LOAM Algorithms for Analysis on NUANCE Datasets

Srijan Dokania, Isabelle Brandicourt, Aayush Sanghvi, Pankaj Gopi
Northeastern University, Boston, MA, USA
[dokania.s, brandicourt.i, sanghvi.aay, gopi.pa]@northeastern.edu

Abstract

This report explores advanced pose estimation and mapping techniques in autonomous robot navigation, focusing on LeGO-LOAM and LiO SAM methods. Utilizing data from Northeastern University's autonomous car, NUance, we analyzed these techniques for their practicality in real-world scenarios. LeGO-LOAM builds upon the basic LOAM algorithm by adding LiDAR scans and ground feature extraction, improving orientation accuracy. LiO SAM further advances this by incorporating odometry with smoothing, IMU, and GPS data, offering comprehensive mapping capabilities at the cost of increased computational intensity. Our study investigates whether LiO SAM's enhanced features justify its higher computational demand, particularly in terms of real-time processing efficiency, compared to the more streamlined LeGO-LOAM approach. The whole code to re-implement can be found at: Gitlab Repo Link

1. Introduction

Real time pose estimation and mapping are incredibly important in robot navigation, especially with autonomous robots. Leveraging the ability to observe, analyze, and make decisions based on their surroundings would optimize robot missions and encourage seamless integration of the robot into the rest of the world.

For our final project, we decided to explore methods of pose estimation and mapping known as LeGO-LOAM (lightweight and ground-optimized lidar odometry and mapping) [3] and LiO SAM (lidar inertial odometry via smoothing and mapping) [4]. Our objectives were to research and understand the methodology, and experiment with running the algorithms on provided data sets from Northeastern's autonomous car, NUance.

Our background about these algorithms revealed some important comparisons between them. As an overview, LOAM (lidar odometry and mapping) is the originally created algorithm that uses point cloud inputs and no loop closure but provides a pretty good pose estimation. Jumping off of that framework, LeGO LOAM improves LOAM by implementing LiDAR scans, focusing on ground feature extractions to orient position, and assumes a point cloud distortion due to sensor movement and therefore employs a loop closure. LIO SAM takes it one step further, improving on LeGO LOAM by coupling odometry and smoothing, implementing IMU and GPS, and keeping loop closure in use. The drawback of this more expanded approach is that it is computationally more intense, resulting in a lag in real time processing. We wanted to compare these two methods to see if the lag in real time processing was a breaking point for LIO SAM or if one was more optimal.

The 2 datasets used are the "car_IR_RGB_lidar" and "KRI_lidar_gps_imu". These datasets are one of the largest datasets from Northeastern's autonomous car (NUANCE) driven manually along the streets of Newbury Street in Boston.

The car dataset has stereo RGB cameras looking forward, IR cameras looking forward, 2 Velodyne VLP-16 lidar mounted on top of the car, IMU and GPS. The main focus of this dataset was to collect camera data with at least one loop closure. The sensors like lidars, gps, imu in combination serve as a ground truth for visual slam algorithms.

1.1. Model Architecture

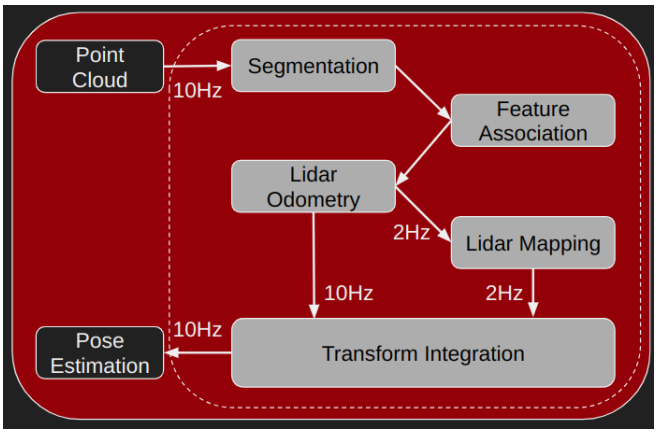


Figure 1. LeGO LOAM Architecture

The model architecture of LeGO-LOAM and LiO SAM involves sophisticated use of LiDAR and Inertial Measurement Unit (IMU) data for Simultaneous Localization and Mapping (SLAM). These algorithms are pivotal in mobile robotics for 3D mapping and autonomous navigation.

LeGO-LOAM and LIO SAM are designed to efficiently process and integrate the data from LiDAR and IMU sensors. The algorithms differ in their approach to preprocess-

ing and scan matching, which significantly affects their outcomes in final reconstruction.

- **LiDAR Data Processing:** LiDAR-based SLAM algorithms focus on acquiring subsequent scans, which are then properly registered to generate a point cloud of the environment.
- **IMU Integration:** The IMU data are used alongside the LiDAR data for several purposes, including correcting potential point cloud distortions due to sensor motion, limiting errors in point cloud registration during rapid sensor movements, estimating the robot's trajectory, and aiding in convergence of algorithms involved in assembling the acquired pointclouds.
- **Scan-to-Scan and Scan-to-Map Registrations:** These algorithms estimate the robot's path in two main steps. Initially, a scan-to-scan alignment is performed between adjacent LiDAR scans to recover an immediate motion guess. Subsequently, a scan-to-map registration is executed between the current scan and the environment's point cloud map to increase global pose consistency.
- **Graph-Based Formulation in Smoothing Methods:** Smoothing LiDAR SLAM algorithms, like LiO SAM and LeGO LOAM, minimize path and map errors using a graph-based formulation with a package called GTSAM [1]. This approach constructs a graph whose nodes represent robot poses or map portions, and edges embed constraints between nodes.
- **Preprocessing and Downsampling:** To reduce computational effort, these algorithms preprocess the LiDAR scans, which may include downsampling and feature extraction. A voxel grid approach is often used for downsampling, where all points in a voxel are approximated by their centroid.
- **Tight vs. Loose Sensor Fusion:** The fusion of LiDAR and IMU data can be either tight or loose. Tight sensor fusion exploits both data sources simultaneously for a unified output like in the case of LioSAM, while loose sensor fusion uses them independently but in combination to compensate for each module's shortcomings like in the case with LeGO-LOAM.

Both the LOAM systems in their codebase [2] utilize various C++ files, each serving a specific purpose in the Lidar Odometry and Mapping process. Key files include:

1. **ImageProjection.cpp:** Manages the segmentation of the point cloud and prepares it for feature association.
2. **FeatureAssociation.cpp:** Responsible for associating features within the segmented point cloud.
3. **MapOptimization.cpp:** Handles the optimization of the map using the features and segmented point cloud.
4. **TransformFusion.cpp:** Fuses transformations calculated from different stages of the process.
5. **Utility.h:** Defines various utility functions and constants used across the system.

Each of these files plays a crucial role in ensuring the efficient processing of LiDAR data for real-time pose estimation and mapping on variable terrain.

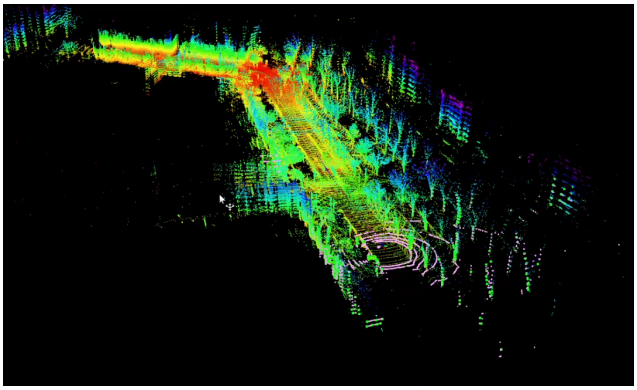
2. Experimental Work and Issues Faced

During the implementation of LeGO-LOAM, several challenges surfaced, necessitating effective troubleshooting. Initially, an issue emerged due to the discontinuation of the C-API in OpenCV 4, prompting the need to upgrade the OpenCV package from version 3.4.0 to 4.2.0. Another obstacle was a conflict between index types in the PCL and OpenCV FLANN module, which was successfully addressed by upgrading the GTSAM library to its latest version. Additionally, while compiling the code, we encountered a situation where C++ was not supported by certain modules. This was resolved by setting the CXX variable standard to 14. Lastly, a topic name change became imperative as we encountered difficulties viewing point cloud data on RVIZ. Through systematic troubleshooting, these challenges were effectively mitigated, ensuring a smoother implementation of LeGO-LOAM.

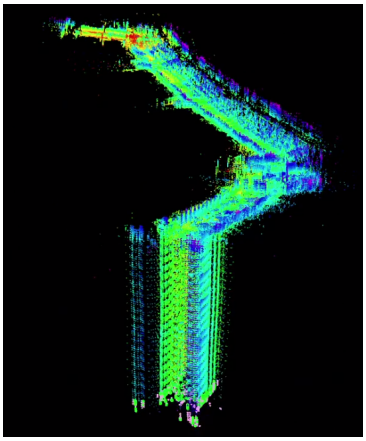
To generate a map depicting the trajectory of the autonomous car, we initiated the process by installing all additional dependencies essential for LeGO-LOAM and modifying all the .cpp files with the correct TF of the /camera topic and /map topic. Following the successful installation of these dependencies, we proceeded to install the Velodyne Point Cloud package. Subsequently, we made crucial parameter adjustments in both the launch file and utility configuration file. These files are instrumental in specifying configurations related to the lidar, odometry, and mapping parameters. This meticulous setup ensures that the autonomous car's trajectory is accurately captured and visualized during the mapping process.

3. Analysis and Discussion

At the time of our presentation on Nov. 28th, we had the developer's code running and executed Feature Association and Image Projection functionalities. We detected features like corner points and surface points from the point cloud dataset provided. This step often involves processing the raw point cloud data using algorithms like segmentation, clustering, or feature extraction. We displayed these features using ROS Visualization, a user interface that configures different types of data.



(a) LeGO LOAM



(b) LIO SAM

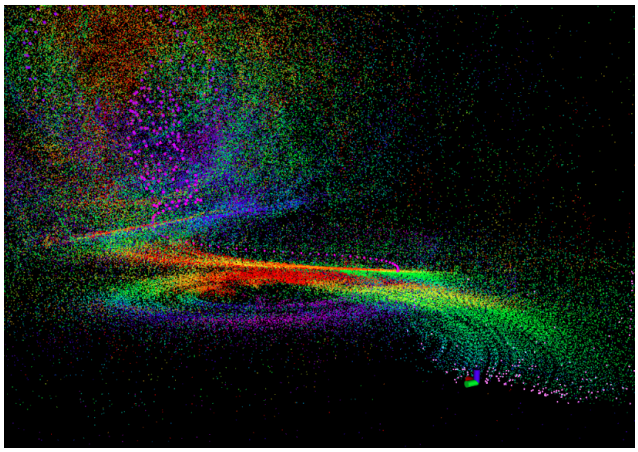
Figure 2. Mapping Status on CAR dataset with LeGO LOAM and LIO SAM.

For the KRI Dataset: While we had successfully generated point clouds for the KRI NUance dataset on both LeGO LOAM and LIO SAM algorithms, the GTSam mapping algorithm used with LeGO LOAM gave us some problems, making our map look like the car was falling in a helical pattern. The dataset being tested used Ouster lidars instead of Velodyne, which are not fully supported yet with LeGO LOAM, possibly causing the helical path. Also, the transformations between different frames were not correct making the robot seem like rotating in one place. The LIO SAM also failed to map properly, drifting by a large error due to IMU integration for Loop Closure. LeGO LOAM was a more suitable choice for this dataset.

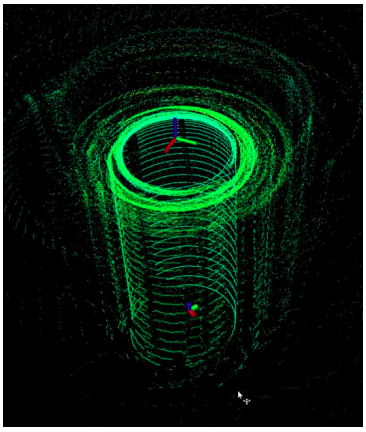
For the Car Dataset: Both LeGO LOAM and LIO SAM algorithms were working accurately initially, but after a while, the LIO SAM failed to map properly, drifting the map by a large error due to IMU integration for Loop Closure. LeGO LOAM was a more suitable choice for this dataset.

Although LIO SAM uses IMU for better map generation, in our testing with the Car dataset, the tight coupling of IMU data in LIO SAM makes the map drift and further results in the overall drifting of robot over time. This happens because IMU Preintegration is not calibrated and IMU data is not synchronized well resulting in a very poor map generation as shown in Figure 2. This also happens when there are problems with pointcloud deskewing while mapping. We can incorporate strategies such as global optimization (using GTSAM), and sensor fusion techniques to mitigate the effects of IMU drift in LIO SAM.

In the contrary, LeGO-LOAM relies more heavily on LiDAR data and is optimized for ground-based environments. Since it does not depend as much on IMU data for its core operations, it might be less prone to the kind of drift associated with IMU sensorsWe can incorporate strategies such as global optimization (using GTSAM), and sensor fusion techniques to mitigate the effects of IMU drift in LIO SAM. However, this also means it might be less robust in environments where LiDAR data alone is insufficient or in scenarios involving rapid or complex motions.



(a) LeGO LOAM



(b) LIO SAM

Figure 3. Mapping Status on KRI dataset with LeGO LOAM and LIO SAM.

Table 1. Comparison of LOAM algorithms with Car and KRI datasets

LOAM	KRI_lidar_gps_imu dataset	car_IR_RGB_lidar Dataset
LeGO LOAM	20ms	45ms
LIO SAM	300ms	500ms

4. Results

Table 1 shows the computation time for LIO SAM and LeGO LOAM for both Car and KRI datasets provided to us. We can observe that the computation time of LIO SAM for both the datasets is more in comparison to LeGO LOAM. We also observed from Figure 3 that LeGO LOAM performed better in Lidar Mapping as it relies heavily on point-cloud matching instead of IMU preintegration.

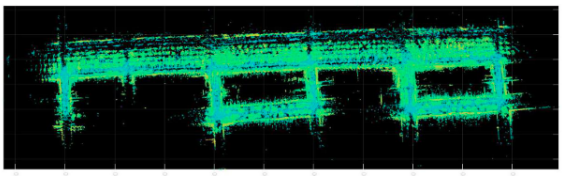


Figure 4. Final Map with LeGO LOAM and Car Dataset

5. Conclusion and Future Work

In this report, both LeGO-LOAM and LIO-SAM were methodically evaluated and implemented for two distinct datasets: NUANCE CAR Dataset and KRI Dataset. This process involved a thorough examination of each system's capabilities in processing and mapping the datasets.

During the analysis, it was observed that both LeGO LOAM and LIO-SAM faced their own challenges in generating accurate maps and achieving loop closures for both the CAR and KRI datasets. In the case of the CAR dataset, the map generated by LIO-SAM exhibited inaccuracies, with noticeable deviations from the actual environment. Loop closures, crucial for maintaining map consistency, were not effectively detected, leading to an accumulation of errors in the map. Whereas the LeGO LOAM achieved loop closure as seen in Figure 4. Similarly, for the KRI dataset, both LIO-SAM and LeGO LOAM produced a map with a helical path that did not correspond to the true path trajectory as shown in Figure **??**. IMU pre-integration issues may have contributed to these discrepancies, highlighting the need for precise sensor calibration and algorithm tuning.

This study underscores the necessity of choosing the right SLAM system based on the specific requirements and constraints of the application environment.

In future scope, limitations identified with LIO-SAM's map accuracy and loop closure detection can be mitigated using a camera feed along with LIDAR for loop closure detections. This can also be achieved through further refinement of IMU integration and calibration procedures. Additionally, exploring alternative SLAM algorithms like ORB-SLAM3 and 3D Graph SLAM can be done for better map generation. Continuous testing and validation on diverse datasets will be crucial for advancing the state-of-the-art in robot navigation and mapping.

References

[1] Factor graphs and gtsam — gtsam. https://gtsam.org/tutorials/intro.html. 2

[2] Robustfieldautonomylab/lego-loam: Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. https://github.com/RobustFieldAutonomyLab/LeGO-LOAM. 2

[3] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018. 1

[4] Tixiao Shan, Brendan J. Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. LIO-SAM: tightly-coupled lidar inertial odometry via smoothing and mapping. *CoRR*, abs/2007.00258, 2020. 1