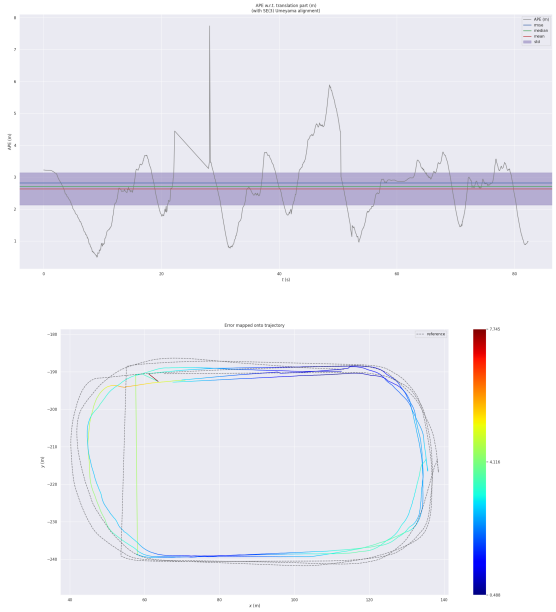

Fig. 16. APE and estimated trajectory for ORBS-SLAM2 RGBD Bag2

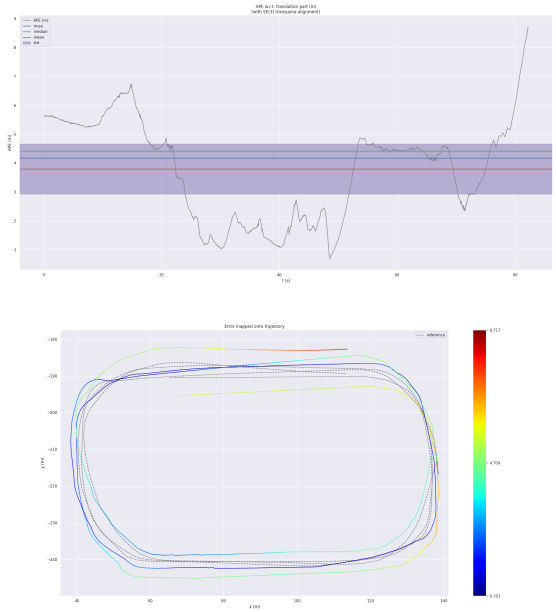

Fig. 17. APE and estimated trajectory for RTAB-MAP RGBD Bag2

Bag3 Data

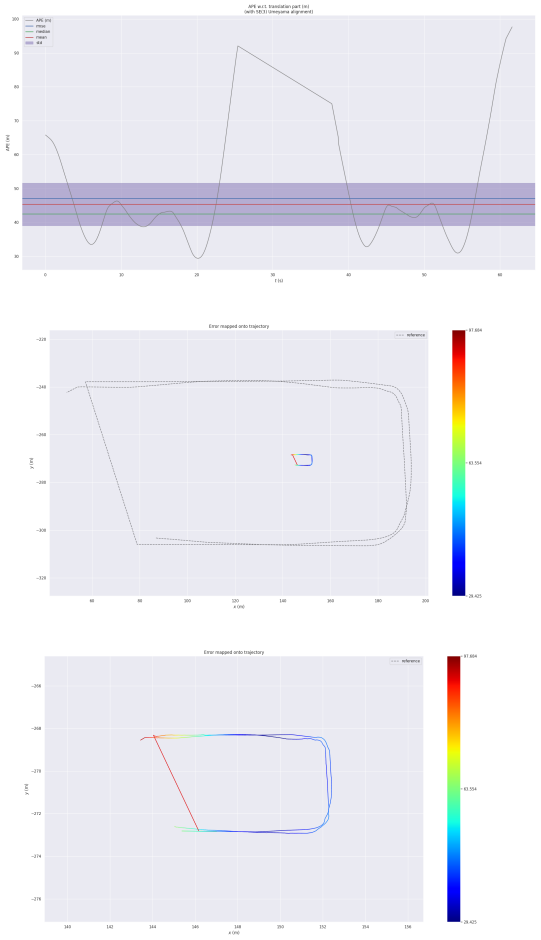

Fig. 18. APE and estimated trajectory for ORBS-SLAM2 Monocular Bag3

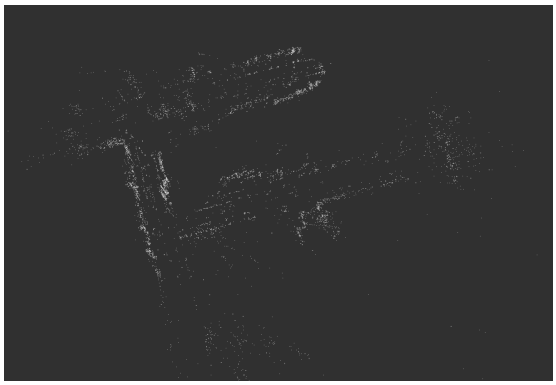

Fig. 19. Point Cloud Map for Monocular ORB-SLAM2 on Bag3 Data

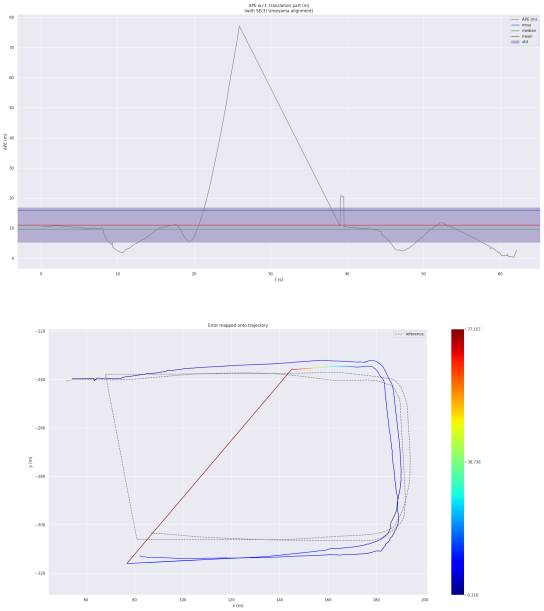

Fig. 20. APE and estimated trajectory for ORBS-SLAM2 RGBD Bag3

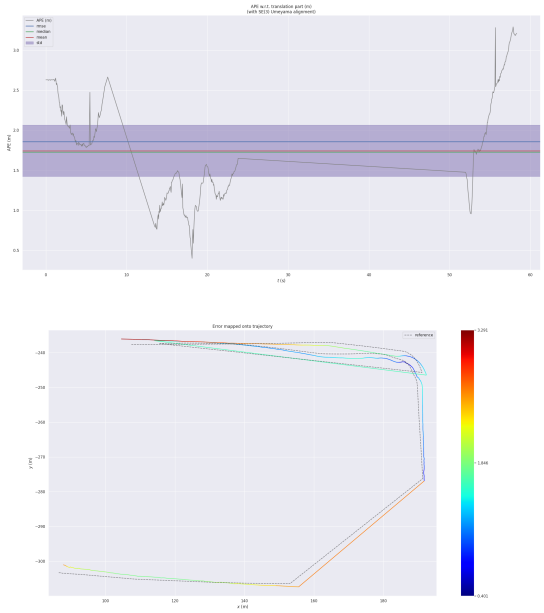

Fig. 21. APE and estimated trajectory for RTAB-MAP RGBD Bag3

Bag 4 Data

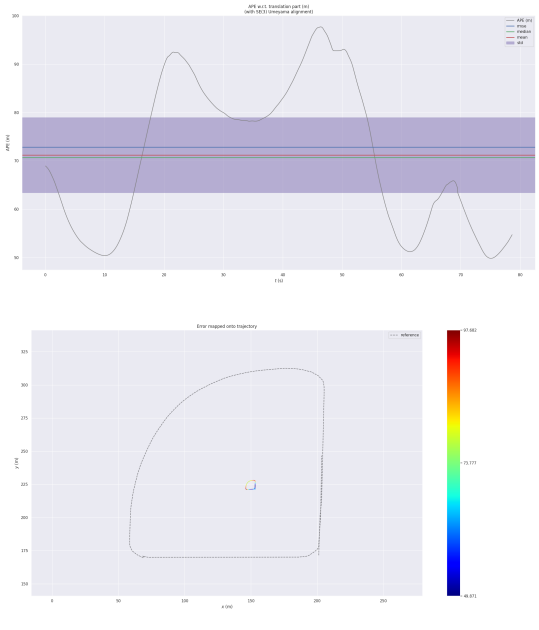

Fig. 22. APE and estimated trajectory for ORBS-SLAM2 Monocular Bag4

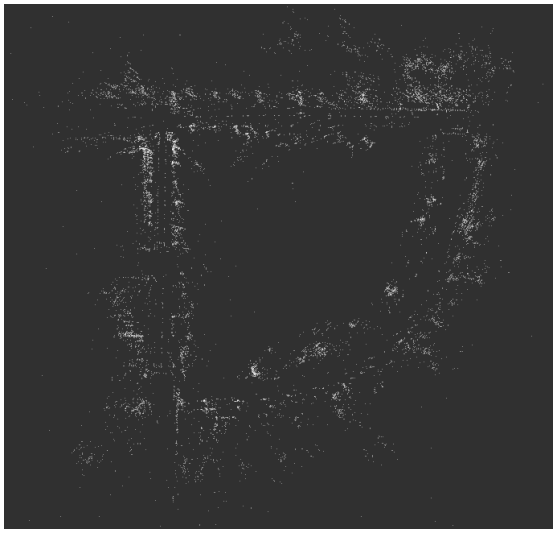

Fig. 23. Point Cloud Map for Monocular ORB-SLAM2 on Bag4 Data

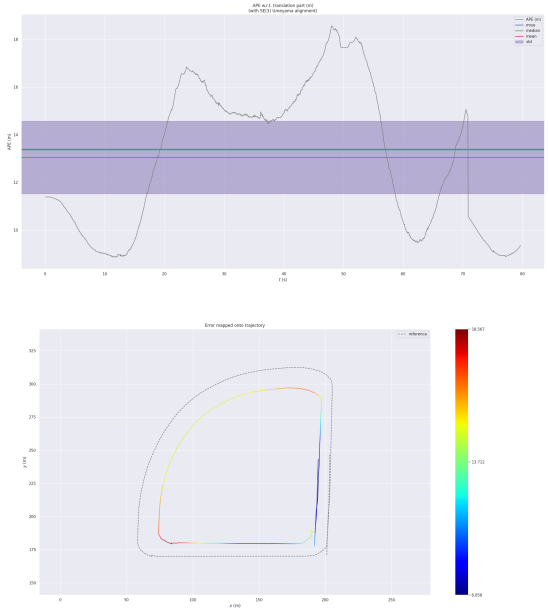

Fig. 24. APE and estimated trajectory for ORBS-SLAM2 RGBD Bag4

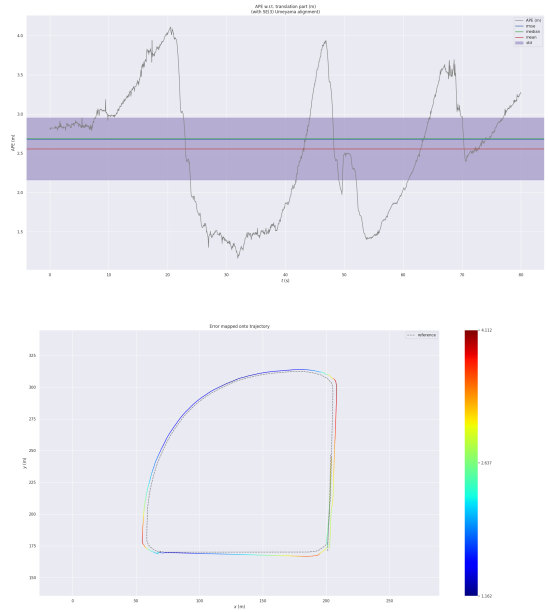

Fig. 25. APE and estimated trajectory for RTAB-MAP RGBD Bag4

TABLE III
ABSOLUTE POSE ERROR FOR CARLA BAG FILES

Dataset	ORB-SLAM2 Mono	ORB-SLAM2 RGBD	RTAB-MAP RGBD	Cartographer
Bag1	30.79	21.50	3.97	16.67
Bag2	33.30	20.11	7.87	-
Bag3	31.57	30.43	3.78	-
Bag4	27.71	23.95	4.28	-