ME5413: Autonomous Mobile Robot

Homework 2: Localisation

AY2024/25-Sem 2

Due date: 20 March 2025 - 2359 (Thurs, Week 9)

Important note: Task 1 should be done in Python and Task 2 will involve ROS and Cartographer (a 3rd party SLAM system). Late submission be penalised. First hour 5%, and every 24 hours after that 33.3% of the grade. No Plagiarism is allowed.

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 algorithm. 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.

Requirements: *Python 3.8 and higher and ROS Noetic*

Task 1 and 2 data download link HERE:
https://drive.google.com/drive/folders/1Ir62WhCqI4Mxln7QZQjFagVv8OcOGg5m

Task 1: Writing the ICP Algorithm for Point Cloud Registration

Note:

- Before proceeding with task 1, please find your data number corresponding to your name and student id in `id_list.pdf`. For task 1, please run your code on your own dataset from `student_data/student_data_{YOUR_DATA_NUMBER}`, which are 2 .ply files. The bunny1.ply is used as the reference point cloud in task1 (fixed), and bunny2.ply is the point cloud you need to transform and match bunny1.ply.
- You are also provided a template code named `<task1.py>`. Please install the dependencies imported at the beginning of code using pip3 or anaconda first.
- **You should submit the completed code with comment, as well as the report that contains your observation, intermediate result (both visualization and transformation), analysis, and conclusion.**

Task 1.1:

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

Please finish all code with TODO label in functions `icp_core` and `solve_icp_with_known_correspondence`.

Task 1.2

In real world point cloud matching problems, the correspondence is usually **unknown**, you need to come up a method to estimate the correspondence and iteratively update the transform until it matches some convergence criteria.

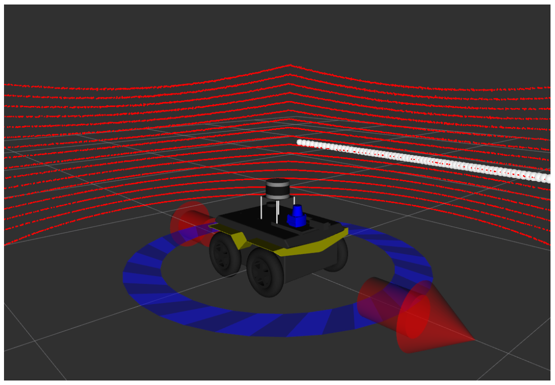
Please finish all code with TODO label in the function \`solve_icp_without_known_correspondence\`.

In your report:

- Briefly describe the algorithm implemented
- Discuss the quantitative results using the following: final accumulative transformation matrix, mean error, time cost
- Show the qualitative results of the point cloud matching, with intermediate results during iterations

Task 2: Running a SLAM Algorithm in ROS

In task 2, a commonly available SLAM system used in both academic and industry settings – **Google Cartographer** (https://google-cartographer-ros.readthedocs.io/en/latest/index.html) will be used to build a map from a robot's sensor data given as a ROS bag file. The performance will then be evaluated using the convenient **evo tool** (Available at: https://github.com/MichaelGrupp/evo).



The simulated robot is a "Jackal" Unmanned Ground Vehicle from Clearpath Robotics with an IMU, a 2D LiDAR (Sick TIM551), and a 3D LiDAR (Velodyne VLP16). The Jackal is then driven in a loop around a test track in a Gazebo simulation and ROS. Transformation between the robot and sensors, and the odometry of the robot is recorded alongside the sensor information.

2.1 Installing tools (Cartographer and evo):

Install Cartographer on your Ubuntu 20.04 system together with ROS Noetic. Detailed instructions can be found in the "Compiling Cartographer ROS" section in Cartographer's documentation (https://google-cartographer-ros.readthedocs.io/en/latest/compilation.html).

Next, install evo by following the instructions on evo's Github page (https://github.com/MichaelGrupp/evo?tab=readme-ov-file#installation--upgrade).

2.2 Building the Map:

The sensor data recorded by the Jackal robot in the Gazebo simulation is in the ROS bag file "task2.bag". Explore data recorded in the bag by using the inbuilt ROS utilities: `rviz` (for visualizing sensor data), `rosbag` (for playing back recorded bag data, https://wiki.ros.org/rosbag/Commandline) and `rostopic` (for displaying data streams, https://wiki.ros.org/rostopic#rostopic_command-line_tool).

Build a 2D map using Cartographer and the data given in "task2.bag". Both the 2D and 3D LiDAR data from the ROS bag can be used.

Although Cartographer gives acceptable SLAM performance using the default values, tuning the parameters used in the algorithm will usually yield better results. A helpful walkthrough to the parameters and tuning them can be found here: (https://google-cartographer-ros.readthedocs.io/en/latest/algo_walkthrough.html)

Hint: When using `rosbag play` *to play a recorded bag in a simulation, use the* `--clock` *flag to run in simulated time.*

Hint: Run `rosbag record -a` *during your best mapping run to generate a ROS bag file to be used in evaluating the performance in task 2.4.*

Note: The ground truth topic should not be used when doing mapping, it should only be used for evaluating the SLAM performance.

2.3 Saving the Map:

Using the `map_server` package from the standard ROS installation, save the generated map using the `map_saver` utility. Please name your map "task2" when saving your map so that "task2.pgm" and "task2.yaml" will be generated as the map file and the descriptor file respectively.

Usage: `rosrun map_server map_saver [--occ <threshold_occupied>] [--free <threshold_free>] [-f <mapname>] map:=/your/costmap/topic`

Example: `rosrun map_server map_saver --occ 70 --free 30 -f task2 map:=/map`

2.4 Evaluating the Performance:

Finally, the results of the running the SLAM system should be evaluated using evo. The performance of the map should be assessed based on the Absolute Root Mean Square Error (RMSE) over the entire sequence.

The ground truth of the Jackal's position as reported from the simulation is in the topic "/ground_truth" in the ROS bag file "task2.bag". Compare the trajectory generated by your tuned Cartographer with the ground truth to find out how well your SLAM did.

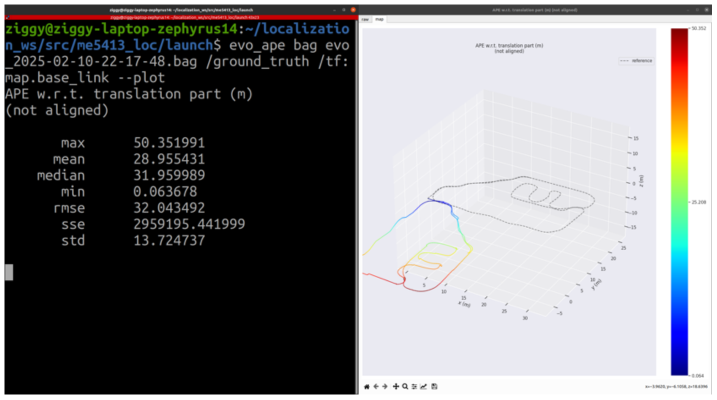
Usage: `evo_ape bag <bag_file.name> /ground/truth/topic /tf:from_map.to_robot --plot`

Example: `evo_ape bag cartographer.bag /ground_truth /tf:map.base_link --plot`

The results will be both printed out in the command line and displayed in a GUI window.

Hint: If you have having issues where the SLAM's generated trajectory and the ground truth looks similar but does not line up, think if the world frame is same as the map frame.

Hint: Your `evo_ape` *output format should look something like this:*



```
ziggy@ziggy-laptop-zephyrus14:~/localizatio
n_ws/src/me5413_loc/launch$ evo_ape bag evo
_2025-02-10-22-17-48.bag /ground_truth /tf:
map.base_link --plot
APE w.r.t. translation part (m)
(not aligned)

       max      50.351991
      mean      28.955431
    median      31.959989
       min      0.063678
      rmse      32.043492
       sse      2959195.441999
       std      13.724737
```

2.5 In your report:

- Briefly describe the process of running Cartographer SLAM on the recorded ROS bag, specify whether 2D and/or 3D LiDAR data was used
- Highlight modifications/tunings done on Cartographer to achieve better results
- Discuss the quantitative results using the absolute RSME, as well as plots generated by the evo tool
- Discuss the qualitative results using the generated map file

Submitting your completed Homework Assignment:

Generate a zipfile of the following folder structure and upload it to CANVAS – under Assignment 2. *We will only use the latest version.*

Name of Zipfile: "YourNusNetID_Homework2.zip" (e.g. E123456_Homework2.zip)

Submission Details:

Important – All code, comments and report must be in English. Text in other languages will not be read.

1.	Report	1. Name 2. Matric number (i.e. number starting with A0..) 3. Maximum number of **5 pages** for the report not including the front cover page (+2 pages for additional results, code snippets and observations) 4. Font: Times New Roman, 12pt, Single Spaced 5. In PDF format (with similarity score < 20%)
2	Folder: task_1	1. Python Code (task1.py file)
3	Folder: task_2	1. Map file (task2.pgm) generated by map_saver 2. Descriptor file (task2.yaml) generated by map_saver