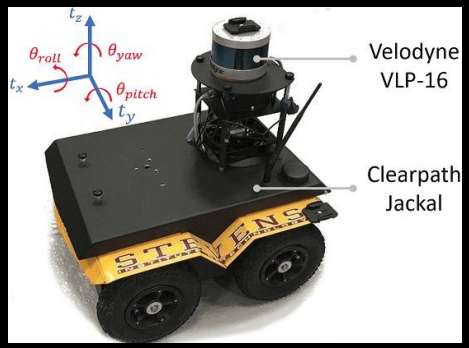
LeGO-LOAM

Sameer Bhatti, Riccardo Dalla,
Jack Bauer, Kevin Sinaga, Afridi Shaik

Project description

- Analyze conceptually pre-existing LeGO-LOAM algorithm and its implementation
- Test on Stevens Institute of Technology dataset and observe loop closure
- Use algorithm on NUANCE dataset to assess its robustness in terms of loop closure and trajectory estimation
- Make conclusions on algorithm performance and why it might not work as claimed on unknown datasets

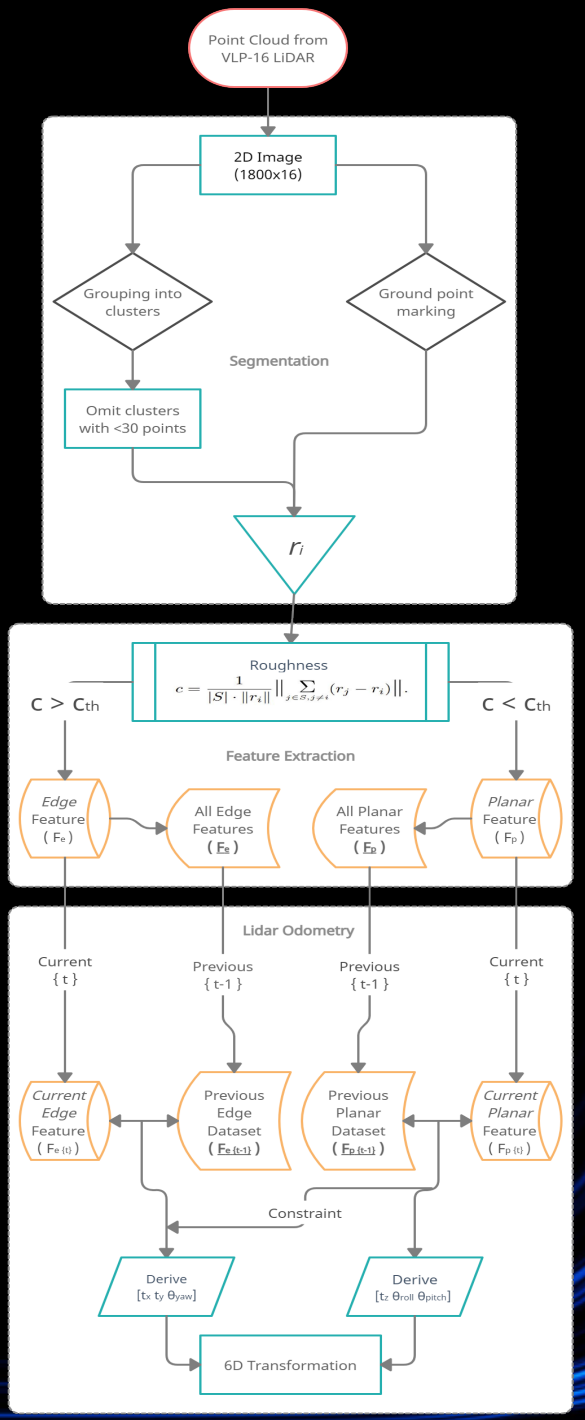


What is LeGO-LOAM?

- A better version of LOAM
- Lightweight and ground-optimized lidar odometry and mapping
- Can achieve real-time pose estimation on a low-power embedded systems.

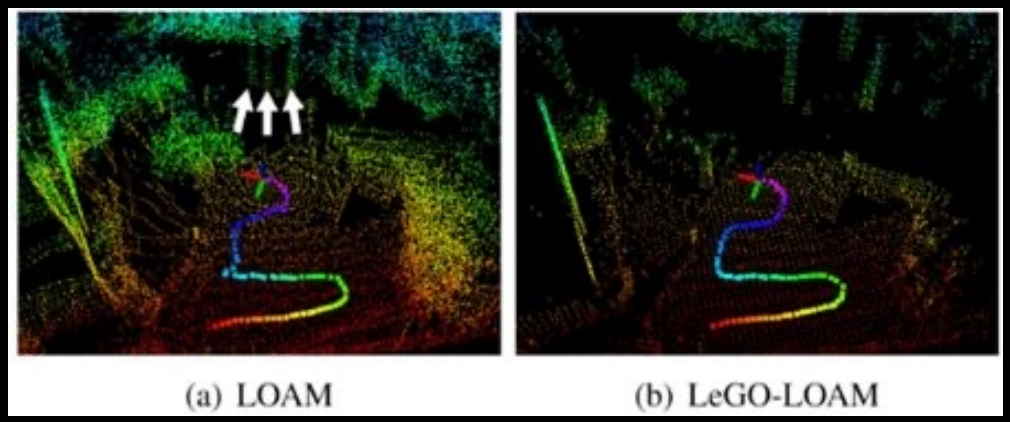
Refresher: LOAM

- Lidar Odometry and Mapping



LOAM vs LeGO-LOAM

- Breaks down computations to smaller parts, allowing for a faster computation process
- Segmentation of ground plane
- Saves features extracted from previous/current scan to generate maps in the next

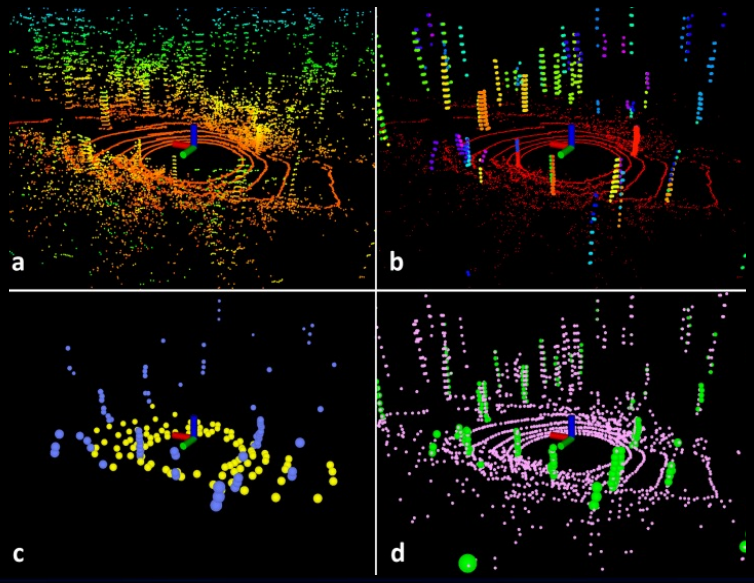


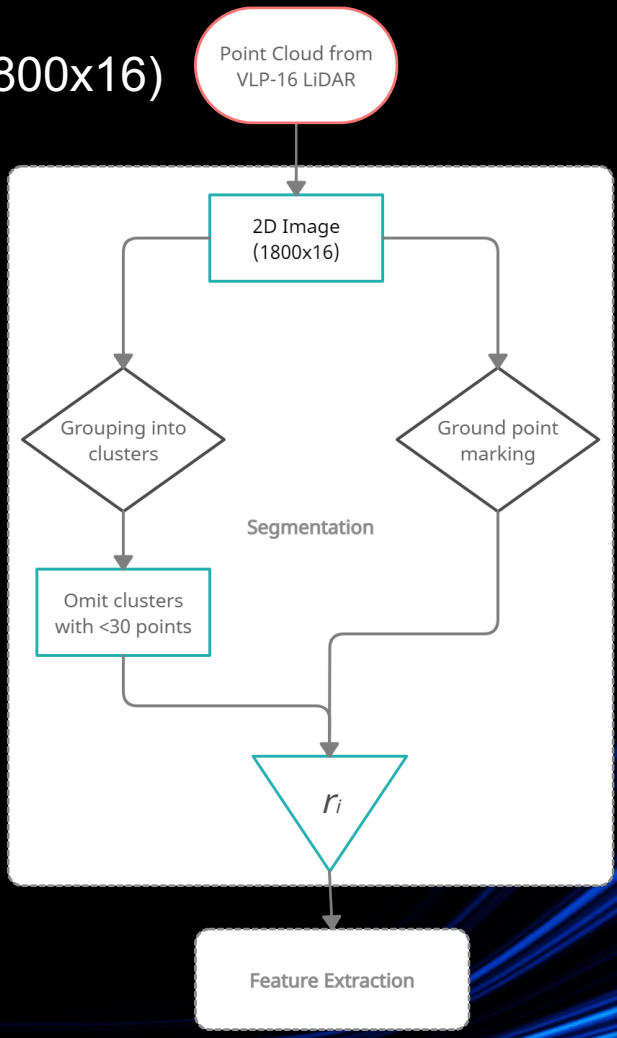
(a) LOAM (b) LeGO-LOAM

Segmentation

Used to discards points that might represent unreliable features after ground separation

- Point cloud projected onto 2D range image (1800x16)
  - Divided into 6 sub-images (300x16)
- Columnwise evaluation to mark ground points
- Points grouped into clusters (if <30 points --> discarded)



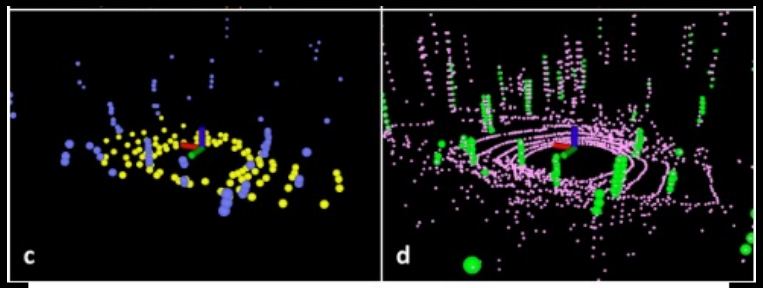


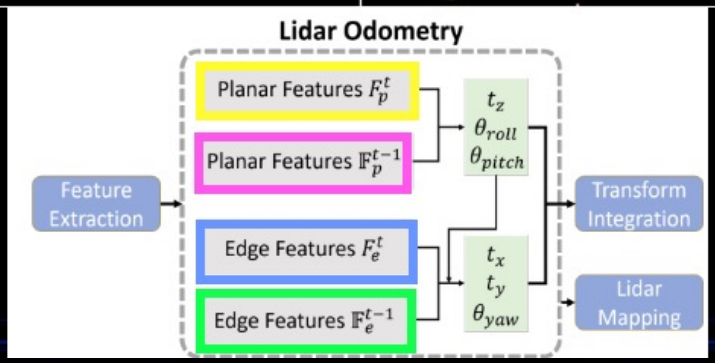
Feature Extraction

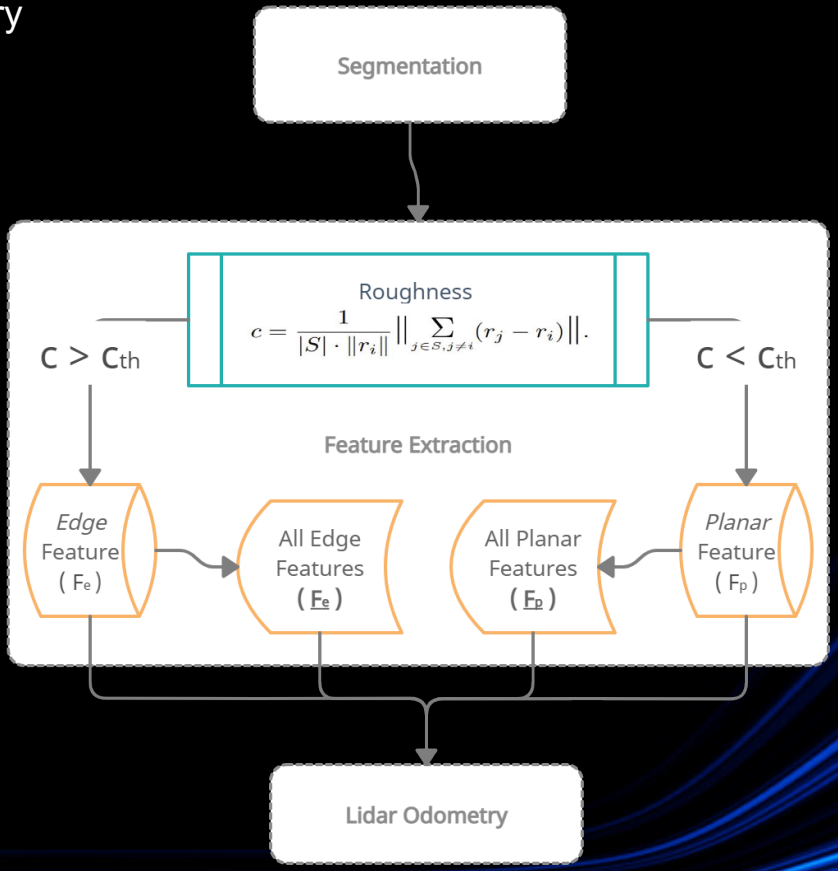
Isolates datapoints for correlation between time segments

- Features are extracted by calculating the roughness of a point in its local region
  - high roughness vs low roughness
- Isolate edge feature points with *maximum* c value from each sub-image
  - This set is F_e used in Lidar Odometry
- Isolate planar feature points with *minimum* c value from each sub-image
  - This set is F_p used in Lidar Odometry

$$c = \frac{1}{|S| \cdot \|r_i\|} \Big\| \sum_{j \in S, j \neq i} (r_j - r_i) \Big\|.$$





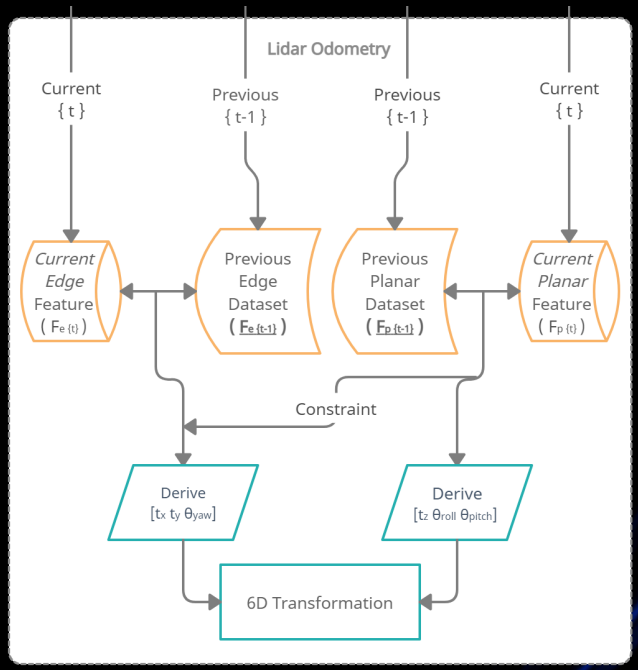


Lidar Odometry

Estimate the sensor motion between two consecutive scans using only the previous scan

- **Label Matching:**
  - Segmentation improves odometry accuracy and efficiency
  - Points from same clusters are matched together

- **Two-Step L-M Optimization:**
  - $[t_z \;\; \theta_{roll} \;\; \theta_{pitch}]$ are estimated by matching planar features F_p^t with previous scans F_p^{t-1}
  - $[t_x \;\; t_y \;\; \theta_{yaw}]$ are estimated similarly with edge features F_e^t with previous scans F_e^{t-1}
    - Edge featutes use [tz θroll θpitch] as constraints



Lidar Mapping

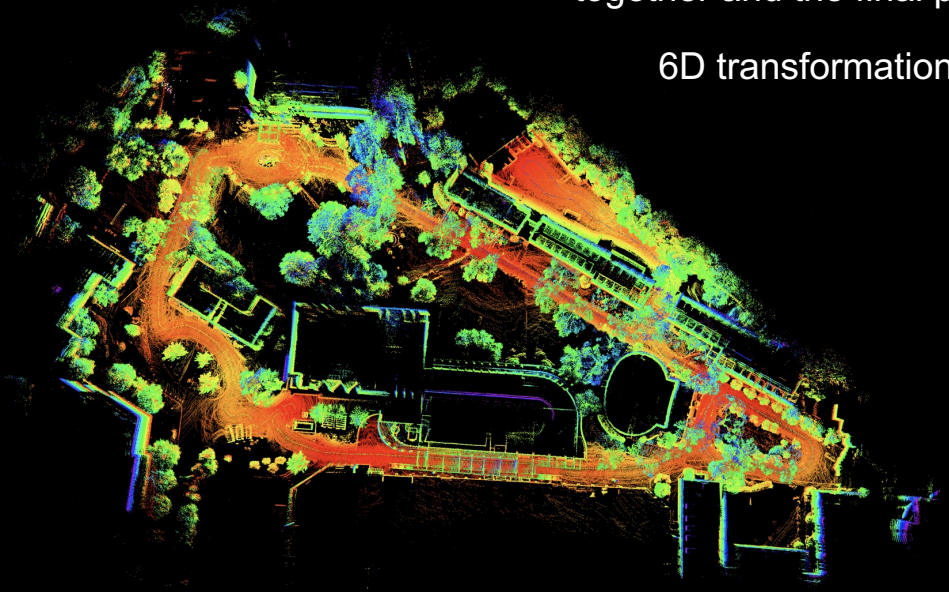
Matches extracted features from current dataset to previous point cloud map

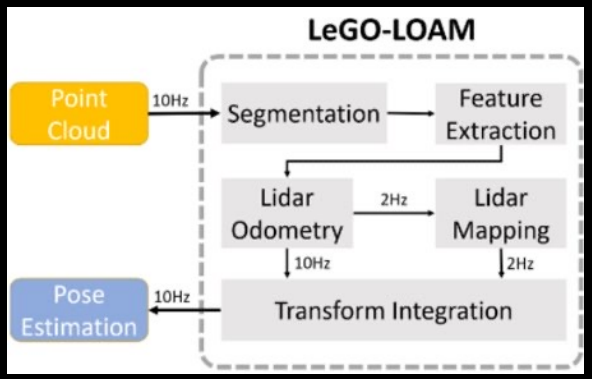
- Difference with LOAM: save individual planar and edge features
- Runs at lower frequency (2Hz) -> more features to process
- L-M approach is used again to obtain final transformation of the sensor.

$$M^{t-1} = \{\{\mathbb{F}_e^1, \mathbb{F}_p^1\}, ..., \{\mathbb{F}_e^{t-1}, \mathbb{F}_p^{t-1}\}\}$$

The pose estimation results from lidar odometry and lidar mapping are fused together and the final pose estimate is computed:

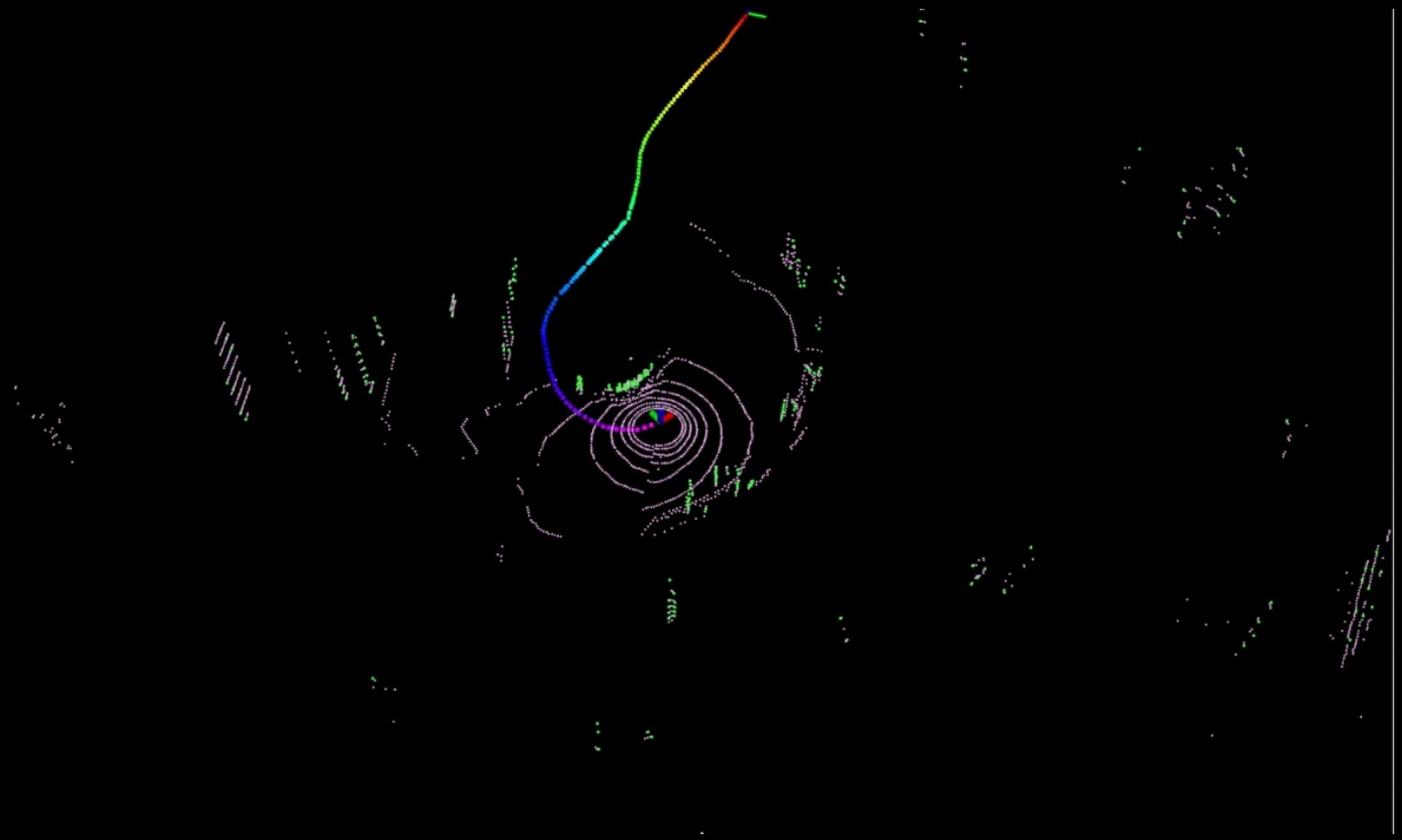
6D transformation [tz,θroll,θpitch] [tx,ty,θyaw]





Results

- Feature detection works well



Results on SIT dataset

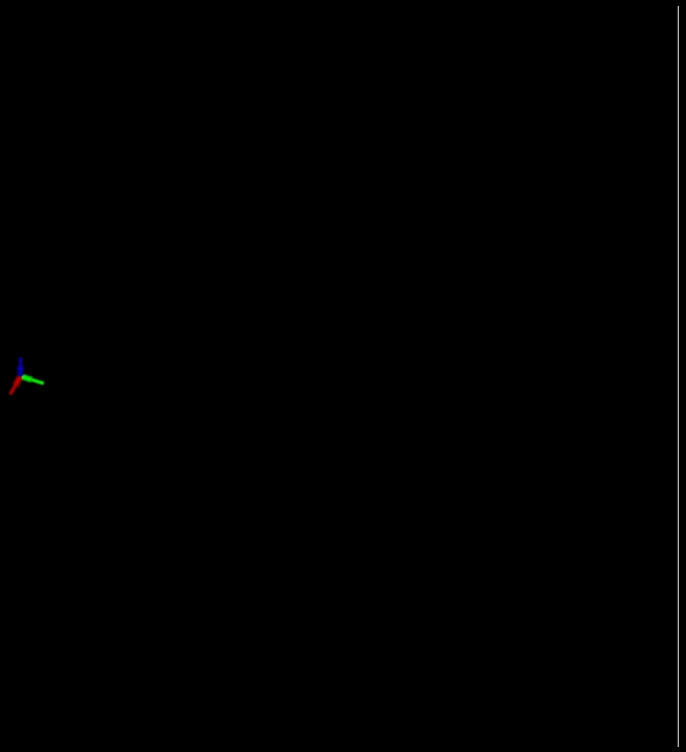
- Loop closure was observed.

Results on NU dataset

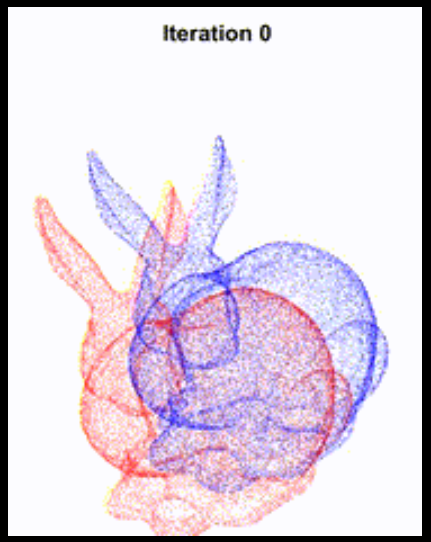
- Not really sure what happened here…

Results on NU dataset

- Loop closure was not observed. Probably because of the speed of the car vs the speed of the UGV used by Steven Institute



Iterative Closest Point (ICP)



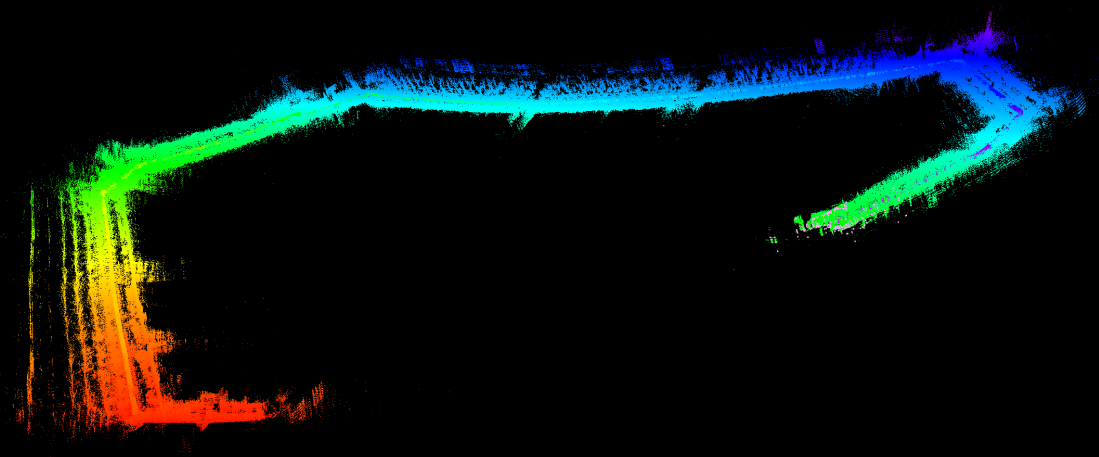
- Computes relative rotation/translation of two corresponding point clouds.
- Input: Current point cloud scan(P1) and target point cloud (P2), Output: rotation R and translation t.
- find the closest neighbor for every point of P1 in P2
- find the minimum R and T for the correspondence assignment
- Rotate and Translate P1 by R and T
- Stop if the difference between previous, and the current R and T is smaller than a thresold.

This pose estimated from this algorithm is used
to minimize the drift after the loop closure

Next steps

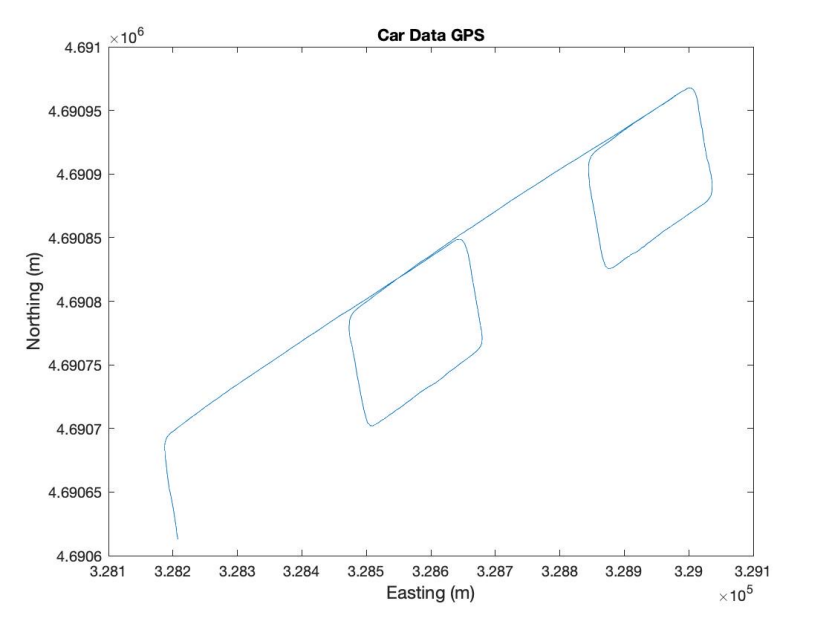
- Possibly collect our own data
- Find out what is breaking lego loam in loop closure – attributing it to being velocity differences between car and robot [changing enviroment (cars)]
- Explore other methods to fix lego loam loop closure i.e. lego-loam bor, sc-lego-loam. What makes these better
- Test on KITTI dataset to actually see what the odometry accuracy is

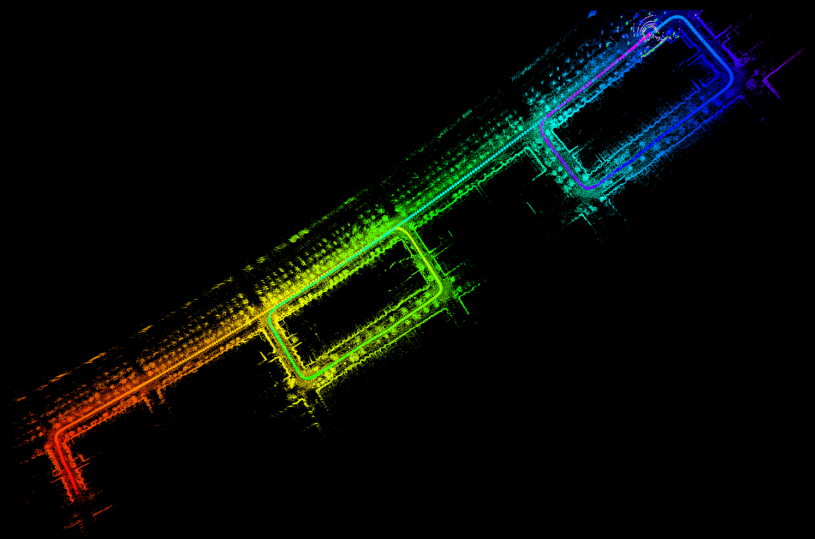
SC-Lego Loam (scan context) w/ IMU



- Loop Closure did not occur
- This is not the correct path similar to regular lego-loam

SC-Lego Loam (scan context) w/out IMU

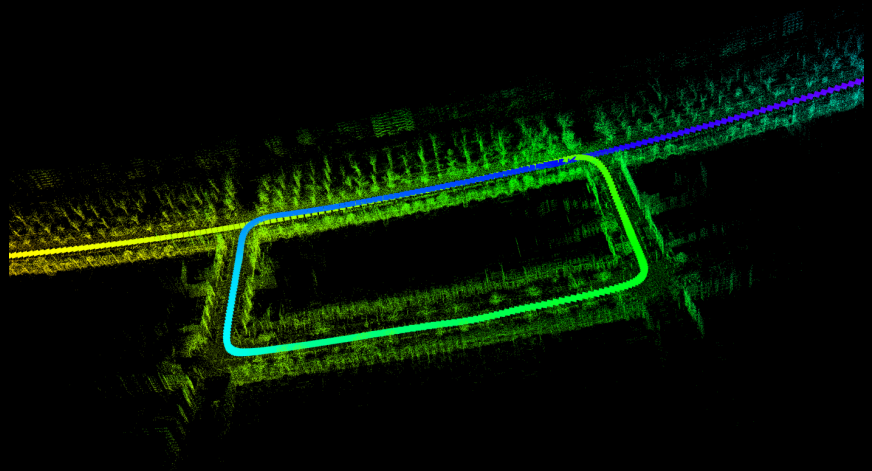




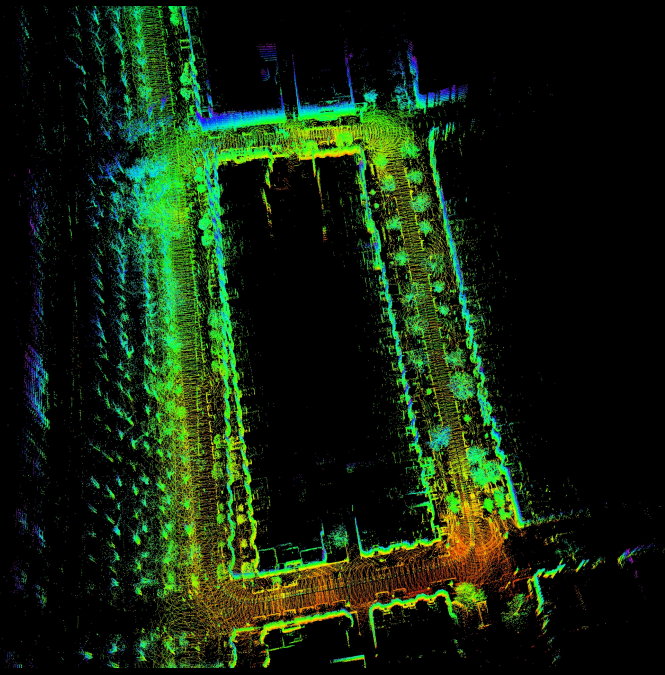
- Successful Loop closure
- Path matches accurately

Loop Closure

SC – LeGO-Loam



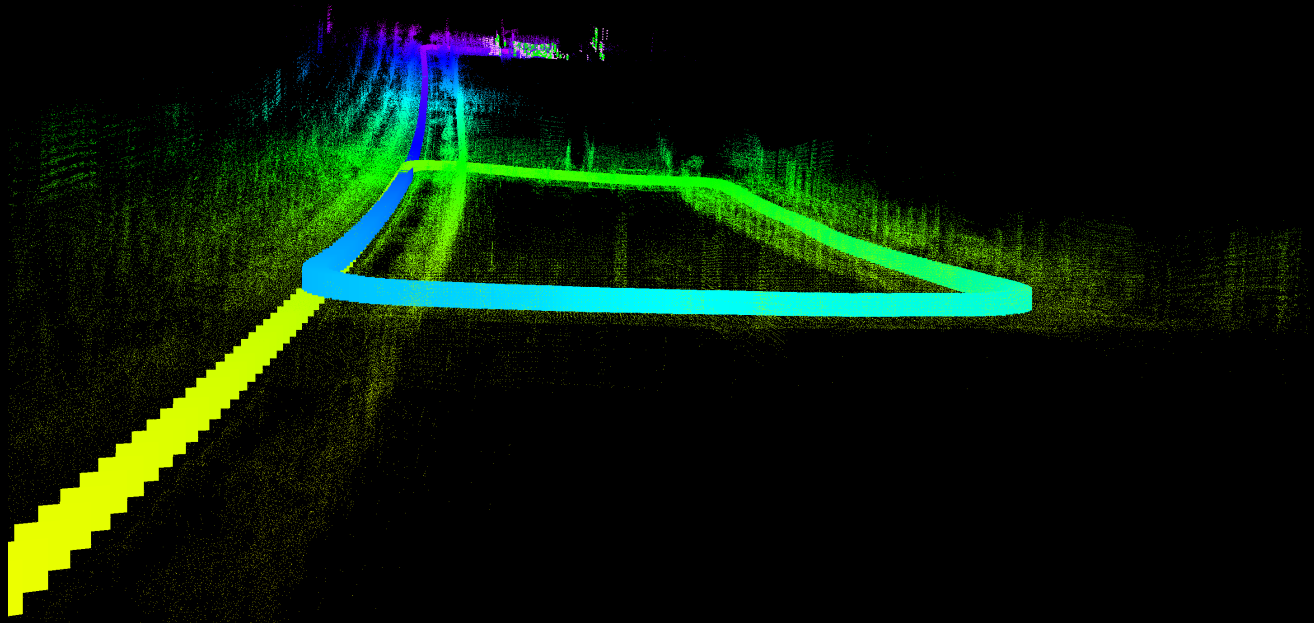
LeGO-Loam



- Loop closure successful in SC LeGO-Loam vs LeGO-Loam

SC-Loop Closure

- Z axis lines up better in comparison to regular LeGO-Loam



Interesting Finding