Evaluation of Visual, LiDAR, and Inertial SLAM Algorithms

Danny Chen*, Junjiang Ye†, Shivam Patel‡, and Kevin Buca§,
University of Michigan
Ann Arbor, MI USA
Email: *dannych@umich.edu, †junjiang@umich.edu, ‡shivamgp@umich.edu, §kbuca@umich.edu

***Abstract*—Sensor fusion is essential for autonomous systems as it serves to collect and combine accurate environment information to make correct decisions. We evaluate tightly-coupled LiDAR inertial odometry via smoothing and mapping (LIO-SAM) [7] and tightly-coupled LiDAR-visual-inertial odometry via smoothing and mapping (LVI-SAM) [8] from different aspects. We use multiple datasets to verify the robustness of each algorithm. Moreover, we compare the runtime and accuracy to evaluate performance. We also simulate the improved algorithms in a self-defined maze on Gazebo to challenge the stability of both LIO-SAM and LVI-SAM.**

I. Introduction

Many of the technological products that are integrated into our lives have seen an increase in both their number of sensors and types of sensors. One prime example of this is autonomous vehicles, which in some cases, may include RGB-D cameras, LiDAR sensors, and an IMU. With the addition of new sensors, fusing the information provided by these sensors is a problem that is constantly being evaluated as autonomous vehicles need to simultaneously perform iterative mapping and localization in real-time. However, as the use of numerous sensors has become an industry norm, certain companies have made a strong claims AGAINST incorporating certain sensors (such as Tesla's stance against using LiDAR sensors on their vehicles).

With this discrepancy in mind, the problems that we will investigate are the following: What is the difference in batch SLAM accuracy between various sensor-fusion SLAM algorithms? What is the difference in execution time for real-time performance between various sensor-fusion SLAM algorithms? We will also test the algorithms with different datasets and also simulate the algorithms within a self-built environment.

This paper is organized as follows: Related work is covered in Section II. The Methodology is described in Section III, where we cover LIO-SAM and LVI-SAM. Section IV comprises Experiments and Results. Finally, Sections V and VI conclude the paper with conclusions and suggestions for future directions.

Our main contributions can be summarized as follows:

- Successfully implement LiDAR-Inertial SLAM such as LIO-SAM, and Visual-LiDAR-Inertial SLAM such as LVI-SAM on a KITTI Dataset and Handheld Dataset
- Simulate LIO-SAM, and LVI-SAM on Gazebo within a self-built environment
- Quantify and compare the performance of each algorithm in terms of SLAM accuracy and execution time for batch implementations.

II. Related Work

Typically, LiDAR is used in combination with other IMU and GPS sensors for state estimation and mapping. Using sensor fusion, such a design approach can commonly be divided into two categories: loosely-coupled fusion and tightly-coupled fusion. IMU in LOAM [10] is introduced to create a motion prior for scan-matching and de-skew the LiDAR scan. The IMU, however, is not engaged during the algorithm's optimization step. Therefore, LOAM is seen as a loosely-coupled method. Another ground vehicle based algorithm, the ground-optimized LiDAR odometry and mapping (LeGO LOAM) method is suggested in [6], which is designed for ground vehicle mapping problems. Its IMU measurements are also fused in the same way as LOAM.

Current research is heavily focused on tightly-coupled systems because they often provide more accuracy. In [5], a robocentric LiDAR-inertial state estimator, R-LINS, is described. A robot's state estimation is corrected by R-LINS in a tightly-coupled recursive way using an error-state Kalman Filter. It experiences drift during long-distance navigation as a result of the lack of additional sensors that are accessible for state estimates. In [9], a closely connected LiDAR inertial odometry and mapping framework, known as LIOM, is presented. When compared to LOAM, LIOM, which stands for LIO-mapping, achieves equivalent or greater accuracy by jointly optimizing measurements from LiDAR and IMU. However, real-time performance cannot be attained since LIOM is built to process all sensor measurements.

The robustness of LiDAR-visual-inertial systems in sensor-degraded tasks has recently gained more attention. LiDAR-based methods can capture the fine details of the environment at long range but they frequently fail when executed in structure-less areas, such as lengthy hallway or a wide open field. On the other hand, the performance of vision-based approaches is sensitive to variation in illumination, rapid motion, and initialization, despite the fact that they are particularly well suited for place recognition and excel in situations with a lot of texture. To boost their respective resilience and accuracy, LiDAR-based and vision-based approaches such as LIC [11]

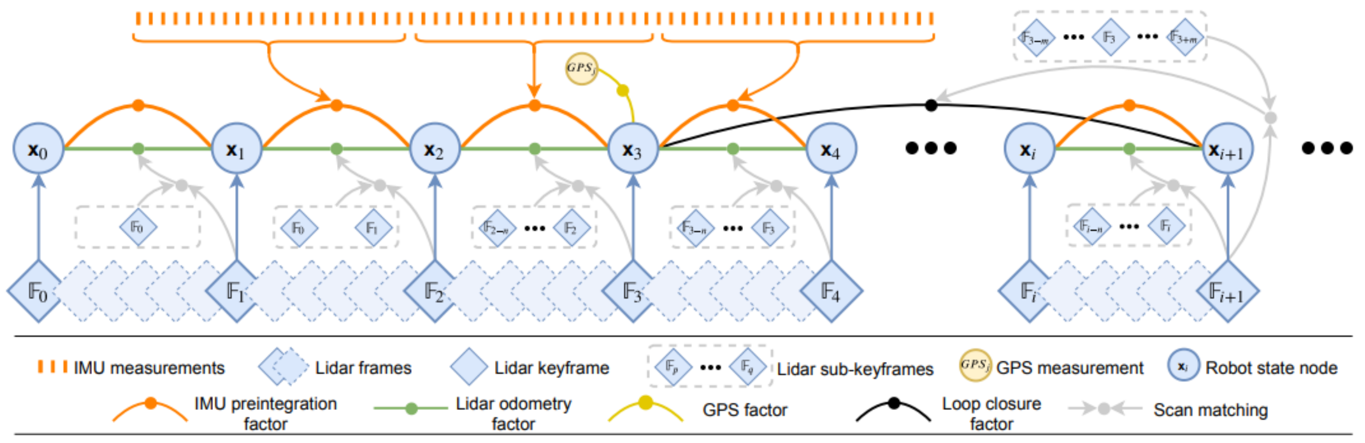


Fig. 1. The system structure of LIO-SAM. The system receives input from a 3D LiDAR, an IMU and optionally a GPS. Four types of factors are introduced to construct the factor graph.: (a) IMU preintegration factor, (b) LiDAR odometry factor, (c) GPS factor, and (d) loop closure factor.

or LVIO [4] Fusion are frequently combined with an inertial measurement unit (IMU).

III. METHODOLOGY

A. LIDAR INERTIAL ODOMETRY VIA SMOOTHING AND MAPPING

We use the measurements of angular velocity and acceleration from the IMU to infer the relative body motion between two timesteps of the robot. This leads to the first constraint for the factor graph - IMU preintegration factors.

For the LiDAR odometry factors, we extracted edge and planar features from a LiDAR scan and adopt the concept of keyframe selection by setting threshold to achieve real-time nonlinear optimization. The generation of a LiDAR odometry factor can be separated into three steps:

- Sub-keyframes for voxel map: We implement a point cloud map by using fixed number of recent LiDAR scans with sliding window and extract the n most recent keyframes, which we call the sub-keyframes, to optimize estimation.
- Scan-matching: Match a newly obtained LiDAR frame with computational efficiency and robustness in various challenging environments.
- Relative transformation: GaussNewton method is used to optimize the relative transformation by minimizing the distance between a feature and its edge or planar patch correspondence.

We also introduce GPS, which offer absolute measurements to eliminate drift in real-world navigation. But we only add GPS factors constantly when the estimated position covariance is larger than the received GPS position covariance because the drift of LiDAR inertial odometry grows very slowly

Finally, we include Euclidean distance-based loop closure detection to generate a point cloud descriptor and use it for place recognition. This constraint factor can correct the drift in a robot's altitude, when GPS is the only absolute sensor available.

B. LIDAR VISUAL INERTIAL ODOMETRY VIA SMOOTHING AND MAPPING

An overview of LVI-SAM system, which receives inputs from a 3D LiDAR, a monocular camera, and an IMU, is shown in Figure 2. Our framework consists of two main subsystems: a visual-inertial system (VIS) and a LiDAR-inertial system (LIS). The VIS handles image and IMU data processing and can include LiDAR measurements if available. Visual odometry is achieved by minimizing the joint residuals of visual and IMU data. The LIS extracts LiDAR features and uses them to perform LiDAR odometry by matching the features with a feature map, which is maintained in real-time using a sliding-window approach.

1) LIS: To solve the state estimation problem, which can be expressed as a maximum a posteriori (MAP) problem, the IMU preintegration constraints, visual odometry constraints, LiDAR odometry constraints, and loop closure constraints we have seen in LIO-SAM, are jointly optimized in a factor graph using iSAM2. It's worth noting that the multi-sensor graph optimization approach utilized in the LIS aims to enhance system efficiency by reducing data exchange. We also added two new methods that increase system robustness:

- Initial guess: The source of initial guess is different before and after LIS initialization. Before LIS initialization, the integrated translational and rotational change between two LiDAR keyframes can produce the initial guess for scan matching even in challenging conditions. Then we send them to the VIS system to improve its initialization; After the LIS initialization, we can get the guesses either from integrated corrected IMU measurements or the VIS to make sure the robustness in any kind of environment.
- Failure detection: The LIS reports failure when the smallest eigenvalue of ATA is smaller than a threshold at the first iteration of optimization.

2) VIS: The visual features are detected using a corner detector and tracked by the Kanade–Lucas–Tomasi algorithm [8]. In VIS initialization, we register LiDAR frames using

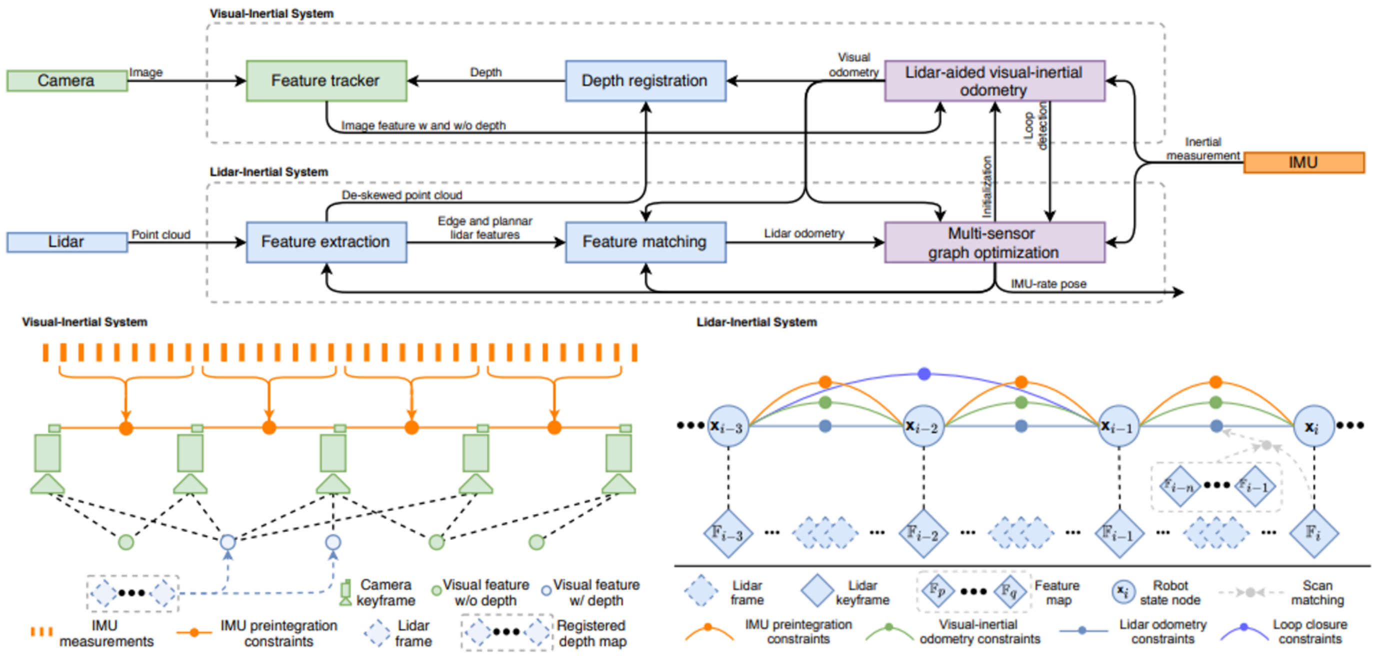
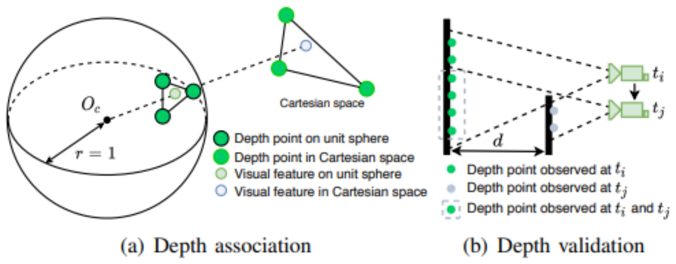


Fig. 2. The system structure of LVI-SAM. The system, which receives input from a 3D LiDAR, a camera and an IMU, can be divided into two sub-systems: a visual-inertial system (VIS) and a LiDAR-inertial system (LIS). The VIS and LIS can function independently while using information from each other to increase system accuracy and robustness. The system outputs pose estimates at the IMU rate

visual odometry and obtain a stacked sparse depth image for feature depth estimation.

- Initialization: VIO frequently fails to initialize when the sensor travels in trivial velocity since metric scale is not observable. Therefore, we use the LIS to obtain the estimated system state x and IMU bias b as an initial guess to implement the VIS initialization.
- Feature depth association: Since we only have sparse point cloud data from LiDAR scan, we stack multiple LiDAR frames to get a dense map. We will project feature and LiDAR depth points on a camera centered unit sphere using polar coordinates. By searching a 2-D K-D tree can find the nearest three points on the surface. In Figure 3, we can see the visualization that feature depth is the length of the dashed straight line. Moreover, we need to validate feature depth via checking distances. We need to reject the estimation once the distance is larger than 2 meters since depth points from different objects results in inaccurate estimation.
- Failure detection: Insufficient features may lead to optimization divergence while the camera receives less features in aggressive motion, illumination variation, and texture-less environments. In order to prevent this, we actively report failure so the function of LIS won't be interrupted. The VIS will inform LIS, initialize, and restart.
- Loop closure detection: We extract BRIEF descriptors [1] and use DBoW2 [2] for loop closure detection.



O_c
$r = 1$
Cartesian space
Depth point on unit sphere
Depth point in Cartesian space
Visual feature on unit sphere
Visual feature in Cartesian space
(a) Depth association
t_i
t_j
d
Depth point observed at t_i
Depth point observed at t_j
Depth point observed at t_i and t_j
(b) Depth validation

Fig. 3. Visual feature depth association.

IV. EXPERIMENTS AND RESULTS

Our code can all be found on our GitHub page:
https://github.com/shivamgp01/
Visual-LiDAR-and-Inertial-SLAM-Evaluation.

A README.md file is also provided instructing how to execute the code.

Additionally, all evaluation and plotting of SLAM algorithms was performed using the open source evo Python package [3].

A. *KITTI Dataset*

This dataset includes camera, LiDAR, and IMU data of a vehicle driving down the road. This is a very short dataset with a duration of only 40 seconds.

Our first experiment was to compare the performance of LIO-SAM and LVI-SAM on this dataset. As part of this evaluation, we tracked the predicted path of the vehicle from the algorithms as well as the execution time for one fusion of a

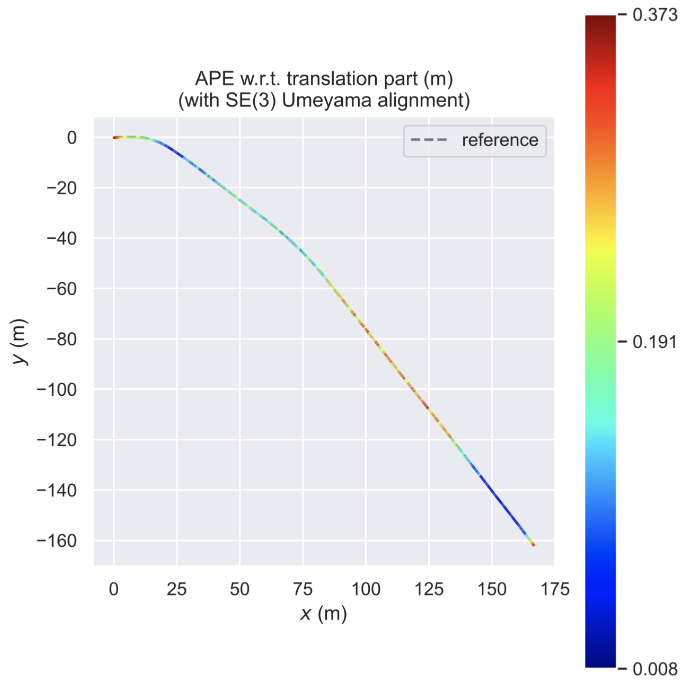


Fig. 4. Path differences between LIO-SAM and LVI-SAM on small KITTI Dataset

single datum from each algorithm. Ultimately, we discovered very little difference between the two algorithms on this dataset because of its very short duration and the simple trajectory of the vehicle. The similarity in paths is illustrated in Figure 4. Still, this was a good baseline for us to see that LIO-SAM was working properly on a dataset that had camera data, which is not a scenario that LIO-SAM had been previously tested on. This experiment was also significant because it proved that we could run both LIO-SAM and LVI-SAM on the same dataset, paving way for our following experiments.

B. Handheld Dataset

The Handheld Dataset is much different from the KITTI Dataset that we used in our first experiment. The Handheld Dataset is nearly 30 minutes long, and, as its name suggests, consists of an individual hand-carrying a robot with a camera, LiDAR sensor, and an IMU while the robot's sensors collect the data from the surroundings. Due to this, there is much more shaking and noise than the KITTI dataset, especially for the camera, so it is a good test of the algorithms' robustness. The trajectory of this dataset is also more complex, as it contains multiple loops and a longer distance.

Our first experiment on with this dataset was to record the reconstructed paths from both algorithms, one at a time, for 150 seconds. We did this at default rosbag playback speed. LIO-SAM performed very well in this experiment, as it reconstructed a very smooth path. LVI-SAM, on the other hand, had some notable differences. First, the reconstructed path was much less smooth than that from LIO-SAM. Most notably, LVI-SAM began to fail localization as the rosbag approached the 150 second mark. This raised concerns about the capability of LVI-SAM to achieve reliable real-time performance.

The next experiment that we performed on the Handheld Dataset was to reconstruct the entire path from the full dataset. The purpose of this experiment was to obtain a longer path for comparison between the algorithms. Ultimately, we were curious to see what impacts the extra data from the camera would have on the reconstructed path. Due to the localization concerns of LVI-SAM, we played back the Handheld dataset at 0.2x speed to ensure that the algorithm would have enough time to process and fuse the data from all three sensors to avoid localization failure. Both LIO-SAM and LVI-SAM were able to successfully reconstruct the entire path from the dataset at this playback speed. The resulting paths are displayed in Figure 5. From the Google Maps overlay, it is very clear that the LVI-SAM trajectory experienced less drift and matches the actual path more closely. The progression of the absolute pose error throughout the rosbag is displayed in Figure 6.

After this experiment clearly showed an advantage to LVI-SAM in terms of accuracy, we wanted to determine the cost of this improvement in terms of computation time. To do this, we determined the average sensor fusion rate for both LVI-SAM and LIO-SAM on both the small KITTI dataset and the Handheld dataset. We were able to do this and obtain comparable rates because both algorithms had the same limiting rates on both datasets. The limiting rate was the 10 Hz input rate from the LiDAR data. The average sensor fusion rates that we collected are displayed in Table 1, below.

TABLE I
AVERAGE SENSOR FUSION RATES FOR LIO-SAM AND LVI-SAM

	Handheld	KITTI
LIO-SAM	2.32 Hz	3.98 Hz
LVI-SAM	1.14 Hz	1.36 Hz

Clearly, the accuracy improvements of LVI-SAM do come at the cost of a slower sensor fusion rate.

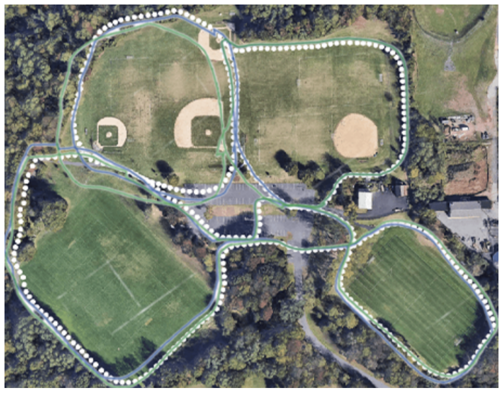


Fig. 5. LIO-SAM (green path) and LVI-SAM (blue path) trajectory reconstructions for the Handheld dataset. LVI-SAM experiences less drift.

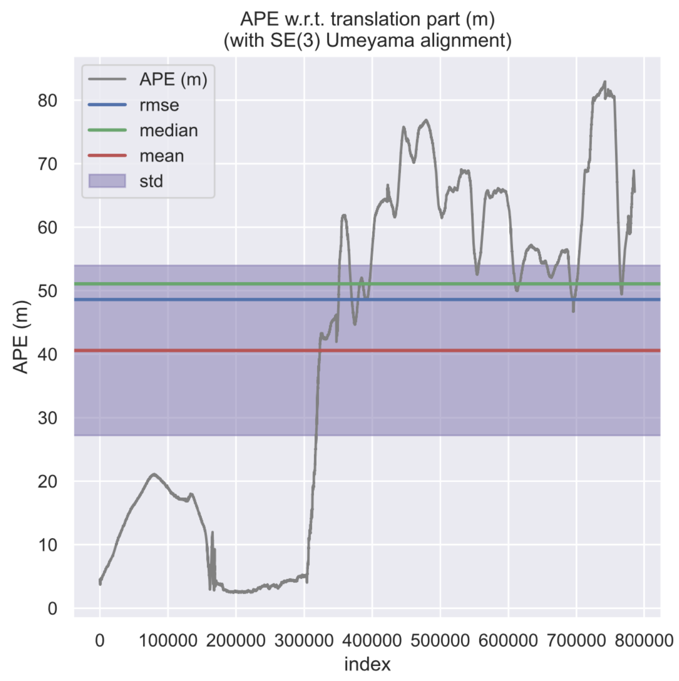


Fig. 6. APE plot between LIO-SAM (green path in Figure 5) and LVI-SAM (blue path in Figure 5) with RMSE, mean, median, and standard deviation. NOTE: due to the lack of ground truth data, we treated the LVI-SAM path as ”ground truth” because it matched the actual path more closely.

C. Gazebo

Following our experiments with the Handheld Dataset which raised concerns about the performance of LVI-SAM, we wanted to test both algorithms in a controlled simulation environment. We constructed an environment, shown in Figure 7. This robot contains a camera, LiDAR sensor, and IMU. It is a four-wheel robot and we can use the keyboard to control its movement.

In this experiment, we intended to test the SLAM capabilities of both frameworks and to compare their respective reconstructed paths to ground truth. The specifications of the simulation were as follows:

- System:
  Ubuntu 20.04
  ros-noetic
  gazebo11
- Robot Sensors:
  IMU
  Velodyne VLP-16 LiDAR
  RealSense d435 Camera

To test the two algorithms, we controlled the robot to do the same movement while running LIO-SAM and LVI-SAM, respectively, and recorded the optimized paths from them. We were able to achieve a stable run-time for both algorithms in this simulated environment for approximately 40 seconds, and we collected absolute pose error statistics for each algorithm during the simulation. These statistics are displayed below in Table 2.

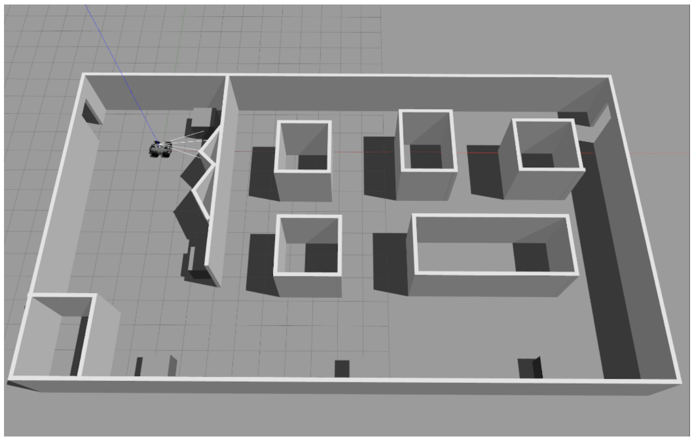


Fig. 7. Simulated environment in Gazebo used to test both LVI-SAM and LIO-SAM with pre-recorded path.

TABLE II
APE STATISTICS (IN METERS) FOR LIO-SAM AND LVI-SAM ON THE GAZEBO SIMULATION

	Max	Min	Median	Mean	STD
LIO-SAM	0.994	0.003	0.196	0.219	0.137
LVI-SAM	0.540	0.002	0.125	0.140	0.078

In this controlled, simulated environment LVI-SAM proved to follow the ground truth trajectory more closely than LIO-SAM. This is also supported by the ground truth trajectory comparison plot which we generated in Figure 8. The result of LIO-SAM tends to have some jittering. One possible reason is in LIO-SAM, we will use the voxel grid filter to compress the point cloud data, and in indoor environment, this process may cause some error while matching the point cloud, thus making the optimized pose shaking. Another reason may be in gazebo, the simulation of IMU may not be so accurate, and in addition, there will be significant cumulative errors in IMU itself, thus causing this problem.

V. DISCUSSION AND CONCLUSION

In this project, we were able to re-implement LIO-SAM and LVI-SAM using the vehicle based KITTI dataset and handheld park dataset. Using these datasets, we were able to compare the LIO-SAM and LVI-SAM algorithms for trajectory mapping accuracy and execution time. We also created a Gazebo simulation for both of these algorithms to create our own maps and trajectories to visualize the LIO-SAM and LVI-SAM algorithms using our own custom control inputs.

After completing this analysis of LIO-SAM and LVI-SAM for this project, we were able to qualitatively conclude that LVI-SAM provides a slight improvement over LIO-SAM for trajectory mapping accuracy. This is only a qualitative conclusion after evaluating the error between the handheld dataset results for LIO-SAM and LVI-SAM to the ground truth Google Maps result and is not a concrete conclusion. However,

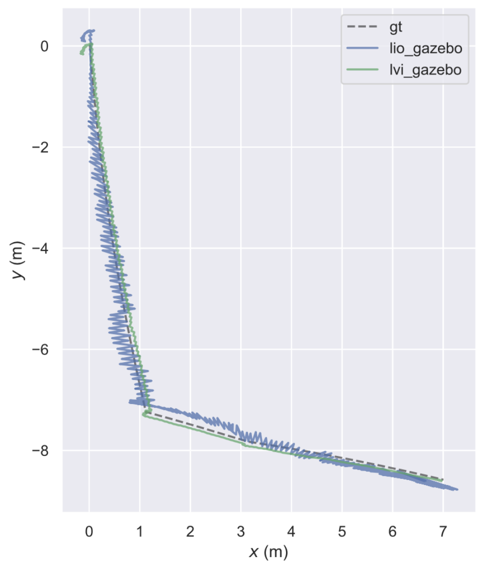


Fig. 8. Ground truth comparison between LVI-SAM and LIO-SAM on the Gazebo simulation. The green path LVI-SAM, proves to follow the ground truth path (dashed path) more closely.

our execution time analysis did allow us to quantitatively conclude that LVI-SAM faces an approximately 2.5 times increase in execution time per fusion of sensor data into an odometry position. As a result, we were able to conclude that LVI-SAM would require increased compute power for viable use on real-time systems. Without sufficient computing resources, we observed that both algorithms perform data sampling for sensor data. This sampling sometimes would result in missing crucial data required for the algorithms to operate without failure. Specifically, for LVI-SAM, we often faced issues with high velocities and losing camera features when rotating, leading to failure states for the algorithm. These failure states would render the algorithm dangerous for any real-world application if not properly handled and avoided.

For the Gazebo simulation, we were able to successfully simulate LVI-SAM in a controlled environment but struggled with LIO-SAM due to jittering in the output trajectory. We believe this was either due to the usage of the voxel grid filter to compress the point cloud data or issues with the IMU sensor simulation (which would have a much greater effect on LIO-SAM than LVI-SAM). Future evaluation of the Gazebo simulation would be needed to help alleviate these simulation issues.

The next steps for this project cover performing a more quantitative analysis of the trade-offs between LIO-SAM and LVI-SAM for accuracy and execution time to evaluate which applications would benefit from each algorithm. Additionally, future analysis would be done for real-time robustness evaluation with sensor corruption and noise testing allowing analysis for safety critical systems such as autonomous vehicles.

VI. FUTURE DIRECTIONS

The future directions and work yet to be completed for this project revolve around the tasks our team was unable to complete in the span of this project.

First, our team was unable to compare LIO-SAM and LVI-SAM output trajectories to ground truth GPS poses for the handheld and small KITTI datasets. This was primarily because we were unable to extract the GPS ground truth data for these datasets with conversion from latitude/longitude to x/y coordinates. Further exploration of these datasets and the tools available with each of these algorithms would most likely allow us to retrieve the ground truth data. Without ground truth data, we were unable to perform true accuracy evaluation with respect to the ground truth poses. We were able to perform a subjective analysis of the handheld dataset with a overhead ground truth plot but were unable to get access to the raw data for an objective analysis. This subjective analysis was able to confirm that LVI-SAM performed better than LIO-SAM for trajectory accuracy at a significant cost of higher execution time but an objective analysis would be able to quantify the increase in mapping performance and evaluate the trade-off between each algorithm.

The second portion of future work involves tuning the Gazebo simulation to avoid jittering with LIO-SAM. We would need to explore the raw simulated IMU data and the effects of the voxel grid filter for compression of the point cloud data to look into these issues and get further instructor support to identify simulation issues.

The third portion of future work that we were unable to complete was a real-time robustness evaluation on a realistic application-based computing system. This task was unable to be completed as we were limited to the resources we had available with our Ubuntu VirtualBox systems. These limited resources made it difficult for us to run LIO-SAM and especially LVI-SAM in real-time. In order to avoid data sampling leading to failed simulations during rosbag playback, we needed to reduce the rosbag playback when running these algorithms. Future work would involve running these algorithms on systems with compute power similar to systems that can be seen on specific applications (eg. autonomous vehicles). Our limited resources made it difficult to find LVI-SAM feasible for real-time operation even with the handheld walking dataset and would definitely not scale well to a vehicle traveling at higher speeds unless we were able to scale up our compute resources and perform testing. Additionally, we would like to perform sensor corruption and noise testing on a real-time system to see which algorithm is more robust to errors and instabilities in the input data. This analysis would provide a lot of insight on abilities of these algorithms to be incorporated into commercial products.

ACKNOWLEDGEMENT

We would like to thank Prof. Maani, the GSIs, the IAs, and the graders for a great learning experience in this course and all the assistance we received on this project.

References

[1] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. Brief: Binary robust independent elementary features. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11* (2010), Springer, pp. 778–792.

[2] Gálvez-López, D., and Tardos, J. D. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics 28*, 5 (2012), 1188–1197.

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

[4] Jia, Y., Luo, H., Zhao, F., Jiang, G., Li, Y., Yan, J., Jiang, Z., and Wang, Z. Lvio-fusion: A self-adaptive multi-sensor fusion slam framework using actor-critic method. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2021), IEEE, pp. 286–293.

[5] Qin, C., Ye, H., Pranata, C. E., Han, J., Zhang, S., and Liu, M. Lins: A lidar-inertial state estimator for robust and efficient navigation. In *2020 IEEE international conference on robotics and automation (ICRA)* (2020), IEEE, pp. 8899–8906.

[6] Shan, T., and Englot, B. 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), IEEE, pp. 4758–4765.

[7] Shan, T., Englot, B., Meyers, D., Wang, W., Ratti, C., and Rus, D. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (2020), IEEE, pp. 5135–5142.

[8] Shan, T., Englot, B., Ratti, C., and Rus, D. Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping. In *2021 IEEE international conference on robotics and automation (ICRA)* (2021), IEEE, pp. 5692–5698.

[9] Ye, H., Chen, Y., and Liu, M. Tightly coupled 3d lidar inertial odometry and mapping. In *2019 International Conference on Robotics and Automation (ICRA)* (2019), IEEE, pp. 3144–3150.

[10] Zhang, J., and Singh, S. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems* (2014), vol. 2, Berkeley, CA, pp. 1–9.

[11] Zuo, X., Geneva, P., Lee, W., Liu, Y., and Huang, G. Lic-fusion: Lidar-inertial-camera odometry. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2019), IEEE, pp. 5848–5854.