## 3.1\_b A-LOAM for 3D LiDAR SLAM

A-LOAM (Advanced Lidar Odometry And Mapping) is an improved framework from LOAM, using Eigen and Ceres Solvers to simplify code structure. A-LOAM use two parallel algorithms: a low-precision high-speed odometer and a low-speed high-precision odometer, combining the two odometers to obtain real-time map updates. The feature matching method adopted improves the accuracy and operation efficiency of the algorithm.

In this task, we refer this repo[1] to help do the task and directly generate odometry file for **EVO** evaluation. The process of algorithm implementation is: 1) Develop the Aloam from repo[1] with modifications needed. 2) Create a folder named *‘txt’* with ‘*00.txt’* and empty *‘aloam.txt’*. 3) Launch the Aloam and then meanwhile play *‘<3dlidar.bag>’*. 4) Evaluate the performance of the algorithm with EVO*.*

Notice that in step 1, there are some modifications we did which the README.md do not mention.

* Firstly, uncomment the *“//generate the KITTI format trajectory result”* part in the *laserMapping.cpp,* so that theAloam lidar odometry result will be stored in *‘aloam.txt’.*
* Secondly, the ‘*ceres::LocalParameterization’* and ‘*ceres::EigenQuaternionParameterization’* are deprecated in the latest release of Ceres Solver (v 2.1.0) and need to be replaced to *‘ceres::Manifold’* and *‘ceres::QuaternionManifold()‘.*

The result of the EVO evaluation of this 3D SLAM task and the screenshot of algorithm running in rviz is shown in Figure 3.1.b.1.

|  |
| --- |
| 黑暗中的灯光  中度可信度描述已自动生成绿色的灯光  低可信度描述已自动生成 |
| Figure 3.1.b.1 3D SLAM algorithm running in rviz and performance |

From the result, it can be seen that the max error value is , the mean error value is and the min error value is . Thus, the performance of this algorithm is not bad.

To further improve the performance, we tried tuning some parameters (‘DISTANCE\_SQ\_THRESHOLD’ from to and ‘SCAN\_PERIOD’ from to ) in *LaserOdometry.cpp* and successfully reduced the maximum error from toas shown in Figure 3.1.b.2.

|  |
| --- |
|  |
| Figure 3.1.b.2 3D SLAM task result after tuning |

The disadvantage of A-LOAM is that it does not do back-end closed-loop detection, so perhaps we can seek to correct motion estimation drift through closed-loop detection. Alternatively, the output can be added to a filter, such as a Kalman filter, to further reduce motion estimation drift.

## 3.2\_BONUS FAST-LIO for 3D LiDAR SLAM

FAST-LIO is a computationally efficient and robust Lidar-Imu Odometry method proposed by the MARS Laboratory of the University of Hong Kong. It utilizes tightly coupled, iteratively extended Kalman filters to fuse lidar landmarks with IMU data for robust navigation in fast-motion, noisy or clutterous environments.

In this task, we refer the repo[2] of the FAST-LIO original authors to help do the task. Firstly, we followed the REDAME.me in the repo to install and build this algorithm in our computer. Then, to implement it in the rosbag “*have\_fun.bag*” given, we still need to develop it with some own operations:

* Confirm that we should use the launch file *“fast\_lio mapping\_velodyne.launch”* for Velodyne.
* To get the topics information in the rosbag, we use the *rosbag info* command firstly to get the topics inside and set the LiDAR point cloud topic name and IMU topic name in “*config/velodyne.yaml”* in FAST-LIO.
* To save the Odometry data in text file, we refer the function in Task 1.b given and add a callback function for Odometry topic and a subscriber for *“/Odometry”* topic in “*LaserMapping.cpp”* which is shown in Figure 3.b.2 and 3.b.3 in Appendix.
* After obtaining the odometry data text file, we found that the lines inside is which is not same as 2762 lines in *“have\_fun.txt”.* To better do the EVO evaluation for them, we skip the record every two sets of data, save the number of *“FASTLIO.txt”* file lines to 2747, and then delete the number of *“have\_fun.txt”* file lines that have more than 2747 to make the number of lines of the two files equal.

The result of the EVO evaluation of FAST-LIO in this rosbag and the screenshot of algorithm running in rviz is shown in Figure 3.2.1

|  |
| --- |
|  |
| Figure 3.2.1 FAST-LIO running in rviz and performance for have\_fun.bag |

From the result, it can be seen that the max error value is , the mean error value is and the min error value is .

The performance in this rosbag may be affected by some parameters setting and our modification of how to output odometry data. Another issue is that we have the warning message *"Failed to find match for field 'time'.”,* during running, which means the timestamps of each LiDAR points are missed in the rosbag file. And that is important for the forward propagation and backward propagation.

The disadvantage of FAST-LIO may be that it must support Livox serials LiDAR firstly. But totally it’s an excellent work. It can be applied to unused lidar, and maintaining maps by incrementing the k-d tree data structure kd-tree improves performance.

## Reference

[1] nuslde (2023) aloam\_lidar\_odom\_result\_generate [Source Code] <https://github.com/nuslde/aloam_lidar_odom_result_generate>

[2] Cris.Wei (2021) FAST-LIO [Source Code] <https://github.com/hku-mars/FAST_LIO>

## Appendix

Callback function ***Odom\_sub*** in ***LaserMapping.cpp***

|  |
| --- |
|  |
| Figure 3.2.2 Callback function Odom\_sub |

ROS Subscriber ***sub\_odom*** in ***LaserMapping.cpp***

|  |
| --- |
|  |
| Figure 3.2.3 Subscriber sub\_odom |