CROCS - Comparison of RTAB-Map, OpenVSLAM, Cartographer and SPTAM

Group 2 - Jitesh Sonkusare, Nitin Somashekhar, Omkar Sargar, Ronak Bhanushali

I. Introduction

Simultaneous Localization and Mapping (SLAM) is a technology that enables autonomous robots to build a map of their environment and track their position within it, which is necessary for tasks such as path planning, navigation, and obstacle avoidance. SLAM algorithms can be used in a variety of environments and work with onboard sensors like cameras and Lidars. This report compares four different SLAM algorithms - SPTAM, OpenVSlam, RTAB-Map, and Cartographer - and evaluates their performance, accuracy, and ease of use for various applications. The report also examines the strengths and weaknesses of each algorithm in terms of accuracy and compatibility with different types of sensors. The goal of the report is to provide a comprehensive overview of these algorithms and help users choose the best one for their specific needs.

II. Method

The SLAM algorithms choosen for the comparative analysis are SPTAM, OpenVSLAM, RTAB-Map and Cartographer. This study aims to assess the relative performance of the three VSLAM algorithms and a top-performing LiDAR-based SLAM algorithm in outdoor settings. The comparison aims to verify or challenge the notion that VSLAM algorithms outperform traditional LiDAR methods, as suggested by previous research. The four algorithms will be tested and compared in order to reach a conclusion on this hypothesis

A. SPTAM

Stereo Parallel Tracking and Mapping (SPTAM) [1] is a real-time algorithm that allows a robot or autonomous vehicle to estimate its 3D pose by analyzing images from a stereo camera. It extracts and tracks feature points in each frame and estimates the camera's pose from one frame to the next, using ORB feature extraction and a keyframe-based approach for loop detection and correction. SPTAM parallelizes camera tracking and map optimization to achieve real-time performance, and uses bundle adjustment to refine estimates of the camera pose and feature points in the map [2]. It reduces storage and computational costs by using binary features to describe visual landmarks and achieves real time execution [15].

B. OpenVSLAM

OpenVSLAM [3] is a visual SLAM system that uses monocular, stereo, and RGBD approaches and consists of various components with clear APIs that can be easily integrated into other programs. The software is organized into three main modules: tracking, mapping, and global optimization [4]. The tracking module calculates the camera pose for each frame and determines if a new keyframe should be inserted. The mapping module creates and extends the map using the inserted keyframes and performs local bundle adjustment [5]. The global optimization module performs loop detection, pose-graph optimization, and global bundle adjustment to correct drift [16].

C. RTAB-Map

RTAB-Map [6] is a RGB-D, stereo, and Lidar graph-based SLAM approach that uses an incremental appearance-based loop closure detector and bag-of-words method to determine if a new image is more likely to come from an old location or a new one. It adds a new constraint to the map's graph when a loop closure hypothesis is accepted and uses graph optimization to reduce map errors. RTAB-Map can use a variety of input topics, such as RGB-D images, stereo images, odometry, laser scans, and point clouds, but requires at least registered RGB-D or calibrated stereo images with odometry to work.

D. Cartographer

Cartographer [7] is a real-time graph-based SLAM algorithm that primarily uses 3D or 2D LiDAR and IMU sensor data for mapping and localizing. It generates "submaps" using Lidar data, matches subsequent scans to these submaps using the Ceres Scan matcher to find the best pose estimate, and finds loop closure constraints for optimization using a branch and bound approach by matching submaps in a given search window. It can also accept GPS fix and odometry data to improve accuracy. Cartographer has a front end (Local SLAM) for scan-to-submap matching and a back end (Global SLAM) for optimization. In this study, Cartographer was used with 3D Lidar and IMU for mapping and localizing [13].

III. Experiment Description

The KITTI dataset [8] is a benchmark for evaluating the performance of visual odometry, SLAM, and other computer

TABLE I
SUMMARY OF THE SLAM ALGORITHMS USED

Algorithm	Feature Extractor	Loop Closure	Graph Optimizer
SPTAM	ORB feature extractor	DBoW2	Pose graph optimization
OpenVSLAM	ORB feature extractor	DBoW2	g2o
RTAB-Map	GFTT	Bag-of Words approach	TORO, g20 and GTSAM
Cartographer	Ceres scan matcher	Submap matching	Grid Based

vision algorithms for autonomous vehicles and robotics. The vehicle used for recording the dataset is a modified Volkswagen Passat B6 with an OXTS RT 3003 Inertial Navigation Unit, Velodyne HDL-64E Laser Scanner, two Point Grey Flea 2 (FL2-14S3M-C) grayscale cameras, two Point Grey Flea 2 (FL2-14S3C-C) color cameras and four Edmund Optics NT59-917 Varifocal Lenses. The setup and positions of the sensors is seen in Figure 1. The dataset consists of annotated stereo images, LiDAR point clouds, and GPS/IMU measurements collected from various driving scenarios captured in the mid-size city of Karlsruhe. Accurate ground truth is provided by a Velodyne laser scanner and a GPS localization system. This report uses Sequence 00 of the Visual Odometry / SLAM Evaluation 2012 dataset to evaluate the above-discussed SLAM algorithms. These algorithms use different types of data from the KITTI dataset, with SPTAM, OpenVSLAM and RTAB-Map using only stereo images and Cartographer using LiDAR and IMU. To compare and evaluate the trajectories generated by the SLAM algorithms, the ground truth poses in the KITTI dataset are used as a reference to measure the accuracy and performance of the algorithms under different conditions.

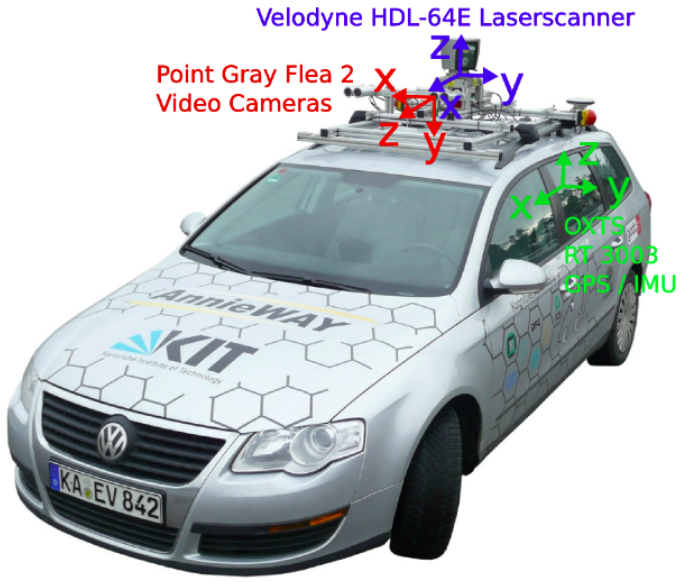

Fig. 1. Orientation and position of sensors on the car used in the Kitti dataset

IV. RESULTS

A. Performance Metrics

The following metrics are used to examine and evaluate the performance of the algorithms.

- Translational RPE: Translational Relative Pose Error measures the difference in position between the estimated and true values.

$$RPE_{trans} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}||trans(F_i)||^2}$$

- Rotational RPE: Rotational Relative Pose Error measures the difference in position between the estimated and true values.

$$RPE^{i,\Delta}_{rot\,rot} = \frac{1}{m}\sum_{i=1}^{m}\angle(rot^{\Delta}_{i})$$

- ATE: The ATE is defined as the root mean square error from error matrices by paring the absolute distances between estimated and ground truth trajectory after aligning them.

$$ATE_{rmse} = \frac{1}{m}\sum_{i=1}^{m}||trans(E_i)||^2$$

B. SPTAM

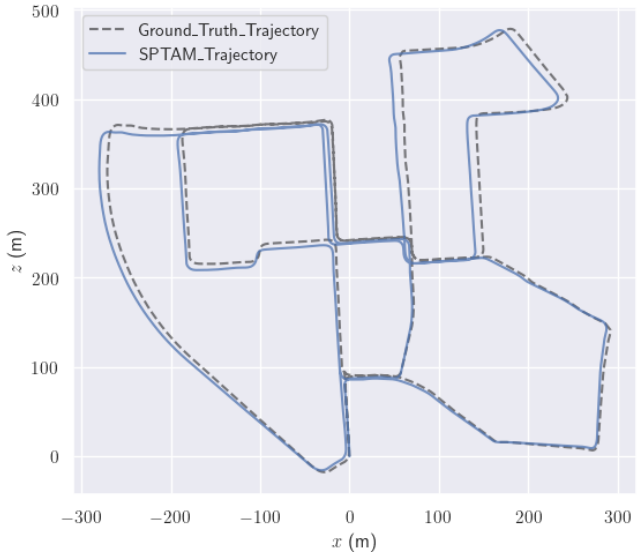

Fig. 2. Comparison of the trajectory generated by SPTAM with the ground truth

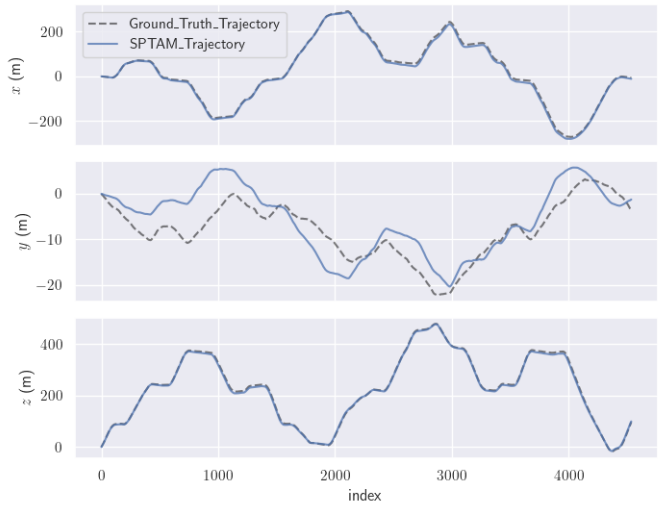

Fig. 3. Comparison of the positions in x,y and z as calculated by SPTAM with the ground truth

Fig. 5. Comparison of the positions in x,y and z as calculated by OpenVSLAM with the ground truth

C. OpenVSLAM

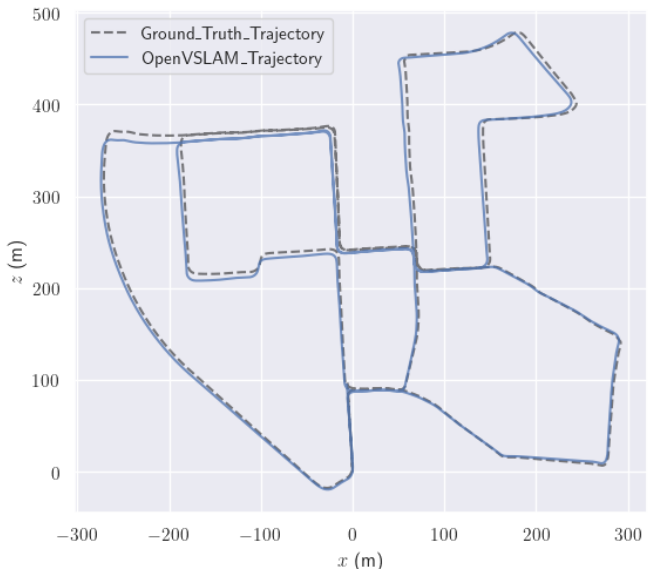

Fig. 4. Comparison of the trajectory generated by OpenVSLAM with the ground truth

D. RTAB-Map

The researchers attempted to implement RTAB-Map, but encountered various issues that prevented them from progressing to a point where it could be compared with the other algorithms. Some of the major issues faced were:

- RTAB-Map struggles to function properly when the robot is moving at relatively high speeds. On the dataset used, it consistently crashed at 35 seconds because it failed to identify the minimum number of common features needed to calculate odometry between two consecutive frames.‘
- In order to create a map, it was necessary to run the rosbag at a rate that was five times slower than the actual rate. This raises concerns about the real-time capabilities of RTAB-Map.
- The features selected for stereo odometry appeared to be biased towards the left side of the images. This caused difficulties when there were insufficient good features on the left side.

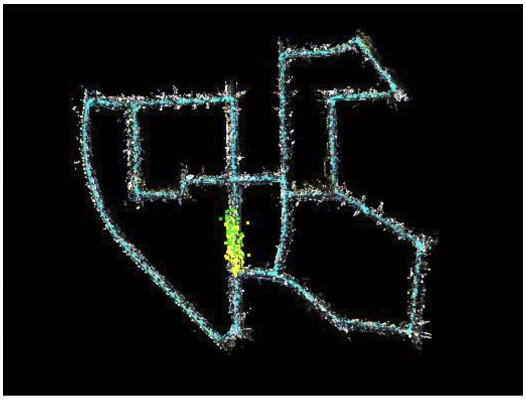

Fig. 6. 3D map created using RTAB-Map

E. Cartographer

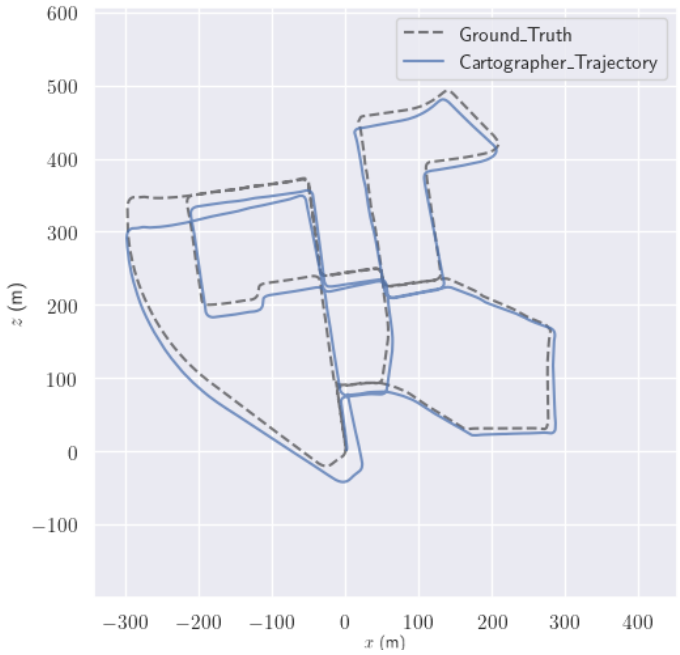

Fig. 7. Comparison of the trajectory generated by Cartographer with the ground truth

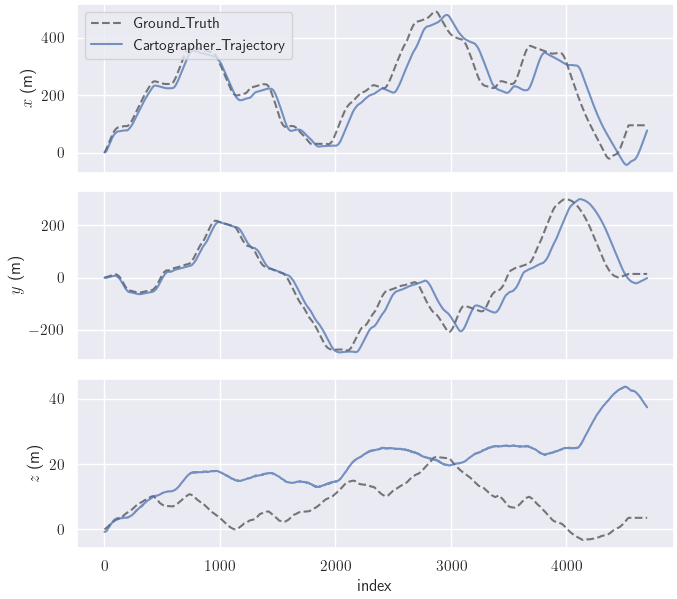

Fig. 8. Comparison of the positions in x,y and z as calculated by Cartographer with the ground truth

F. Analysis

As seen in Figs. 2 and 4, OpenVSLAM's anticipated trajectory is more similar to the actual trajectory than SPTAM's. Both OpenVSLAM and SPTAM's errors in the x and z directions agree with ground truth values, as shown in Figs. 3 and 5, while SPTAM's error in the y direction is noticeably larger. Given that the camera's y-axis is angled downward and in the direction of gravity, this is a result of the potholes and unevenness in the road (as shown in Fig 1). Overall, OpenVSLAM outperforms SPTAM in handling error encountered in the y direction [9].A comparison of the variations between the three algorithms is shown in Table 2. It demonstrates that SPTAM has a larger rotational error (0.344m) than OpenVSLAM (0.019m), which is caused by the fact that SPTAM's feature extractor doesn't work as effectively during abrupt turns. OpenVSLAM (7.79m) and SPTAM (9.225m) have lesser errors than Cartographer in terms of absolute positional error (56.667m) [10]. This is primarily due to the fact that although Cartographer and other lidar-based algorithms work better indoors, visual slam algorithms do better outside [14].

TABLE II
TRANSLATIONAL RPE, RELATIONAL RPE AND APE WITH RESPECT TO THE GROUND TRUTH TRAJECTORIES

SLAM Algorithms	**Translational RPE (m)**	**Rotational RPE (deg)**	**APE (m)**
SPTAM	0.0349	0.344	9.225
OpenVSLAM	0.059	0.019	7.790
Cartographer	2.05	4.53	56.667

Cartographer algorithm performed poorly in the outdoor environment of this study, as demonstrated by the errors in all three dimensions (x, y, and z) shown in Fig 8. This is due to Cartographer's primary design for indoor environments, as mentioned in [7]. The fast-moving and dynamic nature of the outdoor environment, with moving cars and pedestrians and repeating static features like sidewalks, may have contributed to inaccurate scan matching. Additionally, the use of IMU data along with the lidar dataset and the drifting in the gyroscope and magnetometer may have impacted the accuracy of the mapping over time, as shown in Fig 8. The lack of computational resources may also have played a role in the poor performance of Cartographer in this case, as a high number of iterations and confidence are needed for accurate scan matching. However, it was noted that the car had traveled quite a distance by the time one scan was matched with its prior submap, resulting in a lack of features that overlap with the prior submap.

The adoption of the Orb feature extractor, one of the fastest feature extractors available, allows the OpenVSLAM and SPTAM algorithms chosen for this study to operate in real time [12]. Fast visual slam algorithms are preferable in outdoor settings with dynamically moving

objects because they consume fewer computational resources and can operate in real time. This allows for accurate mapping of features and rejection of outliers using effective RANSAC operations, resulting in accurate pose estimation from each captured frame. Visual slam algorithms extract a large number of feature points in each frame, which are used as reference points from frame to frame. The detection of a large number of feature points is crucial for finding correspondences in a highly dynamic and fast-moving environment, as seen in the KITTI dataset. The use of the RANSAC method helps to remove outliers and find accurate correspondences, which are used to calculate accurate poses from frame to frame. As shown in the table 2, the VSLAM algorithms [11] have lower rotational and translational RPE values compared to Cartographer [17].

G. Conclusion

In conclusion, OpenVSLAM and SPTAM performed better than Cartographer in this study in terms of accuracy in outdoor environments. OpenVSLAM showed particularly strong performance in handling error in the y direction, while SPTAM had lower rotational error than OpenVSLAM. Both OpenVSLAM and SPTAM used fast feature extractors and the RANSAC method to accurately map features and calculate poses, allowing for real-time operation in a dynamic outdoor environment. Cartographer, on the other hand, performed poorly due to its primary design for indoor environments and the impact of computational resources on its accuracy. Overall, visual slam algorithms tend to perform better in outdoor environments while lidar-based algorithms like Cartographer are more suitable for indoor environments.

REFERENCES

[1] Pire, T., Fischer, T., Castro, G., De Cristóforis, P., Civera, J., Berlles, J. J. (2017). S-PTAM: Stereo parallel tracking and mapping. Robotics and Autonomous Systems, 93, 27-42.

[2] de Jesus, K. J., Kobs, H. J., Cukla, A. R., Cuadros, M. A. D. S. L., Gamarra, D. F. T. (2021, October). Comparison of Visual SLAM Algorithms ORB-SLAM2, RTAB-Map and SPTAM in Internal and External Environments with ROS. In 2021 Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE) (pp. 216-221). IEEE.

[3] Sumikura, S., Shibuya, M., Sakurada, K. (2019, October). OpenVSLAM: A versatile visual SLAM framework. In Proceedings of the 27th ACM International Conference on Multimedia (pp. 2292-2295).

[4] Raúl Mur-Artal and Juan D. Tardós. 2017. ORB–SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. IEEE Transactions on Robotics 33, 5 (2017), 1255–1262. https://doi.org/10.1109/TRO.2017.2705103

[5] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. 2010. A Tutorial on Graph-Based SLAM. IEEE Transactions on Intelligent Transportation Systems Magazine 2, 4 (2010), 31–43. https://doi.org/10.1109/MITS.2010.939925

[6] Labbé, M., Michaud, F. (2019). 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), 416-446.

[7] Hess, W., Kohler, D., Rapp, H., Andor, D. (2016, May). Real-time loop closure in 2D LIDAR SLAM. In 2016 IEEE international conference on robotics and automation (ICRA) (pp. 1271-1278). IEEE.

[8] Geiger, A., Lenz, P., Stiller, C., Urtasun, R. (2013). Vision meets robotics: The kitti dataset. The International Journal of Robotics Research, 32(11), 1231-1237.

[9] Grupp, Michael (2017). Evo. https://www.github.com/MichaelGrupp/evo

[10] Zhan, Huangying (2019). kitti-odom-eval. https://github.com/Huangying-Zhan/kitti-odom-eval

[11] Prokhorov, D., Zhukov, D., Barinova, O., Anton, K., Vorontsova, A. (2019, May). Measuring robustness of Visual SLAM. In 2019 16th International Conference on Machine Vision Applications (MVA) (pp. 1-6). IEEE.

[12] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, ¨ "g2o: A general framework for graph optimization," in Robotics and Automation (ICRA), 2011 IEEE International Conference on. IEEE, 2011, pp. 3607–3613.

[13] J. Clausen, "Branch and bound algorithms-principles and examples," Department of Computer Science, University of Copenhagen, pp. 1– 30, 1999.

[14] Rauf, A., Irshad, M. J., Wasif, M., Rasheed, S. U., Aziz, N., Taj, H. (2022). Comparative Study of SLAM Techniques for UAV. Engineering Proceedings, 12(1), 67.

[15] Pire, Taihú and Fischer, Thomas and Castro, Gastón and De Cristóforis, Pablo and Civera, Javier and Jacobo Berlles, Julio (2017) https://github.com/lrse/sptam

[16] Sumikura, S.,(2021, May). OpenVSLAM(pp. 1-47).

[17] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (pp. 3354–3361). https://doi.org/10.1109/CVPR.2012.6248074