**LeGO – LOAM**

**Overall Goals**

1. Have understanding of data set & LeGO – LOAM algorithm.
2. Implement of LeGO-LOAM with IMU & loop closure for improvement.
3. Compare the difference between LeGO-LOAM, SC-LOAM.

**Resources Used :**

NUance car for collecting our dataset

Velodyne VLP-16 lidar

Vectornav vn-100 imu

ROS Noetic on Ubuntu 20.04 to collect data using various sensors

ROSbag for collection, RVIZ for visualization

**LeGO-LOAM implementation :**

**scanRegistration.cpp**

Point of interpretation for the raw PointCloud data from LiDAR

Feature extraction (corners & flat surfaces) on current scene

**laserOdometry.cpp - 10Hz**

Feature comparisons on consecutive frames

**laserMapping.cpp - 1 Hz**

Interprets the movement of the LiDAR using odometry, publishes & updates a world map

**Error’s encountered while running lego-loam :**

* OpenCV4: No such file named opencv/cv.h: #include <opencv2/opencv.hpp>

* Failed to transform between the frames: Deleted all the backslash in frame\_id

backslash is not allowed in tf2 standard

* ‘Index’ is not a member of ‘Eigen’ : Using the latest version of gtsam
* C++ standard in Cmakelist.txt

**Mapping without the IMU data:**

Chart

Description automatically generated Chart

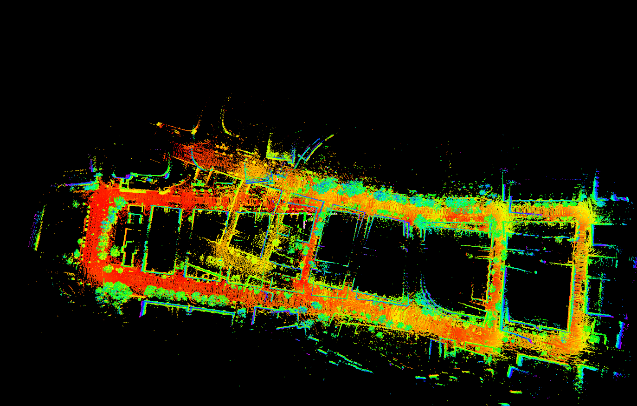
Description automatically generated

A picture containing dark

Description automatically generated

**Mapping with the IMU data:**

Diagram, engineering drawing

Description automatically generated 

Lidar + IMU Mapping, without loop closure comparison.

As we see, IMU improve the mapping quality a little bit. But still not comparable to the Loop closure mapping. Probably we didn’t find a perfect parameter.

We also tried to get the extrinsic transformation by using toolbox, such as ethzasl/lidar\_align and APRIL-ZJU/lidar\_IMU\_calib from github. But the error we got is really high, which is more than 20000. So the result of calibration(shown below) is not good.

**Calibration Result:**

Text

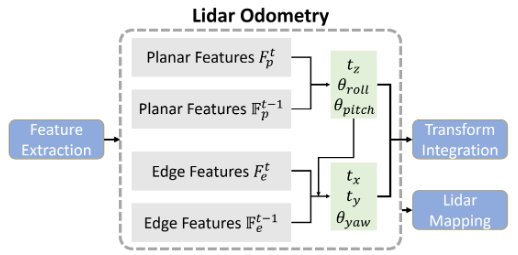
Description automatically generated with low confidence

**The system**

LeGO-LOAM is specifically optimized for a horizontally placed VLP-16 on a ground vehicle. It assumes there is always a ground plane in the scan. The vehicle we are using is Northeastern's Autonomous Car NUANCE. It has a built-in IMU.

[](https://github.com/pagidik/Lidar-Mapping-using-LeGO-LOAM-and-SC-LeGO-LOAM/blob/main/LeGO-LOAM/launch/nuance.jpg)

Lidar odometry performs two-step Levenberg Marquardt optimization to get 6D transformation.

[](https://github.com/pagidik/Lidar-Mapping-using-LeGO-LOAM-and-SC-LeGO-LOAM/blob/main/LeGO-LOAM/launch/odometry.jpg)

**New Lidar**

The NUANCE car is equipped with three different Lidars. We chose Velodyne VLP-16 as it is calibrated with all the other sensors. Make sure before your collect the data you are using the correct address to collect data. The key thing to adapting the code to a new sensor is making sure the point cloud can be properly projected to a range image and the ground can be correctly detected. For example, VLP-16 has an angular resolution of 0.2° and 2° along x & y directions. It has 16 beams. The angle of the bottom beam is -15&deg. Thus, the parameters in "utility.h" are listed below. When you implement a new sensor, make sure that the ground\_cloud has enough points for matching. Before you post any issues, please read this.

extern const int N\_SCAN = 16;

extern const int Horizon\_SCAN = 1800;

extern const float ang\_res\_x = 0.2;

extern const float ang\_res\_y = 2.0;

extern const float ang\_bottom = 15.0;

extern const int groundScanInd = 7;

In case you are using Ouster OS1-64, Use the below configuration in the "utility.h" file and change the lidar topic name accordingly. Note: Ouster lidar users may need to uncomment this line //cloudHeader.stamp = ros::Time::now();

extern const int N\_SCAN = 64;

extern const int Horizon\_SCAN = 1024;

extern const float ang\_res\_x = 360.0/float(Horizon\_SCAN);

extern const float ang\_res\_y = 33.2/float(N\_SCAN-1);

extern const float ang\_bottom = 16.6+0.1;

extern const int groundScanInd = 15;

**New**: The existing LeGO-LOAM algorithm is tested on kinetic, melodic. We have tested the algorithm on ROS Noetic and modified the code accordingly.

If you are using your lidar with an IMU, make sure your IMU is aligned properly with the lidar. The algorithm uses IMU data to correct the point cloud distortion caused by sensor motion. If the IMU is not aligned, the usage of IMU data will deteriorate the result.

**Run the package**

1. Run the launch file:

roslaunch lego\_loam run.launch

Notes:

1. While running the package make sure you have only one package of LeGO-LOAM or SC-LeGO-LOAM in the src folder.

2. The parameter "/use\_sim\_time" is set to "true" for simulation, "false" to real robot usage.

1. Play existing bag files:

rosbag play \*.bag --clock --topic /ns1/velodyne\_points /imu/imu /camera\_array/cam0/camera\_info /camera\_array/cam0/image\_raw

Notes: Though /imu/imu is optional, it can improve estimation accuracy greatly if provided.

**Loop Closure**

The loop-closure method implemented in this package is a naive ICP-based method. It often fails when the odometry drift is too large. For this reason, we have used more advanced loop-closure methods, Scan Context LeGO-LOAM, which features utilizing point cloud descriptor. To run this we need first remove the current lego loam from the catkin\_ws and add the SC-LeGO-LOAM in the src folder. Use catkin\_make to make the files. This algorithm takes two inputs one is the query feature points and the second is the existing map. The performance of this algorithm can be best visualized if there is atleast one loop closure.