ME5413 Report Homework 2: SLAM

Zhao Yimin A0285282X

Chen Ruiyang A0285289J

Luo Yuanchun A0285275U

1 Task 1: Running Cartographer on the given ROS bag

1.1 SLAM Brief

Cartographer is an open-source Simultaneous Localization and Mapping (SLAM) project developed by Google. It supports multi-sensor fusion involving 3D/2D Lidar, IMU, Odometer and GPS. Hence, Cartographer is a powerful tool in the autonomous vehicle scenario and navigation.

1.2 Configuration tuning and algorithm running

After finishing the building and installing procedure according to the official documentation, the lua configuration file needs to be modified to fit the provided 2dlidar.bag. By using the 'rosbag info' command, the sensor data of nav_msgs/Odometry and sensor_msgs/LaserScan is found. In that case, the option of 'use_odometry' is turned to true. We also set 'num_laser_scans = 1' and 'num_multi_echo_laser_scans = 0' for there is only one normal Laser scan topic in the rosbag. Furthermore, the 'TRAJECTORY_BUILDER_2D.use_imu_data' is set to false. Finally, 'catkin_make' to ensure that modifications take effect, and run the command of 'roslaunch cartographer_ros demo_2d_my_robot.launch bag_filename:=<FILE_PATH>/2dlidar.bag' to initiate the whole SLAM algorithm. Fig. 1 shows the SLAM result.

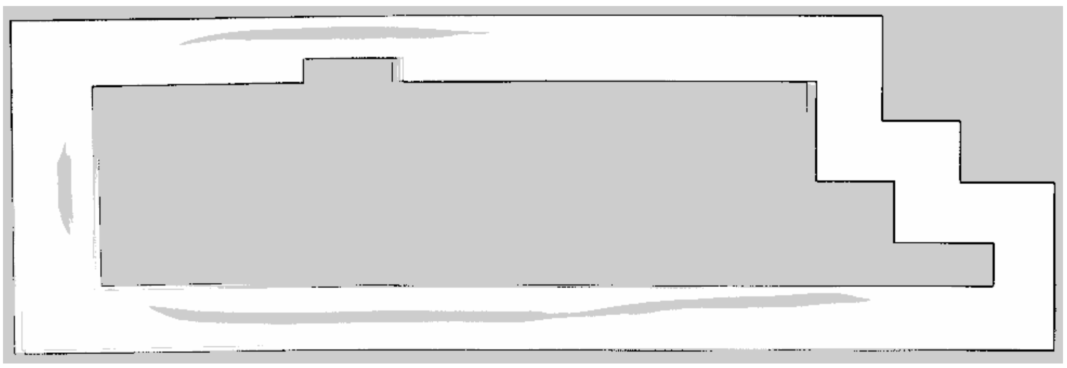
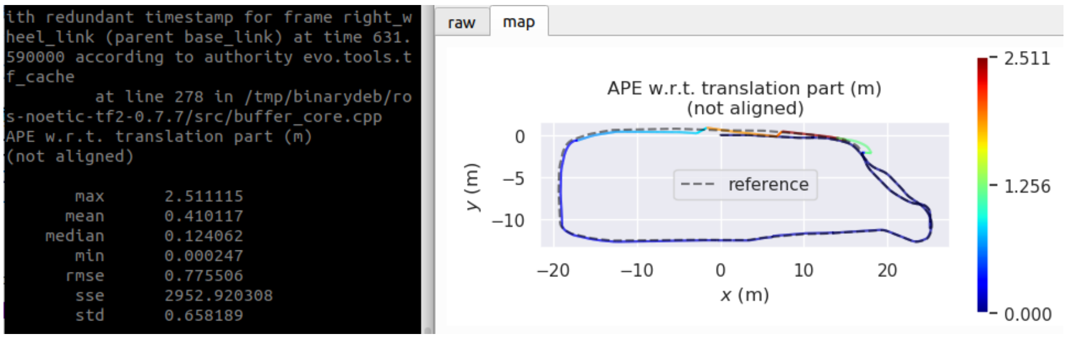


Figure 1: Task 1 final SLAM result picture.

1.3 Bonus: EVO evaluation and discussion

The plot and the metrics table are presented as Fig. 2 generated by the EVO tool. The RMSE in the graph is based on APE, and it indicates that the Cartographer 2d-Lidar SLAM performs well cause the value is 0.775506. The plot of odometer data also confirms this because the red line only appears when loop closure is encountered. The displacement is acceptable in the process of optimization.



```
ith redundant timestamp for frame right_w
heel_link (parent base_link) at time 631.
590000 according to authority evo.tools.t
f_cache
         at line 278 in /tmp/binarydeb/ro
s-noetic-tf2-0.7.7/src/buffer_core.cpp
APE w.r.t. translation part (m)
(not aligned)

       max      2.511115
      mean      0.410117
    median      0.124062
       min      0.000247
      rmse      0.775506
       sse      2952.920308
       std      0.658189
```

Figure 2: Metrics and plot for evaluation gained by EVO tool

Referring to the Fig. 1, the calibration is not perfect. We have analyzed the situation and concluded that the sensor categories are enough. A GPS could be beneficial to loop closure optimization to improve performance.

2 Task 2: Running 3D SLAM algorithms on the given ROS bag

In this part, totally four 3D SLAM algorithms have been applied on the "have_fun.bag" to produce a point cloud map. These four algorithms include pure Lidar SLAM algorithms, specifically ALOAM and LeGO-LOAM, along with multi-sensor fusion SLAM approaches, namely VINS and Cartographer. Subsequently, odometry data has been recorded and transformed into the camera frame for performance evaluation in both TUM and KITTI formats.

2.1 Configuration of SLAM algorithms

ALOAM. Before building the ALOAM repository, it had two prerequisites: Ceres and PCL. Ceres could help provide functions to analyze the point cloud data, while PCL was employed for backend optimizations. Since ALOAM was initially validated on Ubuntu 16.04 and 18.04, certain adjustments were required to ensure the algorithm's proper functioning. For instance, the C++ version in CMakeLists.txt need to change from 11 to 14, and the frame name in four '.cpp' files needs to be modified from '/camera_init' to 'camera_init'. Subsequently, this algorithm could be executed in Ubuntu 20.04. To make it applicable to the "have_fun.bag", the point cloud topic subscribed in the "scanRegistration.cpp" file needs to modify to "/kitti/velo/pointcloud".

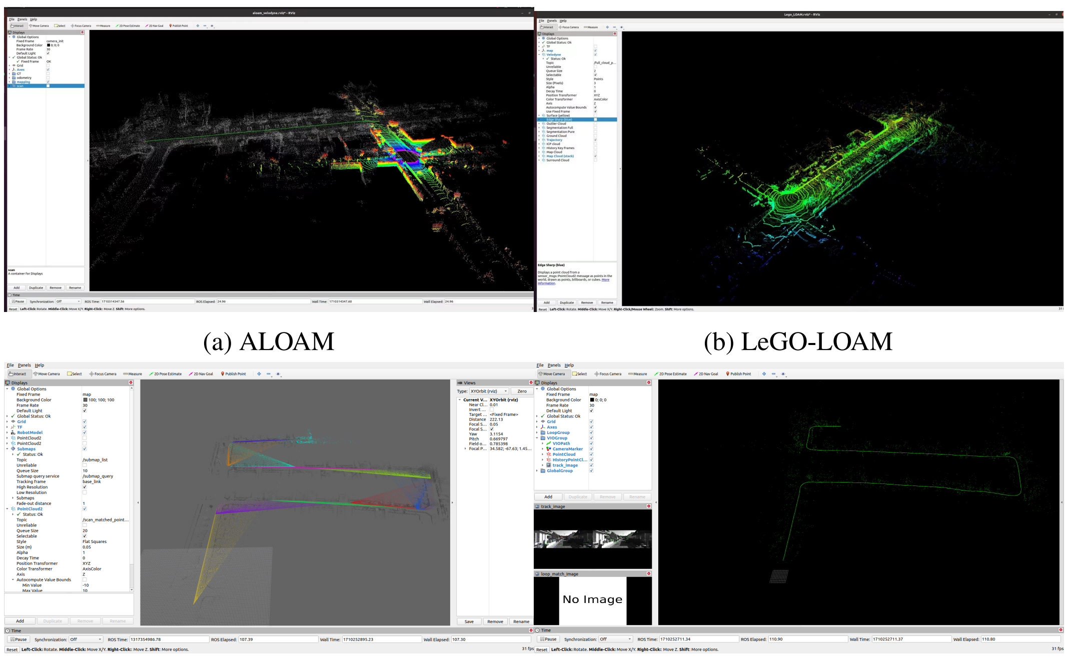
LeGO-LOAM. The installation process of LeGO-LOAM is similar to ALOAM, and this algorithm could apply the data from imu to improve the accuracy of SLAM. Through using "rosbag info" command to review the topics of the given rosbag, it becomes evident that this rosbag contains both imu and point cloud topics. For LeGo-LOAM, the name of the subscribed topic is defined in "utility.h" requiring separate modifications for IMU and point cloud topics. Additionally, a challenge arises with the provided rosbag, recorded using a LiDAR without a distinct ring channel. Consequently, the "useCloudRing" parameter is set to false. Moreover, the command to run the Rosbag should be "rosbag play have_fun.bag –clock", which adds "–clock" when playing the bag to publish the simulation time.

Cartographer 3D-Lidar fusion. Similar to 2D-Lidar Cartographer, the lua configuration file is modified. According to the topics that have_fun.bag provides, the odometer is turned off, **GPS, IMU and 3D point cloud** are turned on. Furthermore, the topic names provided by have_fun.bag are remapped to the ones that the Cartographer understands.

VINS-Fusion. This algorithm fuses the data from **IMU and stereo cameras**. After setting the required topic name for the corresponding sensor, the transformation matrices from IMU to the left camera and right camera are set up. The configuration file of **gray camera and color camera** is tuned separately. Furthermore, the loop closure configuration is turned on.

2.2 SLAM running demonstrations.

After configuration, run launch files to initiate SLAM. Some algorithms require playing rosbag manually. The following Fig. 3 visual demonstrates the SLAM progress for the corresponding algorithm.



(a) ALOAM (b) LeGO-LOAM

(c) Cartographer (d) VINS-Fusion

Figure 3: Rviz screenshots from different category of SLAM algorithm.

2.3 EVO evaluation method

In rosbag data, the recorded format includes timestamps, translations (x, y, z), and rotations (x, y, z, w). During evaluation, we deal with coordinate transformations from an initial frame to a target frame. This process mainly includes the following two key steps: translation adjustment and rotation adjustment.

Translation Adjustment. Translation adjustment is achieved through vector subtraction. The position in the initial frame is represented as $\mathbf{P}_{\text{init}} = (x_{\text{init}}, y_{\text{init}}, z_{\text{init}})$, and the translation vector $\mathbf{T} = (t_x, t_y, t_z)$ describes the displacement from the initial frame to the target frame. After adjustment, the position vector in the target frame $\mathbf{P}_{\text{targ}}$ is:

$$\mathbf{P}_{\text{targ}} = \mathbf{P}_{\text{init}} - \mathbf{T} \tag{1}$$

Rotation Adjustment. The quaternion for rotation is represented as $q = w + xi + yj + zk$, where w, x, y, z are real numbers, and i, j, k are the imaginary units. A quaternion can also be represented as $q = (w, \mathbf{v})$, where w is the real part, and $\mathbf{v} = (x, y, z)$ is the vector part of the imaginary components.

In three-dimensional space, quaternions are used to represent rotations around a unit vector $\mathbf{n} = (n_x, n_y, n_z)$ by an angle θ radians. The rotation quaternion can be calculated as follows:

$$q = (\cos(\frac{\theta}{2}), \mathbf{n}\sin(\frac{\theta}{2})) \tag{2}$$

$$q = (\cos(\frac{\theta}{2}), n_x \sin(\frac{\theta}{2}), n_y \sin(\frac{\theta}{2}), n_z \sin(\frac{\theta}{2})) \tag{3}$$

Rotation Transformation. If q_{init} and q_{targ} respectively represent the rotations of the initial and target frames relative to the global reference frame, then the rotation quaternion from the initial frame to the target frame can be obtained as follows:

$$q_{\text{init\_to\_targ}} = q_{\text{targ}} \otimes q_{\text{init}}^{-1} \tag{4}$$

Here, q_{init}^{-1} represents the inverse quaternion of q_{init}. The inverse q^{-1} for a quaternion $q = (w, x, y, z)$ is:

$$q^{-1} = \frac{(w, -x, -y, -z)}{w^2 + x^2 + y^2 + z^2} \tag{5}$$

Since q is a unit quaternion (common in rotation transformations), the denominator $w^2 + x^2 + y^2 + z^2 = 1$, hence $q^{-1} = (w, -x, -y, -z)$.

Next, we calculate the product of q_{targ} and q_{init}^{-1}, suppose $q_{\text{targ}} = (w_1, x_1, y_1, z_1)$ and $q_{\text{init}}^{-1} = (w_2, x_2, y_2, z_2)$, according to the quaternion multiplication formula: ($q_{\text{init\_to\_targ}} = q_{\text{targ}} \otimes q_{\text{init}}^{-1} = (w_3, x_3, y_3, z_3)$)

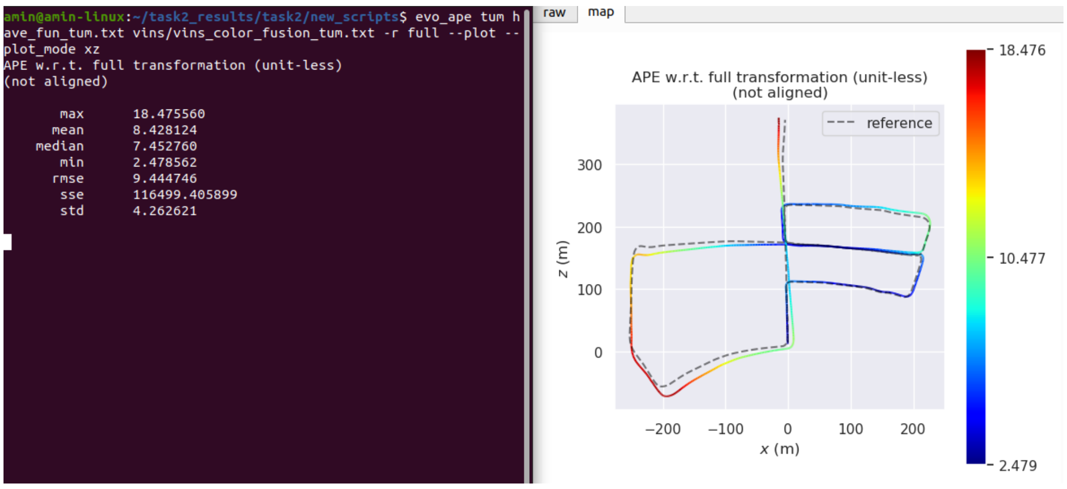
$$w_3 = w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2 \tag{6}$$

$$x_3 = w_1 x_2 + x_1 w_2 + y_1 z_2 - z_1 y_2 \tag{7}$$

$$y_3 = w_1 y_2 - x_1 z_2 + y_1 w_2 + z_1 x_2 \tag{8}$$

$$z_3 = w_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 w_2 \tag{9}$$

Through this method, $q_{\text{init\_to\_targ}}$ can be calculated. In practical applications, by combining translation and rotation transformations, it is possible to calculate transformations like the rotation from the vehicle frame to the camera frame.



```
amin@amin-linux:~/task2_results/task2/new_scripts$ evo_ape tum h
ave_fun_tum.txt vins/vins_color_fusion_tum.txt -r full --plot --
plot_mode xz
APE w.r.t. full transformation (unit-less)
(not aligned)

       max	18.475560
      mean	8.428124
    median	7.452760
       min	2.478562
      rmse	9.444746
       sse	116499.405899
       std	4.262621
```

Figure 4: Enter Caption

<table>
<tr><th rowspan="2">SLAM</th><th colspan="7">EVO results</th></tr>
<tr><th>Max</th><th>Mean</th><th>Median</th><th>Min</th><th>Rmse</th><th>Sse</th><th>Std</th></tr>
<tr><td>ALOAM</td><td>22.55</td><td>13.25</td><td>13.10</td><td>3.03</td><td>14.11</td><td>531249</td><td>4.85</td></tr>
<tr><td>LeGO-LOAM</td><td>95.92</td><td>29.69</td><td>27.64</td><td>2.93</td><td>34.60</td><td>822436</td><td>17.76</td></tr>
<tr><td>Cartographer</td><td>65.11</td><td>34.10</td><td>35.94</td><td>2.47</td><td>36.87</td><td>3738879</td><td>14.03</td></tr>
<tr><td>VINS-Fusion (gray)</td><td>21.50</td><td>10.54</td><td>9.73</td><td>2.52</td><td>11.53</td><td>182258</td><td>4.71</td></tr>
<tr><td>VINS-Fusion (color)</td><td>**18.48**</td><td>**8.43**</td><td>**7.45**</td><td>**2.47**</td><td>**9.45**</td><td>**116499**</td><td>**4.26**</td></tr>
</table>

Table 1: Comparison of median, mean and RMSE value based on APE and RPE, between the different algorithms

2.4 Discussion

Through analyzing the data results, it could find out that multi-sensor fusion SLAM surpasses pure Lidar SLAM in terms of overall performance. This outcome aligns with expectations, as multi-sensor SLAM applies additional sensors such as IMU and camera to capture richer environmental information, thus enhancing the robustness and accuracy of localization and mapping. It is noticable that the VINS fusion SLAM demonstrates the best performance overall performance among the total x algorithm tries in this task attempts in this task. Additionally, utilizing color images instead of grey images as input further enhances the algorithm's accuracy.

In further work, other SLAM methods such as FAST-LIO and SC-LeGO-LOAM would be applied to the given ROS bag to evaluate their performance against the SLAM algorithms already attempted. Besides, the group would take actions to add a loopback detection for A-LOAM to improve its performance. Moreover, experimenting with different ROS bags providing point cloud data in alternative formats could facilitate compatibility with various SLAM algorithms, such as LIO-SAM.

3 GitHub

https://github.com/ztony0712/ME5413_Homework2_SLAM