Road Sign Detection and Comparative Analysis of SLAM and VSLAM Algorithms

Arjun Rajeev Warrier, Arsh Tangri, Divyansh Gupta, Sahil Kiran Bodke, Sriram Pandi

I. Introduction

Autonomous vehicles operating in real life settings must be able to navigate in large, unstructured, dynamic environment and make sense of semantic information from environment to make decisions. Visual SLAM and SLAM are two of the most common methods used today for mapping the environment and simultaneously estimating autonomous vehicles ego and location on the map. However, real world constraints in terms of cost, accuracy, computation, power and ease of integration poses challenges on the sensor suit configuration and SLAM approaches. In terms of semantic information, one crucial safety factors for autonomous vehicles could be registering and mapping of road signs. This issue is critical in terms of road safety especially in case of self-driving cars. In this project, we have tried to address these challenges by doing a comparative analysis of different sensors (Stereo cameras, IMU, 3D Lidar) and algorithms (VSLAM-RTABMap, ORB Slam 3; SLAM - LEGO LOAM, HDL-Graph SLAM) configuration along with understanding the implementation challenges of each algorithm. This would help in better understanding the limitations and benefits of different sensor and algorithm configurations for self-driving cars use-case. Further, we implemented a neural network based technique for road-sign detection. Major criteria for selecting the SLAM, V-SLAM and Road-sign detection algorithms was their relevance in current autonomous vehicle space. Additionally, time-constraint was another criteria for choosing these algorithms that allows implementation within project time-frame (3 weeks). The code for the project can be found in the following repository: https://gitlab.com/gupta.divy/eece5554/-/tree/main/group_prj

II. Dataset

We worked with KITTI-Sequence 00 dataset collected by Karlsruhe Institute of Technology [2], and converted to rosbag using kitty2bag package by Tomas. This dataset is commonly used for benchmarking of SLAM and V-SLAM algorithms and it contains data from different sensor configurations including stereo cameras, IMU, GPS and LIDAR. In our project, algorithms used were based on loop-closure and this KITTI data sequence has multiple loop-closures which made it relevant for our analysis.

Fig. 1. KITTI sensor placement [2]

Sensors used in our project:

1) **RGB Camera:** RGB camera is based on the capture and separation of light into its component colours using red, green and blue filters.
2) **LIDAR:** It uses a laser based range finding technique called "time-of-flight" measurement to determine distance to objects in its field of view. The laser rangefinder measures the distance to nearby objects by emitting a laser beam and measuring the time it takes for the beam to bounce back.
3) **IMU:** IMUs have three components accelerometer, gyroscope and magnetometer. Gyroscope and Accelerometer works on the principle of detecting change in forces acting on suspended mass (via spring and dampers) due to acceleration and rotation of sensor. Magnetometer works on the principle of Faraday's law of induction.

III. Methodology

A. SLAM

SLAM algorithms use lidar data in the form of point clouds to build environments through scan matching and optimisation. Two such lidar algorithms were implemented for the purpose of this project.

1) LeGO-LOAM: LeGO-LOAM is a light-weight SLAM algorithm that achieves similar or better performance against state-of-the-art SLAM methods like LOAM, but with significantly reduced computation. Instead of extracting features directly from a point cloud, which is the case in most Lidar-based algorithms, LeGO-LOAM first projects a 3D LIDAR scan onto a range image, and then applies Image-based segmentation on the obtained range-image to group points into clusters. Points belonging to the ground are considered as a separate cluster. The segmented points and the ground points are then used for feature extraction, following which, the Lidar Odometry Module and Lidar Mapping module uses the features to estimate the transformation between consecutive scans and to register them in a global point-cloud map. Fig.2 gives an overview of the algorithm.

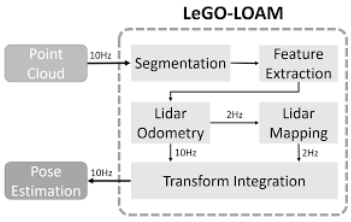

Fig. 2. LeGO-LOAM Flow Chart

2) HDL-Graph Slam: Graph SLAM tackles the SLAM problem by generating and optimizing a graph in which the nodes indicate the parameters to be optimized, such as vehicle poses or landmark locations. The edges of the said graph would then describe constraints that are applied in the optimisation, such as relative poses between the generated nodes in terms of likelihoods. High definition lidar graph slam or HDL graph slam, was developed by Koide et al [4],

for application in a portable people behaviour measurement system. This has been adapted for a vehicle mounted lidar. In the offline mapping phase, we create a 3D environmental map which covers the entire measurement area. For the mapping, we employ a graph optimization-based SLAM approach (i.e., Graph SLAM [3]). The system estimates its pose on the map created offline by combining a scan matching algorithm with an angular velocity-based pose prediction using Unscented Kalman filter. The same is shown in Fig.3.

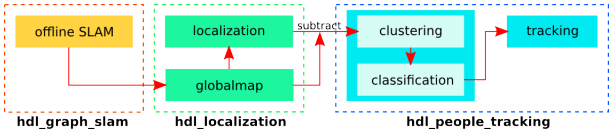

Fig. 3. HDL Graph SLAM Flow Chart

B. Visual SLAM

Visual SLAM algorithms use camera data(stereo or RGBD) in the form of image frames to build environments through feature correspondence. Two industry standard VSLAM algorithms were implemented in this project.

1) RTABMap: Real-Time Appearance-Based Mapping is a graph based SLAM approach based on loop closure detector that uses bag-of-words approach to determine how likely a new image is from a previous location[5]. RTABMap package allows flexibility in use of different feature detectors, mapping methods and external odometry as input, which make it easy to integrate with any sensor configuration. Further, it also has memory management system and multi-session mapping integration which makes it useful to work in real-time with large-scale environments. Working flow of RTABMap for stereo camera based visual SLAM is shown in Fig. 4.

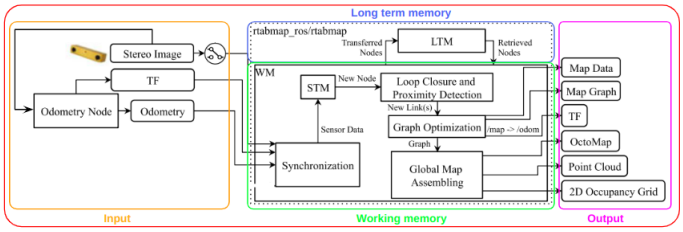

Fig. 4. RTABMap algorithm working

2) ORB Slam 3: ORB-SLAM3 is a complete open source library for visual and visual-inertial, and multisession SLAM and relies on MAP estimation and BA for real-time operation. It is built upon ORB-SLAM[7] and ORB-SLAM2[8]. The major novelties of this algorithm are introduction of an integrated library, fast IMU initialization and multisession mapping. It uses Atlas, a set of disconnected maps, which is a database of keyframes. The incoming frames are computed and compared with this database and its keyframes are added to the active map. Using FAST and BRIEF, feature matching is done to carry out loop correction and map merging. Fig 5 gives an overview of the algorithm.

C. Road sign Detection and Classification

Road sign detection and classification involved the use of the YOLO-v5 model to effectively detect and classify road signs. The detection model was trained on the German Traffic Sign Detection Benchmark (GTSDB) using 666 and 74 RGB images for training

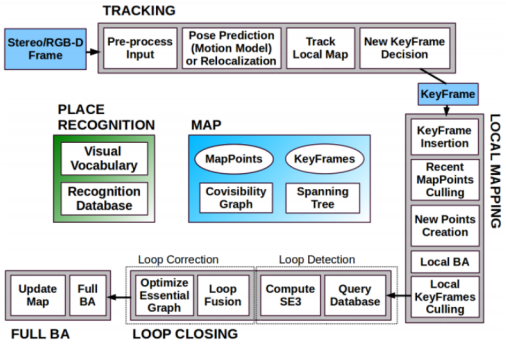

Fig. 5. ORB-SLAM3 Flow Chart

and validation, respectively. The obtained model was used to detect the road signs ,and classify them into one among the 4 categories, i.e., Speed Limit Sign, Yield to Pedestrians Sign, Yield to Incoming Traffic Sign, and Mandatory Turn signs. After training, the model was used for detecting road signs in the Kitti dataset, and the locations of the detected signs were then marked at their corresponding location on the trajectories obtained from the SLAM algorithms. An example of this can be seen in Fig.6

Fig. 6. Object Detection using YOLOv5

IV. Analysis and Results

A. LeGO-LOAM

The algorithm proposed in [9] has been adapted for implementation with the KITTI dataset. The following changes were made:

- As the original implementation was speficifally optimized for a horizontally placed VLP-16 LIDAR while the KITTI dataset was collected using the Velodyne-64 LIDAR, LIDAR-specfic parameters had to be changed to account for the use of Velodyne-64.
- Since the KITTI dataset was pre-processed to remove the effects of motion-distortion, the Point-Cloud Distortion correction step had to be ignored in our implementation.

A high RMSE of **26.147 meters** was obtained after comparison with the ground truth poses. The comparison can be seen in Fig.7.

B. HDL Graph SLAM

The algorithm proposed by Koide has been adapted to a vehicular SLAM problem for implementation with the KITTI dataset. The height at which the sensor is located and other calibration details are included along with the recorded point clouds in a rosbag with the assistance of pre-defined python libraries.
Some issues that were raised during the implementation are:

- As an algorithm aimed at covering short distances in small duration (on a backpack), the initial implementation gives high definition 3D maps with large amounts of recurring features for easier scan matching results. However, on a moving vehicle, the

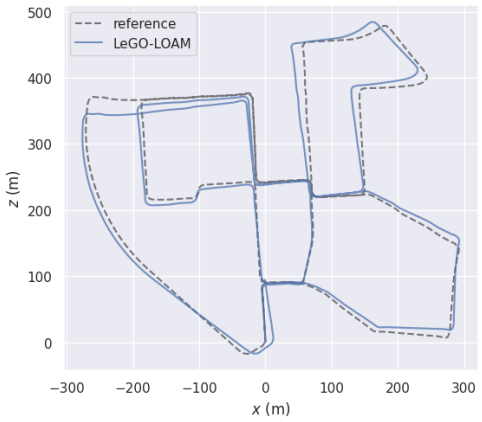

Fig. 7. Comparison between Estimated and Ground Truth Trajectories for LeGO-LOAM

amount of data to be processed in the same amount of time is also increased twofold while having less recurring features. To solve these issues, the data has to be down-sampled and the scan matching constraints had to be relaxed.

- With the relaxed conditions, inaccuracies arose in the pose estimation; and thereby inaccurate transformations and shifts occurs in the data. To resolve this, inertial data was included in the pose optimisation algorithm (in a manner much similar to dead reckoning).

These steps finally give a result that is highly similar to the actual path as seen from the camera data. However, after the 37 second mark, there seems to be some lapse in the collection of data. This lapse in data, causes the graph slam to lose its likelihood estimation and the map after this mark can be seen in red colour, which implies estimations with less than optimal likelihoods in Fig.8.
While this algorithm might be slower compared to the other algorithms, it does give exceptionally detailed maps from just point clouds that are comparable to the results of its visual counterparts.

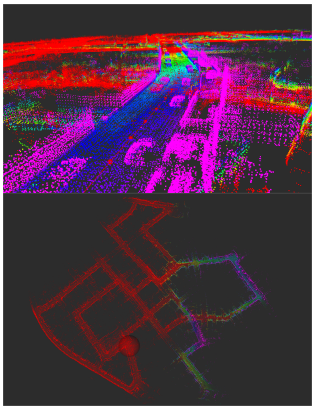

Fig. 8. Top view(down) and follower view(top) of KITTI output for Graph SLAM.

C. RTABMap

Implementation of RTABMap was done using ROS-RTABMap package published by labbe et al [4] on KITTI and NUANCE car dataset, however it failed after 36 seconds with KITTI data and didn't worked with NUANCE data. Working with KITTI data raised multiple issues related to mapping environment for which we tuned visual registration, feature detection and other 3D mapping parameters to improve performance (working from 25 seconds initially to 35 seconds later). Issues with implementing RTABMap on NUANCE dataset were related to incompatible data collection. Some of the issues raised are highlighted here

- RTABMap requires input data feed to have correct transformations (base-link to sensors) and stereo calibrated camera-information published in-sync. In Nuance rosbag, camera information was based for extrinsic monocular calibration. Also, the /tf topic on Nuance rosbag didn't had any transformations between the base-link and sensor frames.
- In KITTI data at around 35 seconds, visual odometry fails to register visual features between corresponding frames. We tried tuning maximum feature detected on image, using different feature detection algorithms but it still failed on that frame as shown in Fig.9

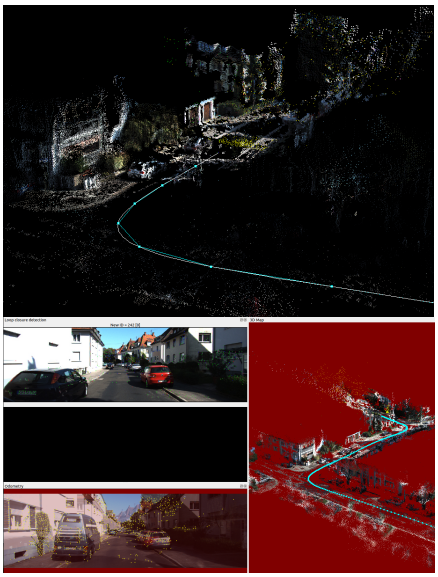

Fig. 9. Top: 3D output of RTABMap, Bottom: Frame showing failure in registering visual odometry

In [6] accuracy on KITTI dataset for RTABMap F2M configuration applied in this project resulted in average predicted error of 1m. Further, failure in this algorithm could be resolved by integrating Robot localization package by Charles River Analytics, which enables EKF based fusion of Visual and Inertial Odometry.

D. ORB-SLAM3

The implementation of ORB-SLAM3 as proposed in [1] was carried out on the Kitti dataset. Following issues were encountered while implementing ORB-SLAM3:

- The repository containing the algorithm seemed to work well only on a fresh installation of ubuntu as multiple errors occurred during setup on older ubuntu installation.
- Multiple errors regarding python, opencv and c++ version compatibility were encountered which demanded peculiar tweaks such as modifying CMakeLists.txt file for correcting opencv version, using the command - *sed -i's/++11/++14/g' CMakeLists.txt* - to resolve c++ version compatibility issues and *ldconfig* for creating required links and cache.

- After setting up the repository, the algorithm was tested. It was found to map the entire sequence of KITTI dataset with a mean tracking time of 0.046s and achieves desirable loop closure performance which is evident from Fig 10. The RMSE obtained after comparison with the ground truth poses was **1.2 meters**.

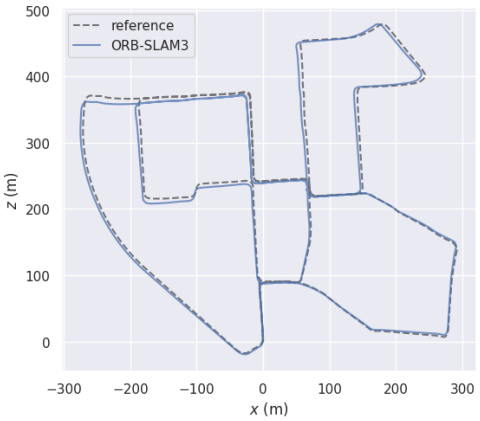

Fig. 10. Comparison between Estimated and Ground Truth Trajectories for ORB-SLAM3

E. Road Sign Detection and Classification

Our Object Detection model achieved an accuracy of 0.868 on the validation images for the task of detecting the four classes of road signs. The Object Detection model was then successful used for inference for the task of identifying the road signs in the KITTI dataset, and then marking them along the trajectory generated using the SLAM algorithms. Fig.11 shows the marked road signs on trajectories proposed by LeGO-LOAM and ORB-SLAM3, respectively.

V. Conclusions

Four different solutions to the SLAM problem; namely RTABMap, ORB SLAM3, LeGO-LOAM and an offline 3D graph slam were implemented on the KITTI dataset, wherein ORB-SLAM3 performed the best. The different issues and challenges faced were recorded and satisfactory results were obtained. The primary objective of road sign detection was also successfully completed and the signs were marked on the generated trajectories.

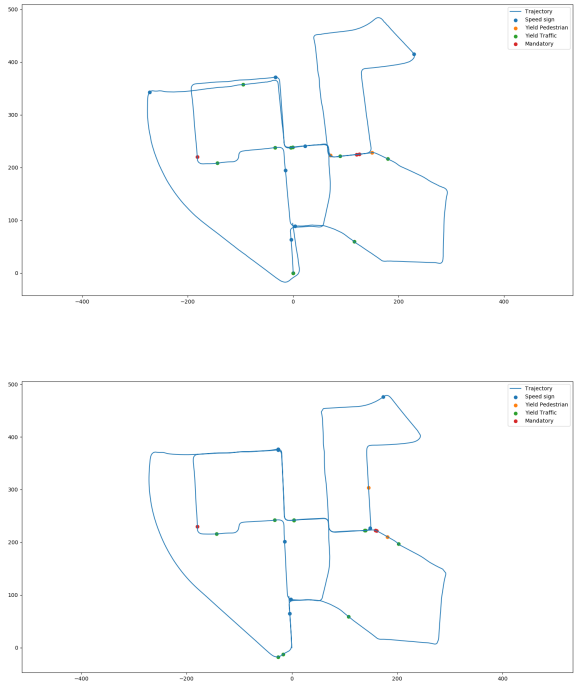

Fig. 11. Mapping of signs on LEGO (top) and ORB (bottom) generated trajectory

References

[1] Carlos Campos et al. “Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam”. In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1874–1890.

[2] Andreas Geiger et al. “Vision meets Robotics: The KITTI Dataset”. In: *International Journal of Robotics Research (IJRR)* (2013).

[3] Giorgio Grisetti et al. “A tutorial on graph-based SLAM”. In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43.

[4] Kenji Koide, Jun Miura, and Emanuele Menegatti. “A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement”. In: *International Journal of Advanced Robotic Systems* 16.2 (2019), p. 1729881419841532.

[5] Mathieu Labbe and Francois Michaud. “Appearance-based loop closure detection for online large-scale and long-term operation”. In: *IEEE Transactions on Robotics* 29.3 (2013), pp. 734–745.

[6] Mathieu Labbé 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”. In: *Journal of Field Robotics* 36.2 (2019), pp. 416–446.

[7] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. “ORB-SLAM: a versatile and accurate monocular SLAM system”. In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163.

[8] Raul Mur-Artal and Juan D Tardós. “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras”. In: *IEEE transactions on robotics* 33.5 (2017), pp. 1255–1262.

[9] Tixiao Shan and Brendan 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)*. 2018, pp. 4758–4765. DOI: 10.1109/IROS.2018.8594299.