ME5413: Autonomous Mobile Robotics

Homework 2: SLAM

AY2022/23-Sem 2

Due: 23:59 Sunday, 12 March 2023

Introduction: The aim of this homework is to get you familiarised with point cloud matching through ICP method and how to run and evaluate popular open-sourced SLAM algorithms. After trying out those algorithms, you should be able to appreciate the real-world challenges in SLAM and the pros and cons of each algorithm.

Task 1 and Task 2 require the usage of Numpy, you may find online resources to learn it.

Task 1: Write the ICP Algorithm for Matched Point Cloud

The aim of this task is to write an SVD-based ICP algorithm to calculate transformation from point cloud 1 to point cloud 2. Note that the correspondence is known in this task, which means the same point index represents the corresponding points in 2 frames.

In this task, you are provided a template code in `<main.py>` and 2 point cloud files `<bunny1.ply, bunny2.ply>` in ply format, find the corresponding files in the folder of your group name. Please install the dependencies imported at the beginning of code using pip or anaconda first. You will need to fill up the places labelled with `Todo`.

1. In function icp_core, complete ICP algorithm:

- Step 1: calculate centroid of each point cloud
- Step 2: de-centroid
- Step 3: compute matrix H
- Step 4: apply SVD (third-party library allowed) on H, solve rotation matrix R and translation t
- Step 5: combine R and t into homogenous transformation matrix T, return T

2. To visualize the transformed point cloud 1, you also need to compute transformed point 1 in function svd_based_icp_matched.

After finishing the above code, you may uncomment svd_based_icp_matched(points1, points2) in main() and run the program. If all code you filled up is correct, you will see 3 point cloud in red, green, blue, which represent point cloud 1, point cloud 2, transformed point cloud 1, respectively. Green and blue points should overlap.

Please submit the code you write with comments, the transformation matrix you obtained, and screenshot of visualization result in your report.

Task 2: Write the ICP Algorithm for Unmatched Point Cloud

In real world point cloud paring problems, the correspondence is usually unknown, one popular method is to regard the nearest neighbor point as corresponding point, and iteratively solve the paring problem.

1. In function `svd_based_icp_unmatched`, complete the following code:

In each iteration (loop in the code):

- For each point in points_1, find the nearest point in points_2 and save these points in `points_2_nearest` (points can repeat, size should be the same as points_1)
- Solve T in current iteration (invoke function `icp_core`, should have been completed in task 1)
- Update points_1 using T obtained above
- Update accumulative T

Uncomment the `svd_based_icp_unmatched(points1, points2, n_iter=30, threshold=0.1)` in `main()` and run the program, you can specify the number of iteration and mean distance threshold in the parameters, the program will end if the iteration reaches set number or the mean distance of 2 point cloud is smaller than the threshold.

You should see the visualization of each iteration in the window (they will be saved in a folder named saved_figs). The current transformation T and accumulated T printed in the console. The time cost will also be printed after iteration stops. The final accumulated T should be near the result you have obtained from task 1 if 2 point cloud nearly overlap.

Please write the following result in your report:

- the code you write with comments
- at least 3 intermediate results during the iteration with screenshot, T, accumulated T and mean error
- the final accumulative transformation matrix, mean error, time cost (mark the iteration number and threshold you set)

Task 3: Running SLAM Algorithms

In this task, you are expected to try out different SLAM algorithms on the given ROS bags and evaluate their performances analytically using the convenient EVO tool (https://github.com/MichaelGrupp/evo). The performance of each algorithm is assessed based on the Absolute Root Mean Square Error (RMSE) over the entire sequence.

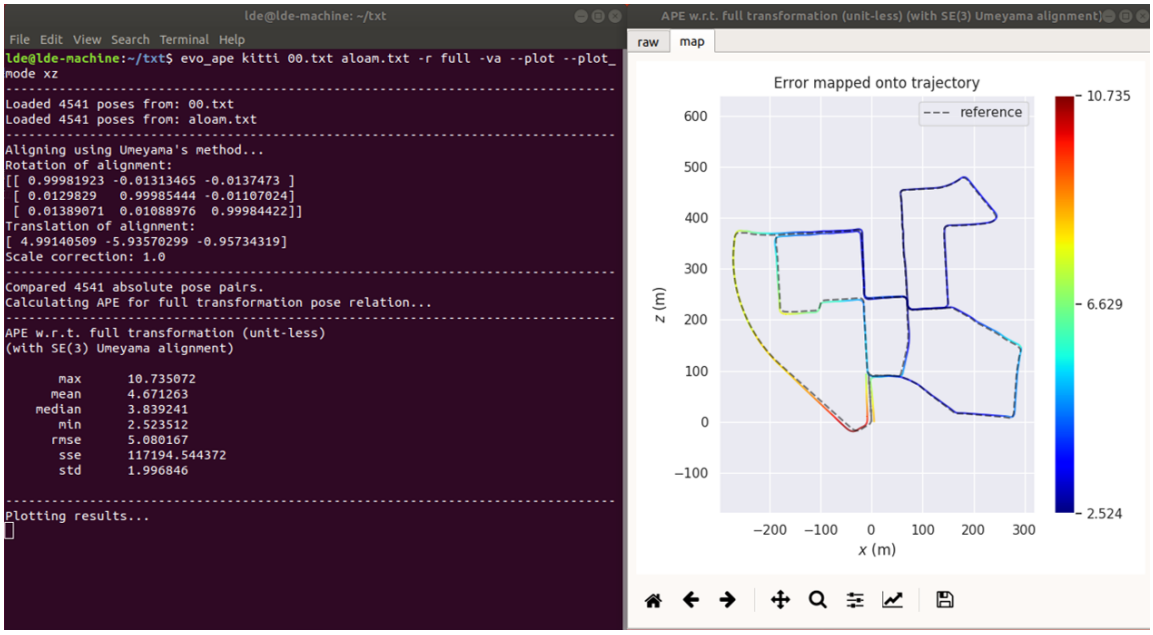
1. Using Cartographer (2D LiDAR) (https://google-cartographer-ros.readthedocs.io/en/latest/index.html) and A-LOAM (3D LiDAR) (~~https://github.com/HKUST-Aerial-Robotics/A-LOAM~~) (Update 21 Feb: please consider referring this repo https://github.com/nuslde/aloam_lidar_odom_result_generate to do 3D SLAM task. Modifications include Ubuntu 20.04 support and directly generating odometry file for EVO evaluation.) to run the corresponding rosbag.
    a. 2D LiDAR SLAM: rosbags `<2dlidar.bag>`
    b. 3D LiDAR SLAM: rosbags `<3dlidar.bag>`

Ground truth of a is stored in the 2dlidar.bag itself under in topic "/odom"

Refer to the following command to use EVO: "evo_ape bag < bag_file_name > / tf: odom.base_footprint / tf: map.base_footprint"

Ground truth of b is stored in the 00.txt (KITTI ground truth format)

After running the algorithms and their evaluation you should get some results like (refer to the command in the figure to use EVO):



A-LOAM EVO usage

2. (**BONUS**) Choose one SLAM algorithm to run the `<have_fun.bag>`. There is no limit on what SLAM algorithm you use, your goal is minimizing the RMSE on the given sequence. Therefore, you are encouraged to use multi-sensor fusion SLAM algorithms like VINS, FAST-LIO etc.

In your report:

- Briefly describe the SLAM1 algorithm you choose to use
- Describe the detailed process of running the SLAM algorithms, give a screenshot of your algorithm running in rviz
- Highlight your modification/tuning done on the algorithms to achieve better results
- Show the Absolute RMSE, as well as the plots generated by the EVO tool
- Discuss the drawbacks/failures in your tests, provide some analysis and propose possible solutions (illustrate with figures and provide your hypostasis)

Submitting your completed Homework Assignment:

You are expected to summarise your observations, assumptions, and your own implementation details in a 5-page report (the maximum length of the report is 5 pages, while there is no page limit for appendices, if you wish to attach more of your results).

Do note that you should practice good code styles in your own code, including proper naming conventions, informative documentations, etc (please refer to the Google Python Style Guide).

Late submission will not be allowed.

Generate a non-password-protected zipfile of this folder and upload it to CANVAS – under Assignment 2. We will use the latest version, regardless of who uploads. Name of Zipfile: "YourHomeWorkGroupNumber_Homework2.zip" (e.g., 43_Homework2.zip - for group 43)

1. Report details:
    a. Homework Group number
    b. Matric numbers of group members (e.g. number starting with A0..)
    c. Maximum number of **5 pages for the report. We are not limited to the pages of the appendix.**
2. All code and results are to be stored in the following three folders
    - Task 1: main.py
    - Task 2: main.py
    - Task 3 & bonus: Images generated by "evo_ape" command
3. Evaluation of tasks will be based on
    a. Performance/Accuracy
    b. Technical explanations
    c. Code executability