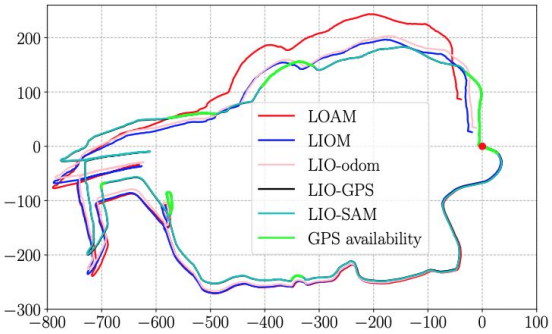
LIO-SAM/FAST-LIO and NUANCE

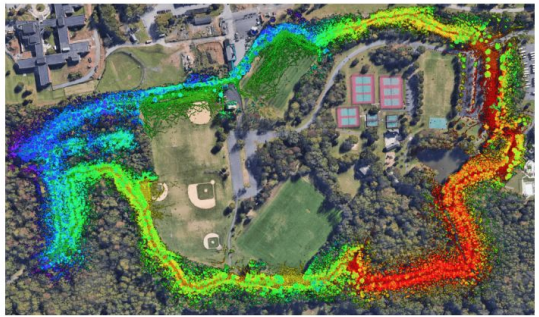
Robotics Sensing and Navigation
Fall 2024
Group 3:
Oliver Hugh
Jack Gladowsky
Zhang Ye
Sasha Oswald
Shail Jagat Mehta

LIO-SAM Overview

- Very accurate real-time SLAM algorithm
- Uses IMU for initial odometry estimate, with LiDAR used to correct bias
  - Can incorporate GPS/other absolute measurements for long-term drift correction
- Uses 4 factors to create a factor graph



(a) Trajectory comparison



(b) LIO-SAM map aligned with Google Earth

Factor Graph Inputs

A). IMU Pre-Integration Factor

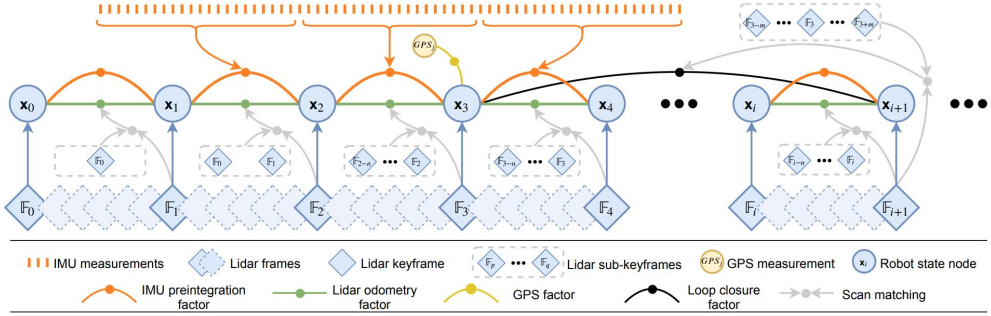
- Combines raw measurements with estimates of noise and moving bias
- Uses this with simple kinematics equations

B). Lidar Odometry Factor

- Feature Extraction
- Scan Matching
- Relative Transformation

C). GPS Factor

- Can use different sensors that provide absolute measurements
  - GPS, altimeter, compass
- Convert coordinates from global to local cartesian frame



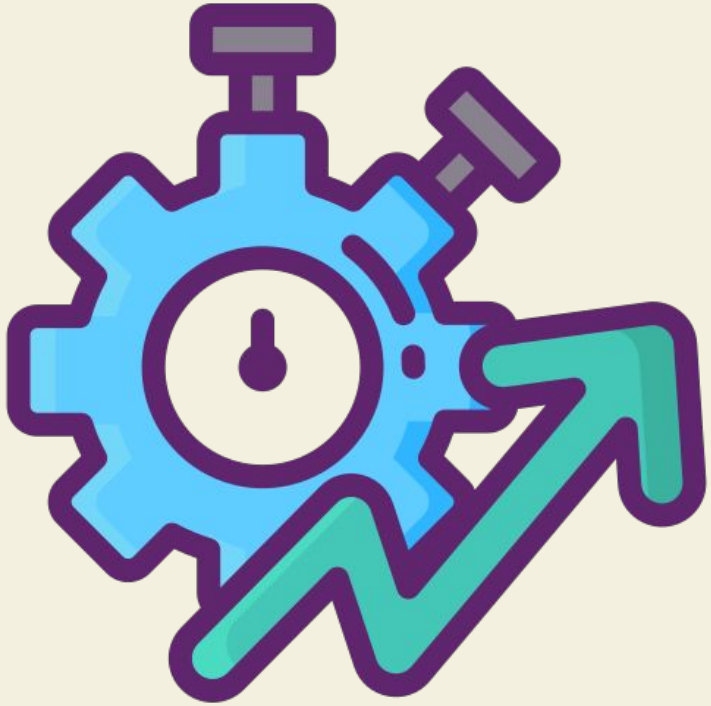
- Add node to factor graph and associate with GPS node

D). Loop Closure Factor

- Look through graph to find prior states that are close in Euclidean space to current frame
- Scan-matching to find match & obtain relative transformation to add to graph
- Different types of loop closure factor implementation could be added

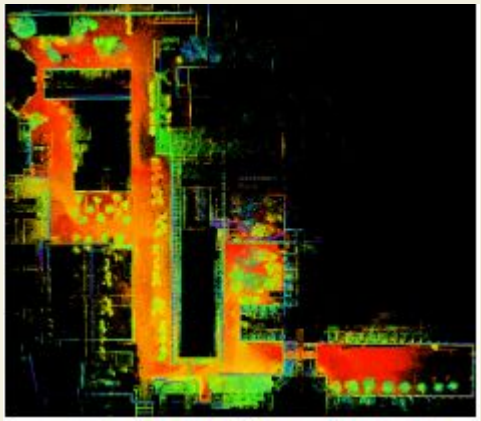
Efficiency

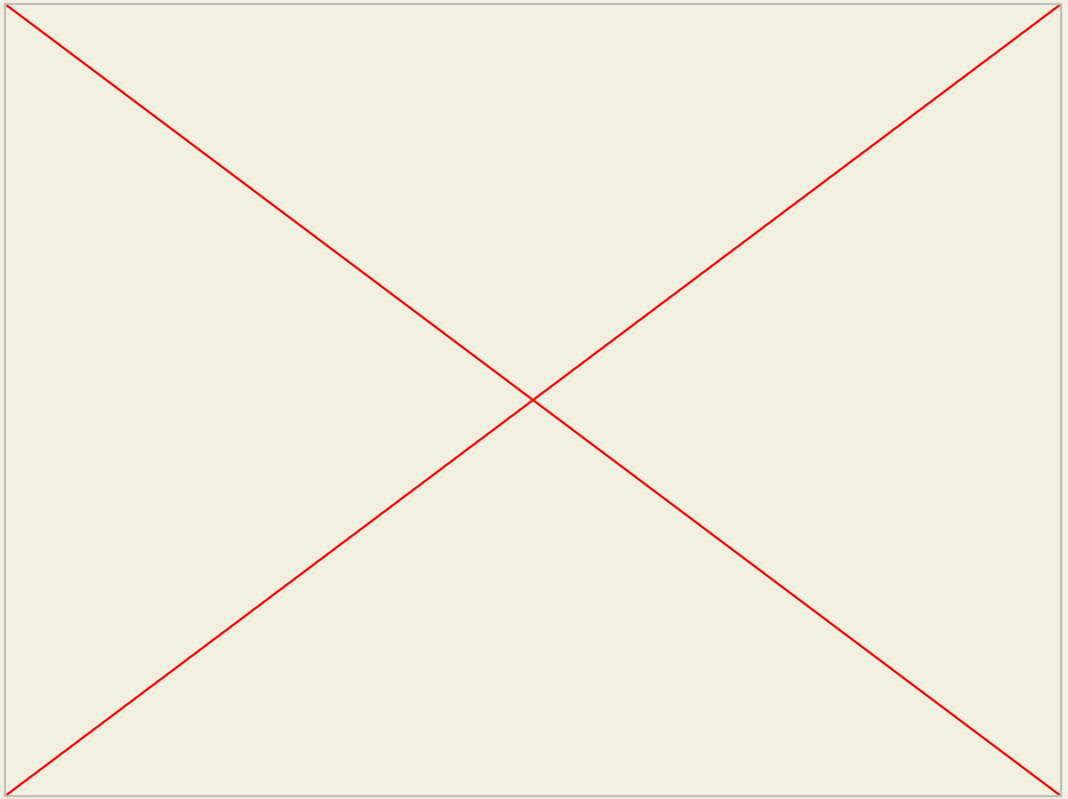
- LIO-SAM is more efficient than older Lidar SLAM algorithms
- Sliding Keyframe method for lidar scans
  - Computationally inefficient to add features from every single lidar scan
  - Select a lidar frame as the next keyframe when change passes threshold. Default values:
    - 1m position change default
    - 10° orientation change
- Limit of GPS factor association
  - Drift from just IMU and lidar only has drift in the long-term
  - Only add GPS factor when estimated position covariance > GPS position covariance
- Still room for improvement!



LIO-SAM: Provided Data

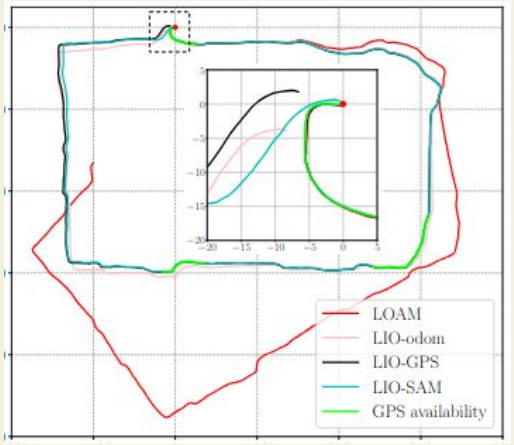
- Ran basic setup of LIO-SAM with ROS1 Kinetic on a dataset of walking around MIT
- This was the example dataset that works out-of-the-box with LIO-SAM

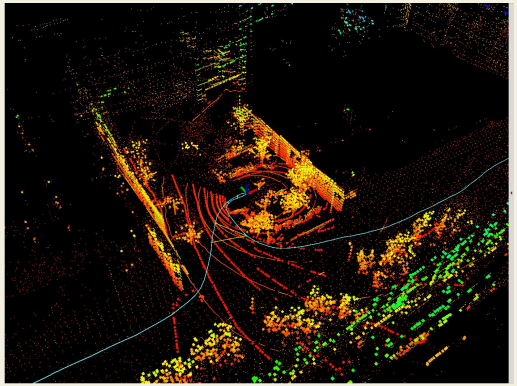




LIO-SAM: Provided Data

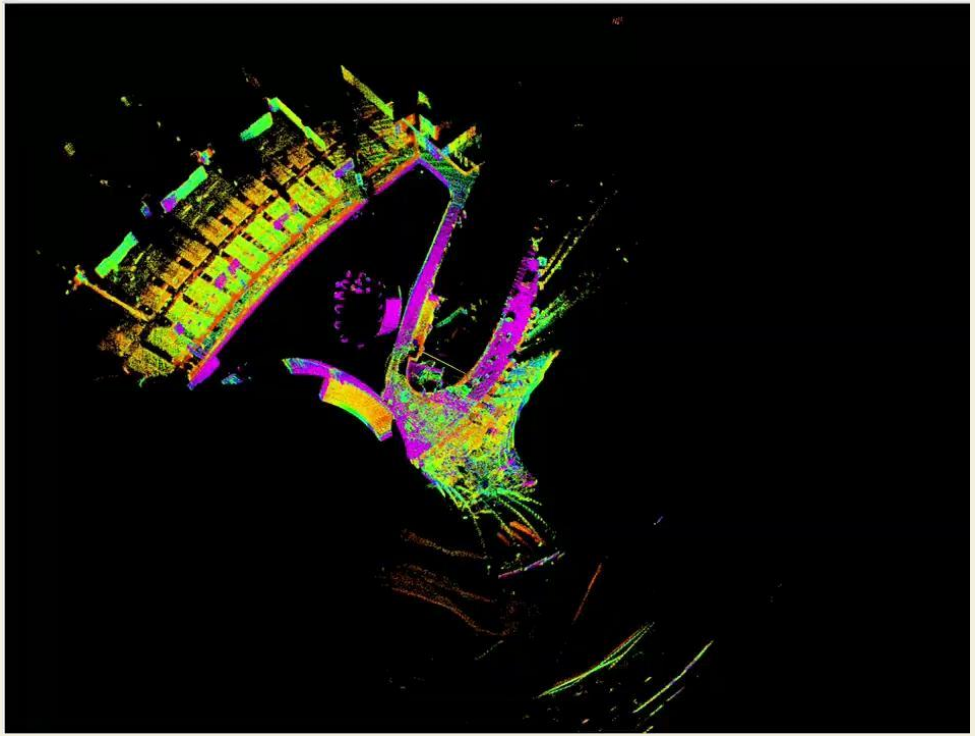
- LIO-SAM comes with ICP loop closure implemented
- ICP finds corresponding points from the current point cloud and the previously recorded point clouds
- Using the corresponding points, ICP calculates transformation between current point and map using least-squares





LIO-SAM: ISEC Data

- We then tested LIO-SAM with the ISEC data
- Dataset consists LIDAR/IMU data that travelled in a loop around ISEC first floor
- Shows how LIO-SAM handles indoor, complex environments



FAST-LIO Overview

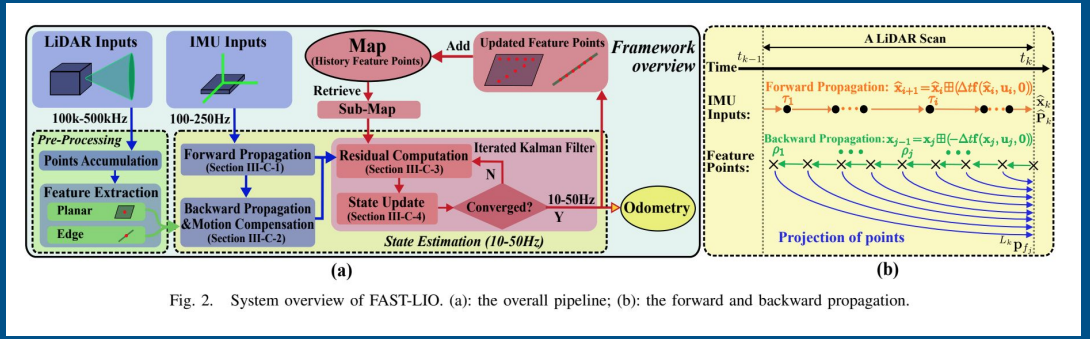


Fig. 2. System overview of FAST-LIO. (a): the overall pipeline; (b): the forward and backward propagation.

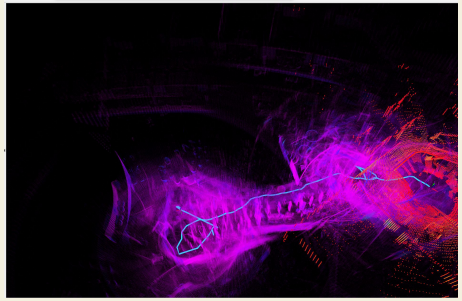
- More efficient, tightly coupled LiDAR-Inertial Odometry
- Can handle fast motion/deals with motion distortion and handles noisy/cluttered environments
- Uses iterated Extended Kalman Filter
- Introduces equivalent formula that greatly reduces complexity of Kalman gain computation
- Handles LiDAR points incrementally
- Forward propagation of IMU readings and back propagation of LiDAR features for bias correction
- Does not explicitly rely on loop closure
- Does not use factor graph – more efficient but less ‘modular’

LIO-SAM vs FAST-LIO

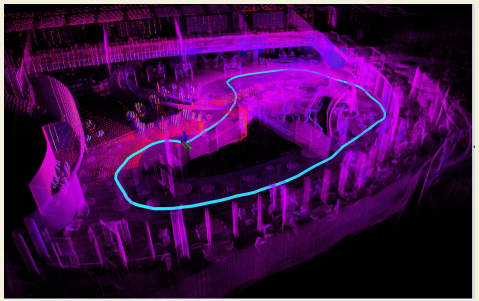
	LIO-SAM	FAST-LIO
Accuracy	- Highly accurate due to loop closure - Can fuse absolute measurements to correct long term drift	- Less accurate - Prioritizes local accuracy - Long term drift over extended paths
Real - Time Viability	- Poor real-time computational efficiency in areas with dense features	- Specially optimized for real time performance with limited resources
Computational Efficiency	- Has higher run time for larger number of features.(Proportional to the number of features)	- Has low run time for large number of features. (1.16 ms for 1802 nodes)

Running FAST-LIO

- Created a script to build a Docker container suitable for running FAST-LIO
    - Checks Ubuntu version and that ROS 1 is Melodic or higher
    - Uses livox_ros_driver and creates workspace, clones repository, and compiles
- Examine topics in the rosbags to identify required point cloud and IMU data topics
- Edit ouster64.yaml file in FASTLIO repository (since all our datasets were collected using Ouster sensors) to set default point cloud and IMU topics and enable path visualization
- Source environment and launch launch file and see RViz appear on the screen. Next, we open a new terminal, enter the Docker container, and play back the dataset.



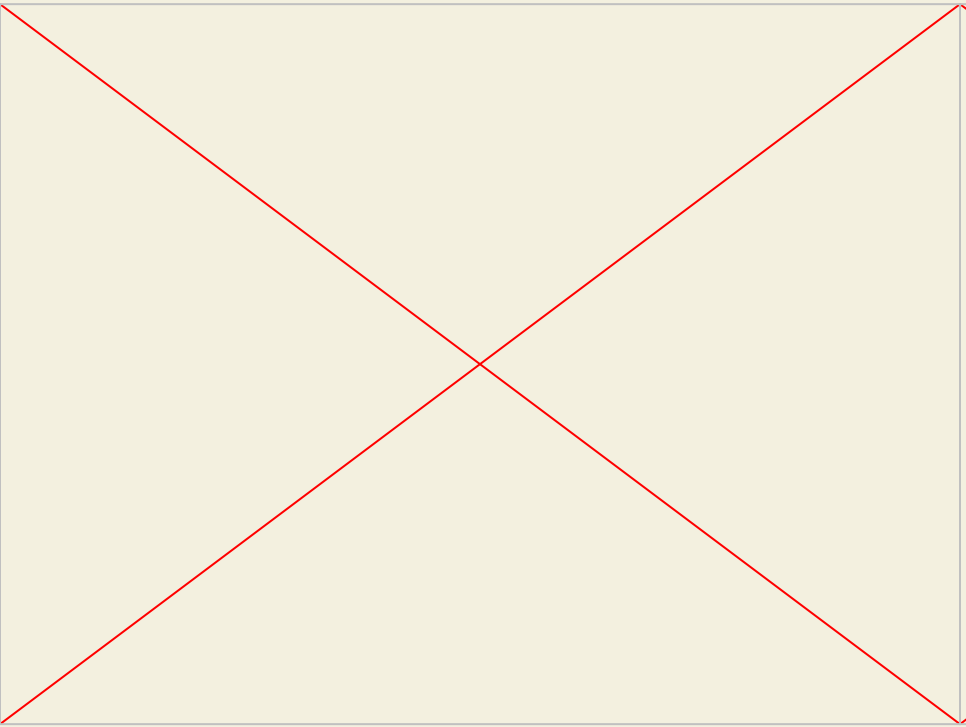
9-DOF IMU ISEC Data



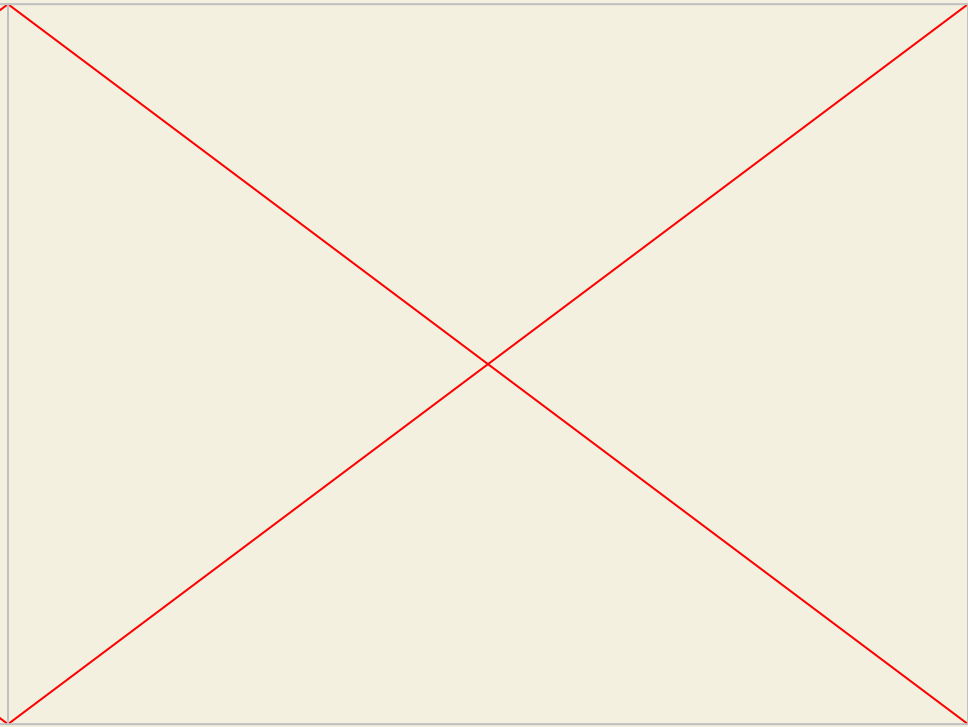
6-DOF IMU ISEC Data

- After running FASTLIO on ISEC datasets, we discovered time synchronization issues between Vectornav IMU 9-axis sensor and LiDAR
- This led to poor results in RViz as seen above on the left
    - Point clouds appear chaotic, highly disorganized
    - Trajectory has significant drift, is unstable and not smooth
- This highlights that time synchronization is critical for FASTLIO
- Time-synchronized 6-DOF data provides stable results (see above right)

FASTLIO: ISEC_1st_floor DATA



Using Vectornav 100 (9 DOF IMU)



Using Bosch BMI088 (6 DOF IMU in Ouster)

Our Data

- Recorded 2 datasets with professor in NUANCE
- Multiple instances of loop closure for both

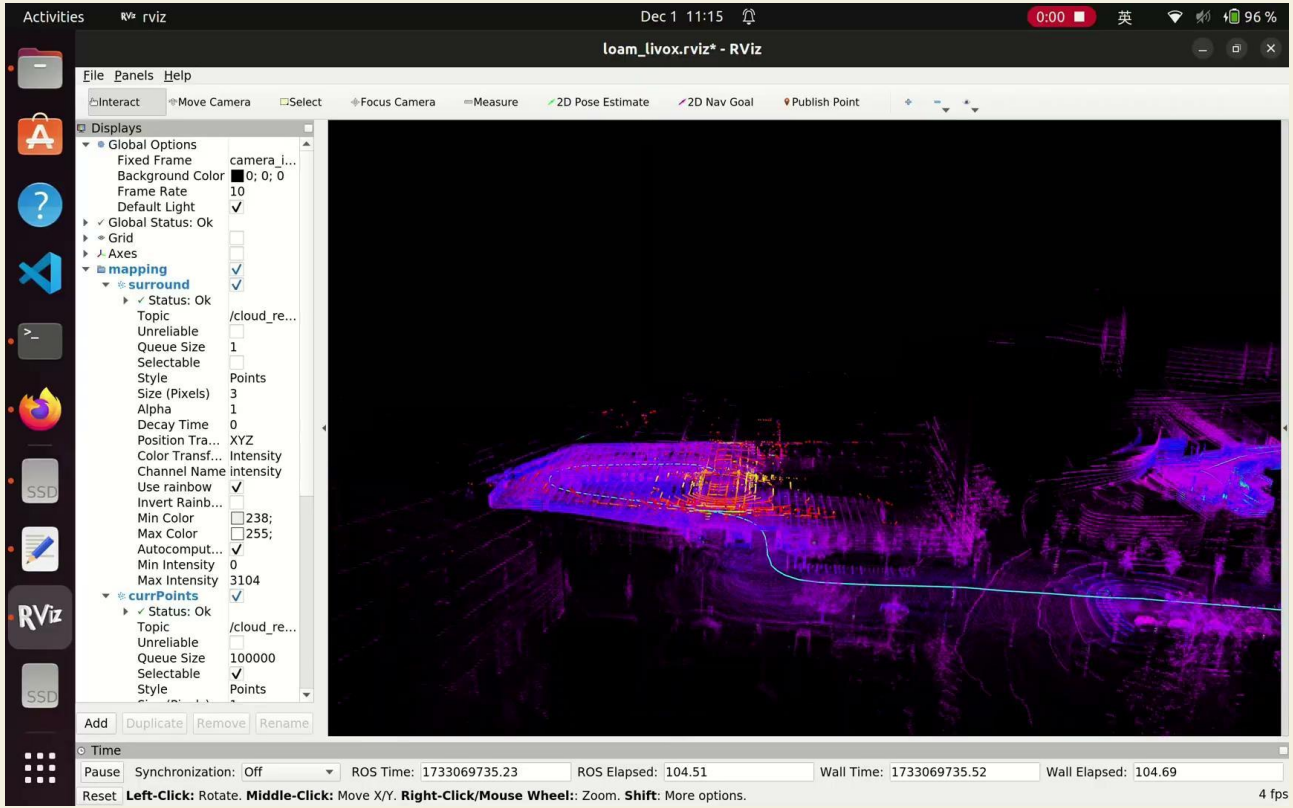
1. Driving Around Boston
   a. Resembles a typical drive, similar to lab 4
2. Driving up and down parking garage
   a. Testing highly variable altitude/3D problem

Issues with Dataset

- Initially tried to run NUANCE data using LIO-SAM
- LIO-SAM requires a 9-DOF IMU, which NUANCE has, but the 9-DOF IMU data is out of sync with the LIDAR data
- Researched alternatives and decided to also research FAST-LIO, as it could run on a 6-DOF IMU, which was synchronized with the LiDAR
- So, ran FAST-LIO on NUANCE data

FASTLIO: Nuance_garage Data

- **Sensor configuration: Ouster lidar + built-in IMU**
- **Point cloud frequency: 10Hz**
- **Scene type: Structured indoor environment: Spiral ramps and parallel parking spaces.**
- **Mapping effect: complete 3D point cloud map**



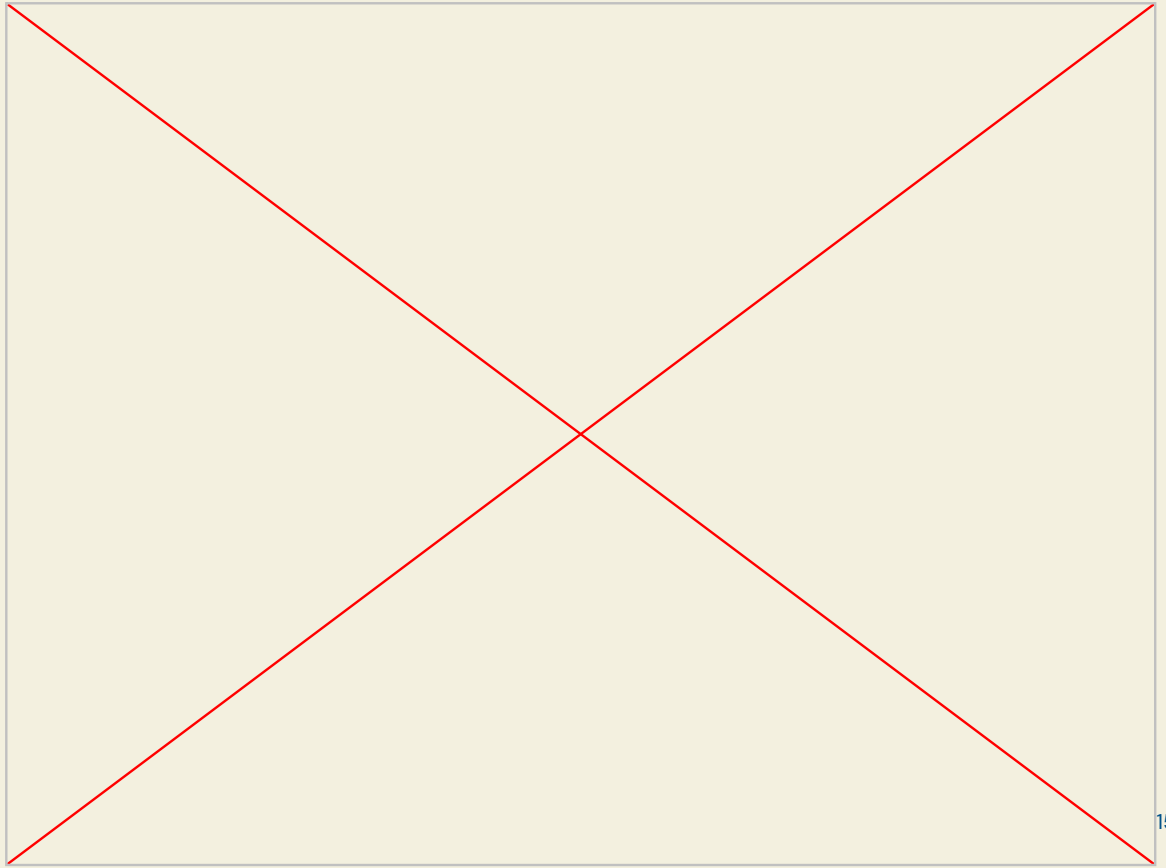
FASTLIO: Nuance_street Data

Environmental characteristics:

- Typical city street scene
- Includes roads and building facades
- There are urban infrastructure such as trees and telephone poles
- Semi-structured outdoor environment

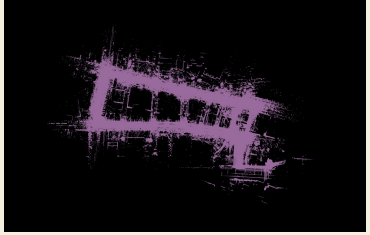
Point cloud quality:

- The outline of the building is clear and complete
- Road surface characteristics remain well maintained
- Vertical structures (such as walls) have sharp edges
- Point cloud density is uniform and coverage is complete

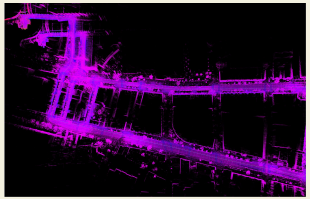


FASTLIO NUANCE Data Analysis

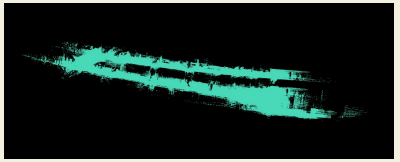
- **Driving Dataset**
- Overall clarity of the mapping point cloud for the driving dataset is relatively good, w/ complete building outlines, continuous road structures, and well-preserved local features like utility poles and sidewalks
- Noticeable drift - most significant issue is the lack of proper loop closure—i.e., the endpoint don't align with starting point.
  - Caused by cumulative odometry errors, with an estimated drift of approximately 10–15 meters.
- Additionally, by examining the side-view PCD scan map, we can see that the entire map does not lie on a single plane.
  - Indirectly indicates that the cumulative error in the Z-axis of the IMU caused the elevation of the map to continually rise, resulting in a "double-layered" map. These issues are primarily caused by the lack of loop closure detection and optimization, as well as the accumulation of IMU integration errors.
- **Garage Dataset**
- Same issue occurred with the parking garage dataset.
- Surrounding garage structures are clearly visible from vehicle's ascent, and the height of each floor is nearly consistent.
- During vehicle's descent on return path, trajectory exhibits vertical deviations, and the point cloud data shows ghosting artifacts.
  - Also caused by accumulation of IMU odometry errors and absence of a loop closure detection module



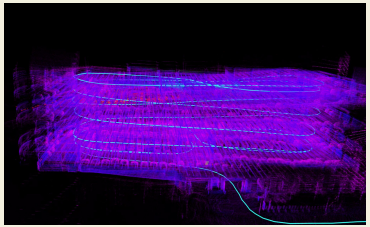
FAST-LIO with NUance_Street_Data: Point Cloud



FAST-LIO with NUance_Street_Data: Trajectory



FAST-LIO NUance_Street_Data Side View



FAST-LIO Garage Dataset Trajectory

Conclusions

- LIO-SAM
  - Good quality/accuracy maps and localization
  - Incorporates loop closure using ICP
  - Decent efficiency
- FAST-LIO
  - Good for faster moving robots, noisy data (UAVs, cars, etc)
  - Does NOT have any loop closure
- NUANCE Dataset
  - LIO-SAM needs 9-DOF IMU while FAST-LIO works with 6-DOF IMU
  - Due to 9-DOF IMU being external from LIDAR, the timestamps do not sync, resulting in LIO-SAM being incapable of running the NUANCE data

References

[1] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, “LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping,” 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2020, doi: https://doi.org/10.1109/iros45743.2020.9341176.

[2] T. Shan, “LIO-SAM,” GitHub, Nov. 08, 2022. https://github.com/TixiaoShan/LIO-SAM

[3] W. Xu and F. Zhang, “FAST-LIO: A Fast, Robust LiDAR-inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter,” IEEE Robotics and Automation Letters, pp. 1–1, 2021, doi: https://doi.org/10.1109/lra.2021.3064227.

[4] “hku-mars/FAST_LIO,” GitHub, Jan. 30, 2023. https://github.com/hku-mars/FAST_LIO