slam test of hdl_graph_slam, LeGO_LOAM and LIO-SAM

1. Introduction

1.1. hdl_graph_slam

Itis an open source ROS package for real-time 6DOF SLAM using a 3D LIDAR. It is based on 3D Graph SLAM with NDT scan matching-based odometry estimation and loop detection. It also supports several graph constraints, such as GPS, IMU acceleration (gravity vector), IMU orientation (magnetic sensor), and floor plane (detected in a point cloud).

The github address: https://github.com/koide3/hdl_graph_slam

1.2. LeGO_LOAM

This framework is designed as a lightweight and ground optimized lidar odometry and mapping (LeGO-LOAM) system for ROS compatible UGVs. The system takes in point cloud from a Velodyne VLP-16 Lidar (palced horizontally) and optional IMU data as inputs. It outputs 6D pose estimation in real-time.

The github address: https://github.com/RobustFieldAutonomyLabT/LeGO-LOAM

1.3. LIO-SAM

It a framework for tightly coupled lidar inertial odometry via smoothing and mapping, for performing real-time state estimation and mapping in complex environments. It takes use of lidar, IMU, GPS and so on for better slam performance. It is designed by the same author of LeGO-LOAM.

The github address: https://github.com/TixiaoShan/LIO-SAM

2. Process

The hardware we use is a notebook with Intel® Core™ i7-1065G7 CPU @ 1.30GHz × 8where we installed the Ubuntu 18.04, ROS melodic, and these three frameworks. We use the same dataset and then use *'top'* command to grab the cpu and memory usage information. After that, we save the maps in the form of PCD and the trajectories by rosbag. We will show them in the result section.

3. Dataset

Since the LIO-SAM need the IMU data for normal running, we must select the dataset with *lidar* data and *IMU* data both for comparing. The dataset we choose is a park dataset which is recommended in the LIO-SAM. This dataset is collected using handheld devices by a walking researcher including lidar data and IMU data. The scene and the pointcloud map is shown in Figure 1.

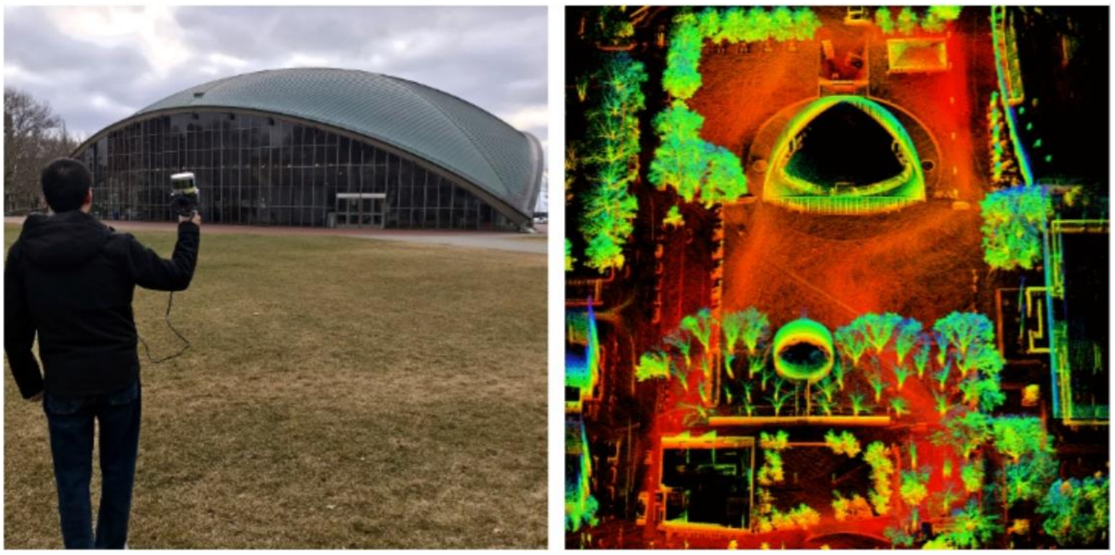

Figure 1 The collection scene and pointcloud map

4. Result

4.1. The slam performance

We use the *pcl_viewer* and *cloudCompare* to display the map using the PCD files saved before. We show the global map and local map both for better comparison. We can see them in Figure 2~7.

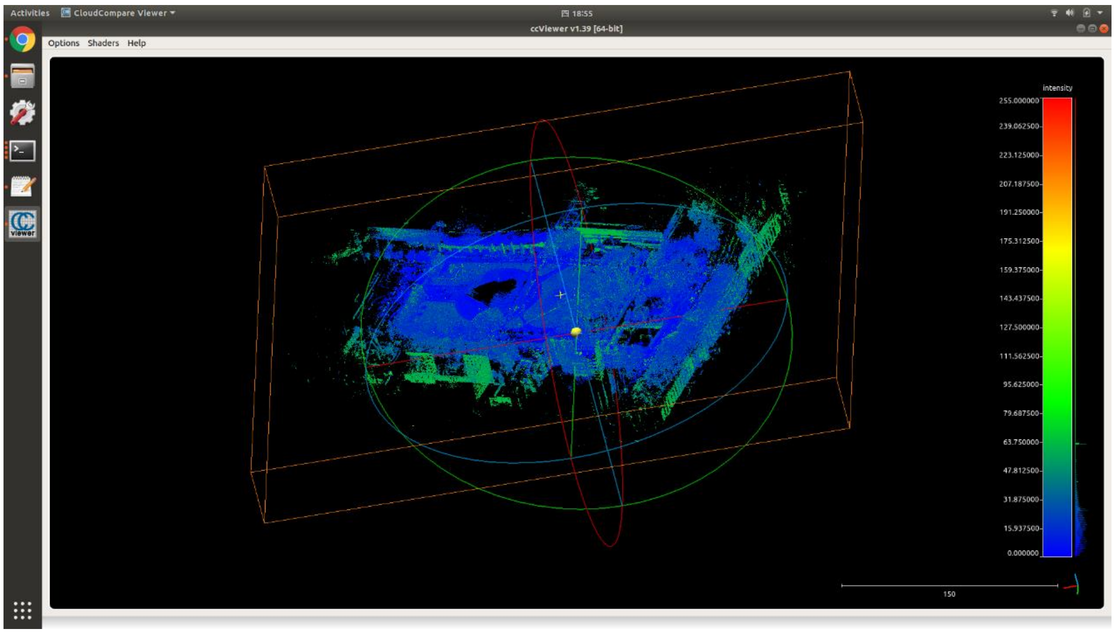

Figure 2 The global map of LIO-SAM

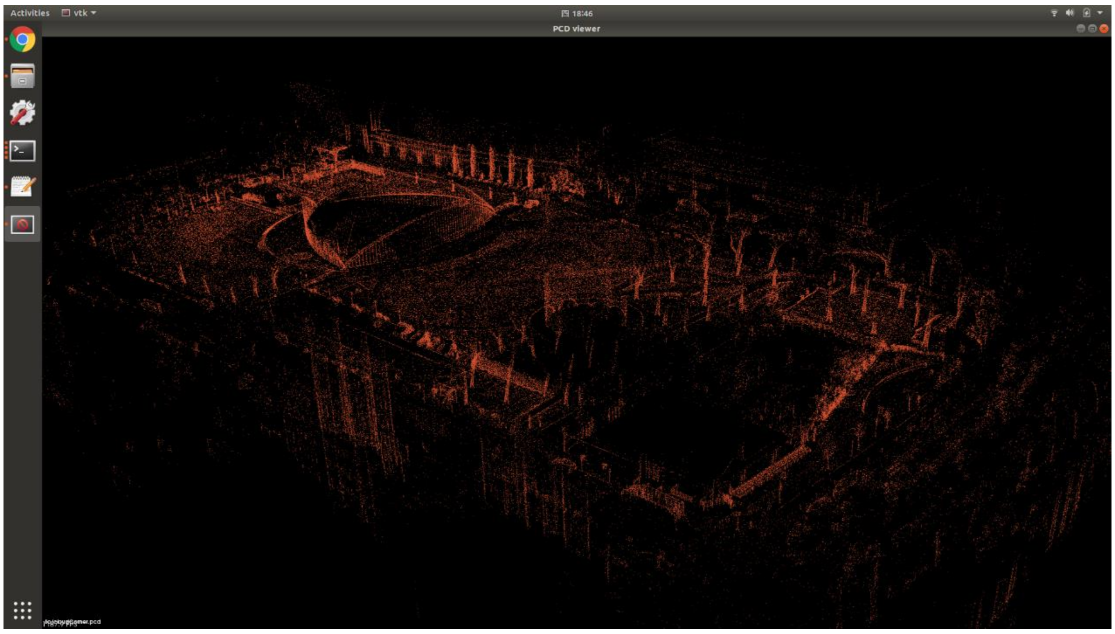

Figure 3 The local map of LIO-SAM

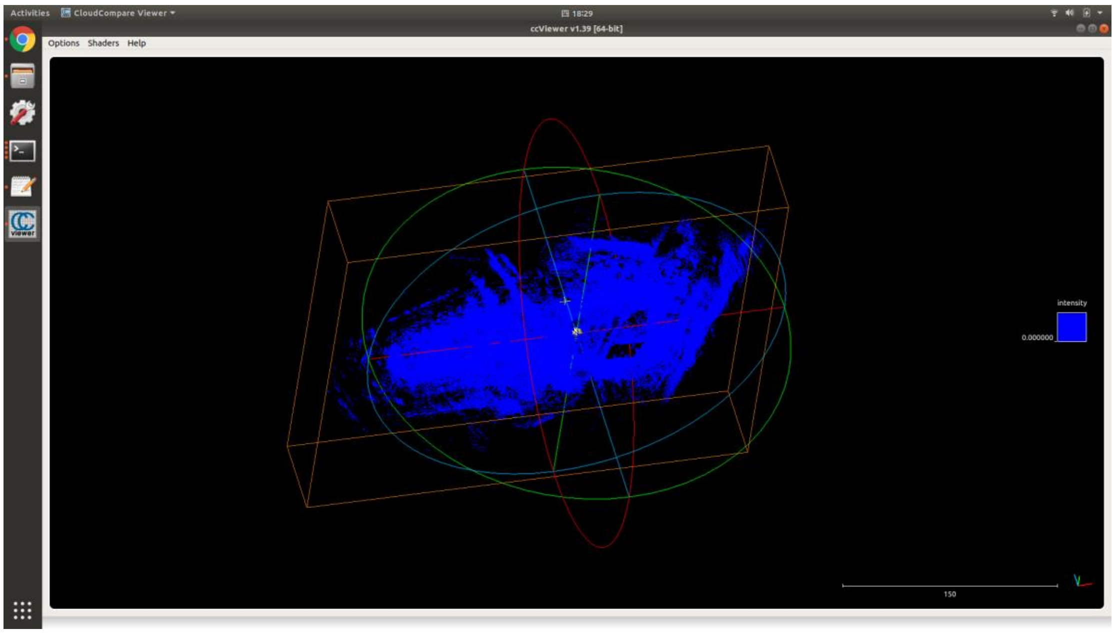

Figure 4 The global map of hdl_graph_slam

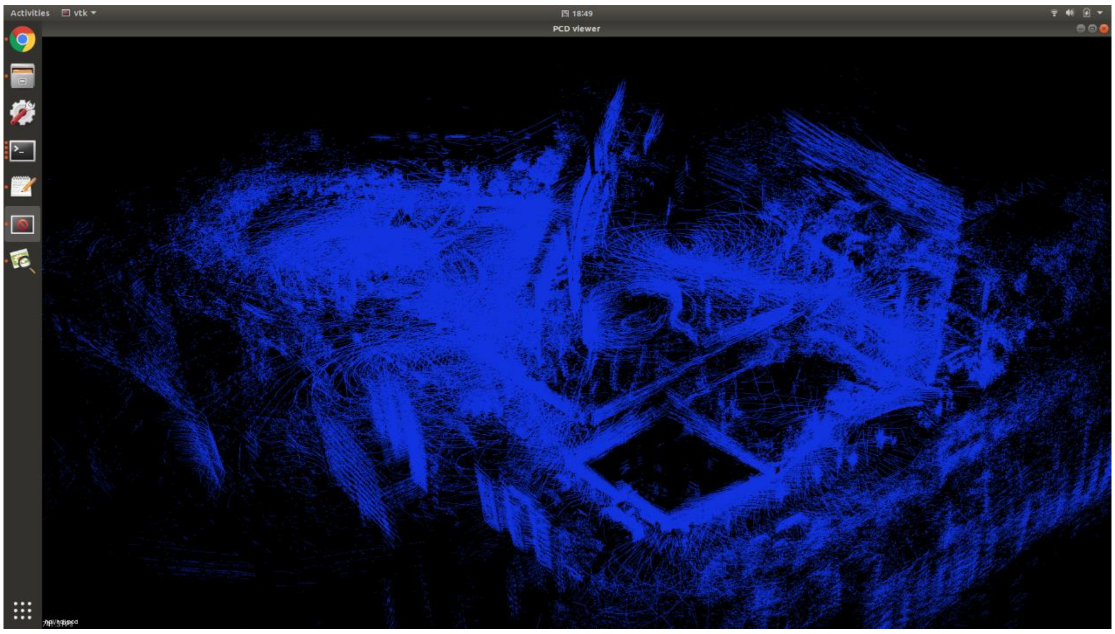

Figure 5 The local map of hdl_graph_slam

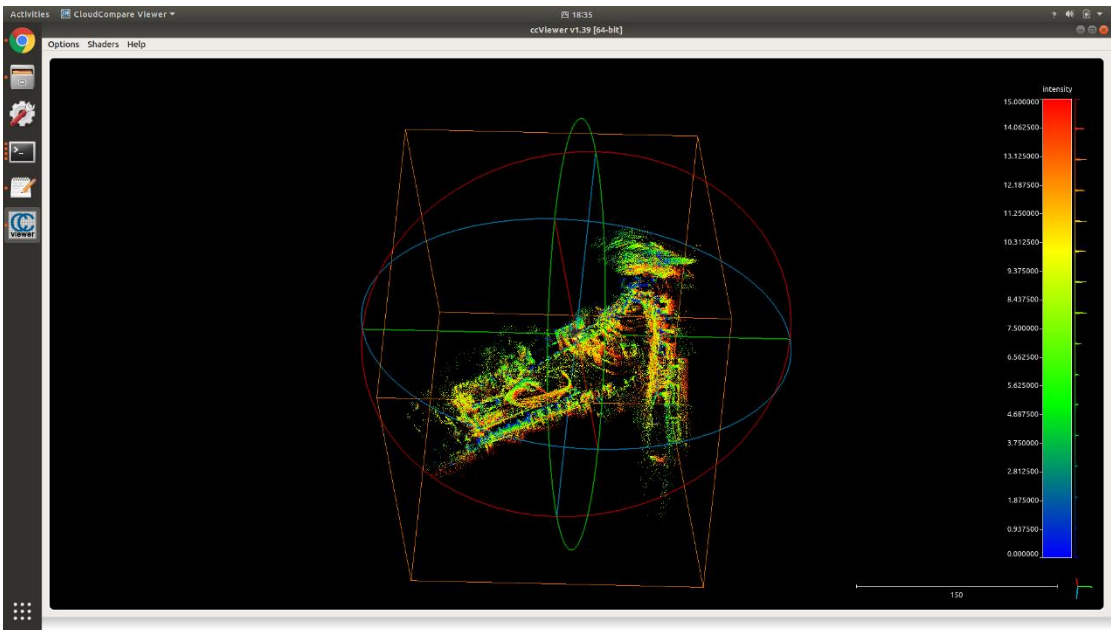

Figure 6 The global map of LeGO-LOAM

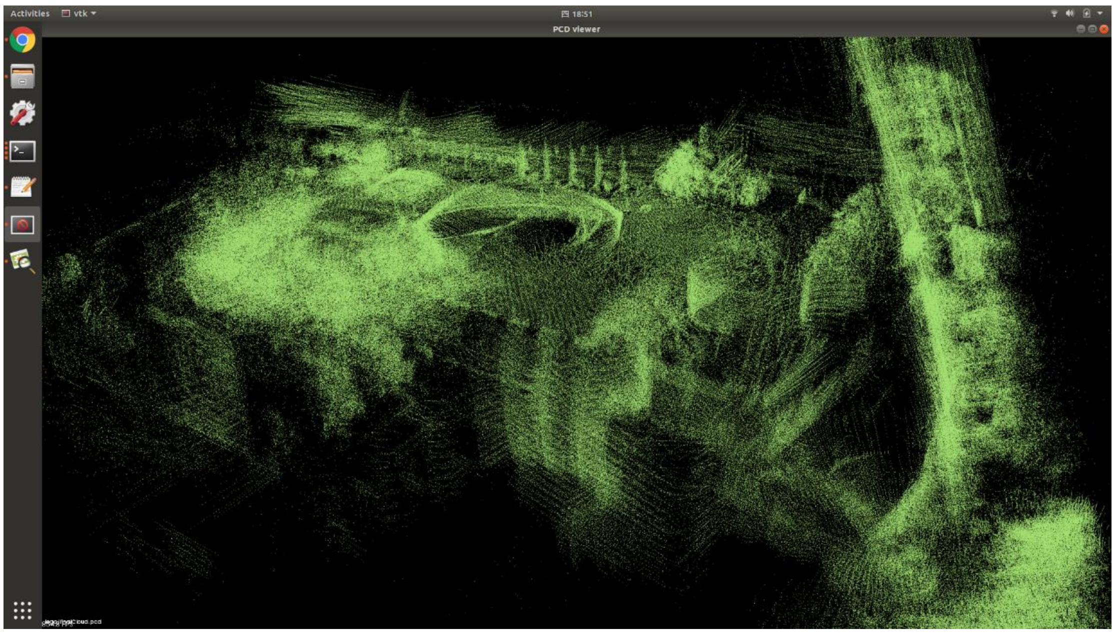

Figure 7 The local map of LeGO-LOAM

As shown in the figures above, we can see that the LIO-SAM well restored the scene and has a better slam performance than the other two. The hdl_graph_slam does well in the early stage, but the error becomes lager as time goes on. The LeGO-LOAM also does well earlier but is the worst in the whole process, even with a very serious offset on the Z-axis.

For better analysis, we use the *evo_traj* to show the trajectories and other information. You can see them in Figure 8~10.

The trajectories are shown in Figure 8. Since there is no ground truth of this dataset, we align the trajectories for better comparison. The green trajectory of LIO-SAM is smooth and has less offset which is consistent with the original dataset. The black trajectory of hdl_graph_slam is relatively normal in the first half and has some offset, but is completely off course in the second half. The LeGO-LOAM is the worst where there is a very serious offset on the Z-axis.

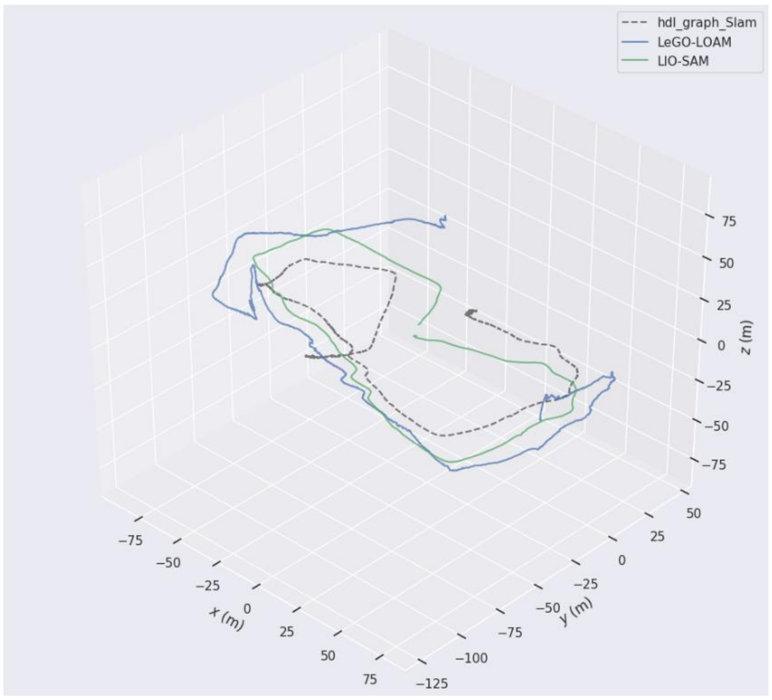

Figure 8 The trajectories of three frameworks

The offsets on xyz axis are shown in Figure 9. We can see that the LIO-SAM has small offsets on three axes. The hdl_graph_slam has small offsets on the x-axis and y-axis in the previous process and z-axis. But there are relatively big offsets on the x-axis and y-axis in the latter part of the process. The LeGO-LOAM is similar to the hdl_graph_slam on the x-axis and y-axis but worse. It also has big offsets on the z-axis.

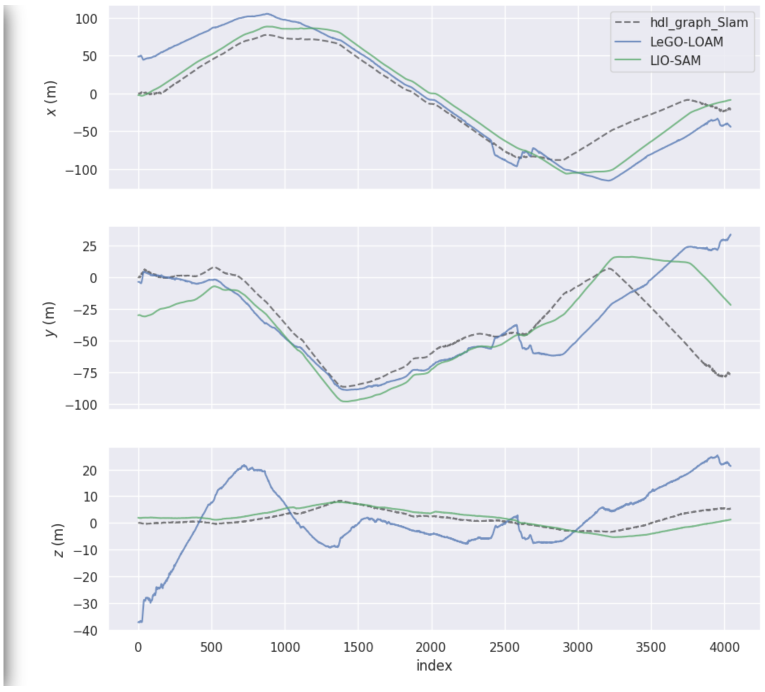

Figure 9 The offsets on the xyz axis of three frameworks

The last is the offsets on the roll, pitch, and yaw which are shown in Figure 9. We can see that the LIO-SAM and hdl_graph_slam both relatively normal on the three scales where the former is better. The LeGO-LOAM is the worst which has big angle offsets on three scales

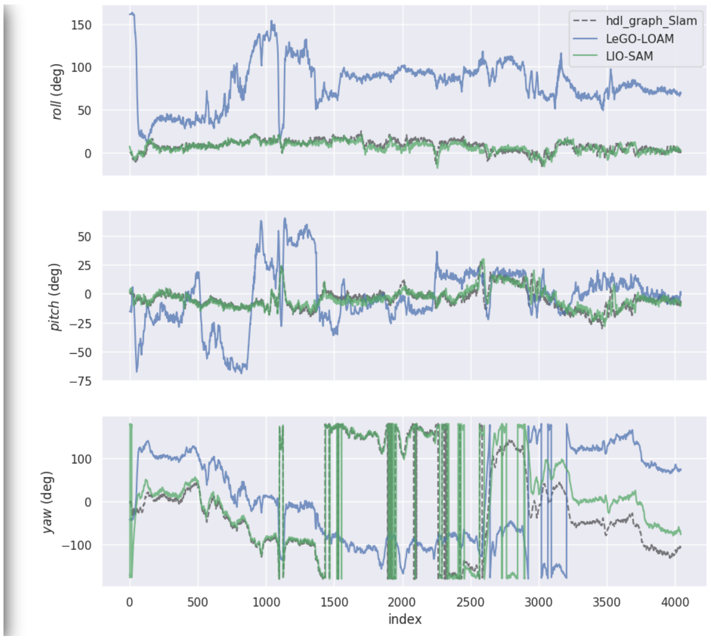

Figure 10 The offsets on the rpy of three frameworks

.

Let’s analyze the reasons. The hdl_graph_slam add the floor detection function relative to the LeGO-LOAM so that it has better stability on z-axis. But the offsets on all parts of two frameworks accumulate on long time slam task which will lead to serious slam error. The LIO-SAM use the IMU data to modify the slam process and improve its stability on long time slam task. Also there are some other methods used in it to improve the slam ability such as the sliding window approach, scan-matching at local scale and so on. This framework also allow user to use the GPS, compass, or altimeter to improve the slam performance.

4.2. The cpu and memory usage of three frameworks

We show the cpu and memory usages of three frameworks in Figure 11 and Figure 12.

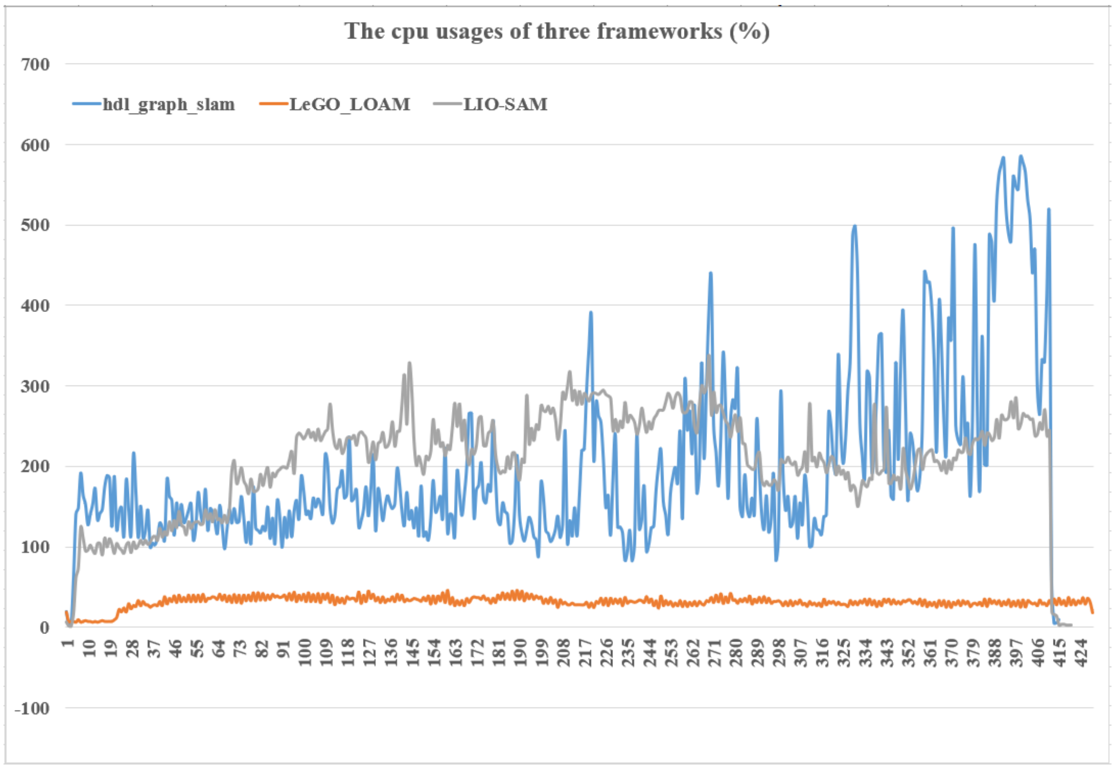

Figure 11 The cpu usages of three frameworks

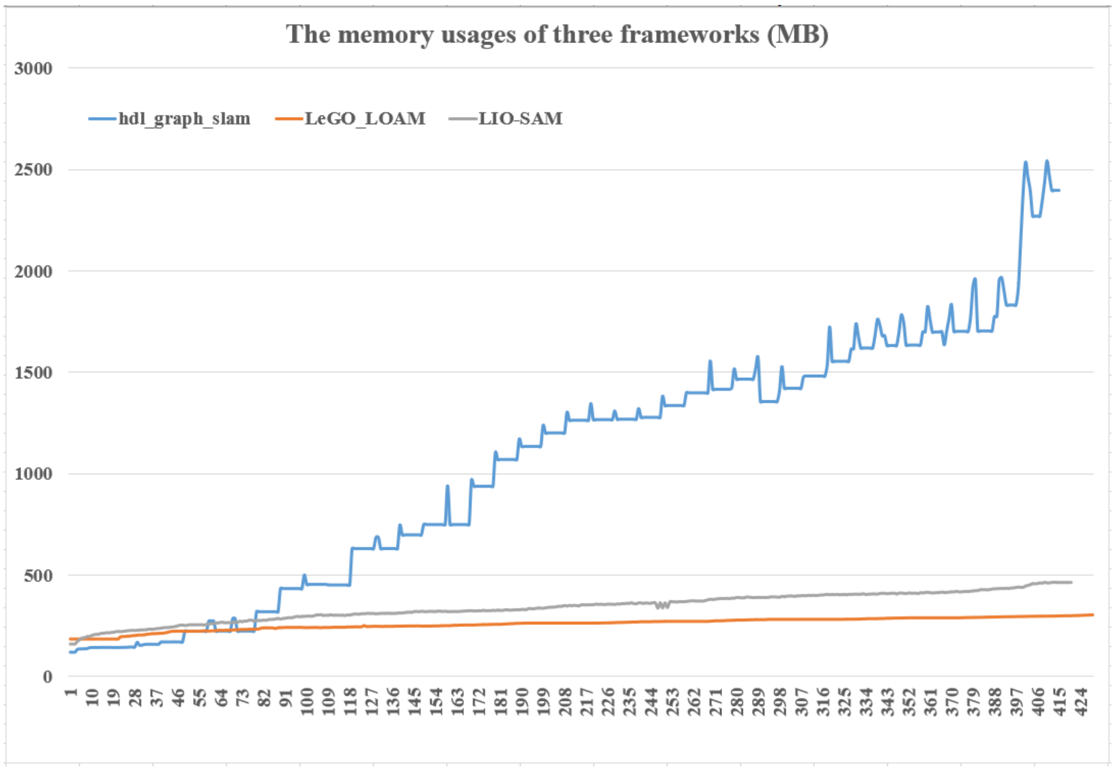

Figure 12 The memory usages of three frameworks

From the figures above, we can see that the LeGO-LOAM takes the least usage of cpu and memory resource which indicates its simple architecture. The LIO-SAM which is designed by the same

author uses more cpu and memory resource to process more types of data and compute in algorithm. It is a very cost-effective thing to sacrifice not a lot of computing resources for better slam performance. The hdl_graph_slam is relatively strange which takes a large usage of resources. I guess it's probably because the framework uses nodelet package in ROS to manage various processes, which results in a large usage of resources.
The detailed data are recorded in another provided TXT file.

5. Conclusion

From the test of three frameworks and papers about them, the multi-sensor fusion is of great help to improve slam performance. Reasonable structural adjustment and algorithm design will not occupy too much computing resources.