Coding exercise 3 for CS 498 GC

- ☐ This homework consists of a written and coding part. Both must be submitted individually on Gradescope.
- ☐ The coding part must be developed in python with ROS2. When submitting this section, only upload the python file *coding_ex3.py* (see attached template).
- ☐ The report part must be submitted in PDF format. When submitting this file, Gradescope will ask you to select the pages where each question is. Please do it precisely.

Learning outcomes

- ☐ Understand the foundations of LiDAR-based SLAM
- ☐ Process data from a real robot, plot and analyze
- ☐ Understand how to interpret data from active radiation based sensors, such as scanning lasers (LIDARs).

Total points: 200

Description

You are provided with data collected from terrasentia robot. It contains LiDAR measurements (/terrasentia/scan) and an EKF-estimated trajectory (/terrasentia/ekf).

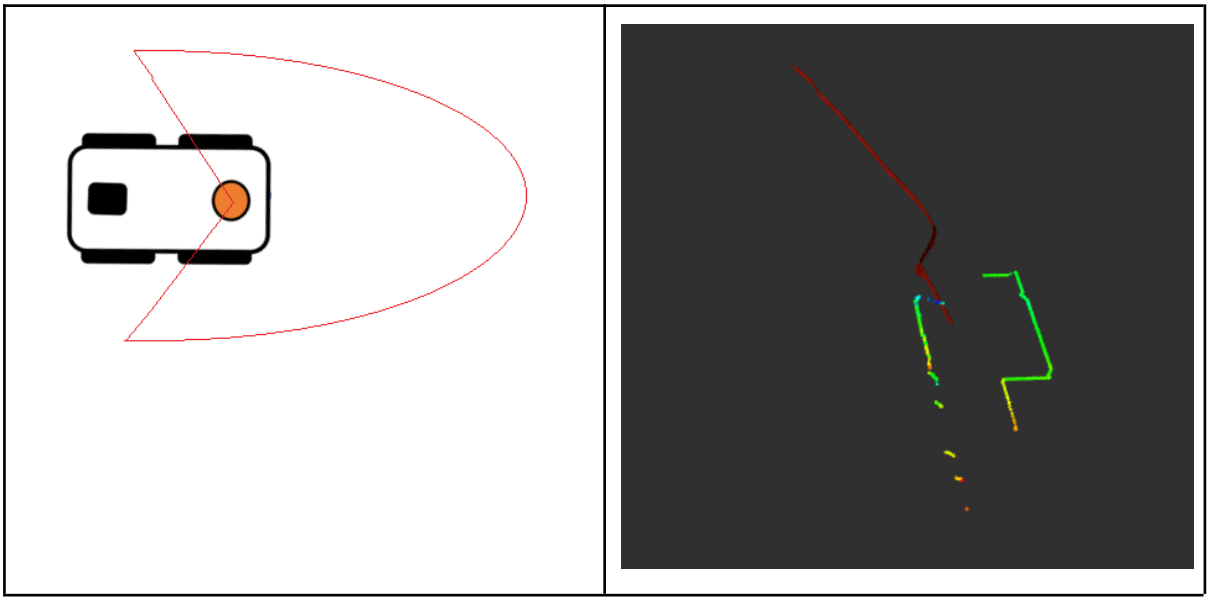

Figure. Robot's trajectory and LiDAR measurements

1. Play the rosbag file '*lidar'* and visualize the robot trajectory and the LiDAR measurements on rviz2. Add screenshots to your report.

2. Write code in Python to find and fit lines and corners from LIDAR data, using one of the line fitting algorithms discussed in chapter 4.7.2 of Siegwart. Test your algorithm on the following data. Add the results to your report (you don't need the ros template for this question).

```
import numpy as np
# test data in polar coordinates
rho_test = np.array([[10, 11, 11.7, 13, 14, 15, 16, 17, 17, 17, 16.5,
17, 17, 16, 14.5, 14, 13]]).T
n = rho_test.shape[0]
theta_test = (math.pi/180)*np.linspace(0, 85, n).reshape(-1,1)
```

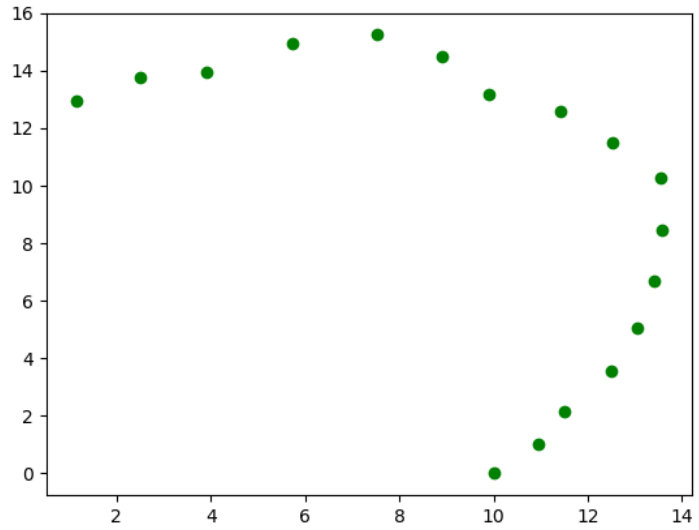

Figure. Simple data to test the line fitting algorithm

3. Use the function created in the previous question to find and fit lines and corners from the LiDAR data in the rosbag file. Visualize the lines on rviz2. Add screenshots to your report.

- The LiDar data is contained in the topic /terrasentia/scan.
- The function callback_scan() in the attached template is already reading the ranges. These ranges are stored in a vector whose first value corresponds to -45 degrees, and the last value corresponds to 225 degrees (the total field of fiew is 270 degrees).
- You may need to convert the LiDAR measurements from polar coordinates to rectangular coordinates, depending on the algorithm you want to implement.
- You can use line markers to draw lines on rviz. See example in the function *draw_line_example_callback()*.

4. **Mapping**. Use the line fitting algorithm and the trajectory of the robot to create a complete geometric map of the corridor. Add screenshots to your report.

- The trajectory of the robot is given by the topic /terrasentia/ekf. You need to subscribe to this topic to get the position and orientation.
- Remember that the LiDAR measurements are given with respect to the robot coordinate frame. You will need to convert the estimated lines to a global reference frame (e.g. odom) for mapping.

5. **SLAM**. Use Gmapping to create a map based on SLAM (see tutorial below). Compare this map with the one you obtained with your implementation. Explain the differences you find.

Sample output

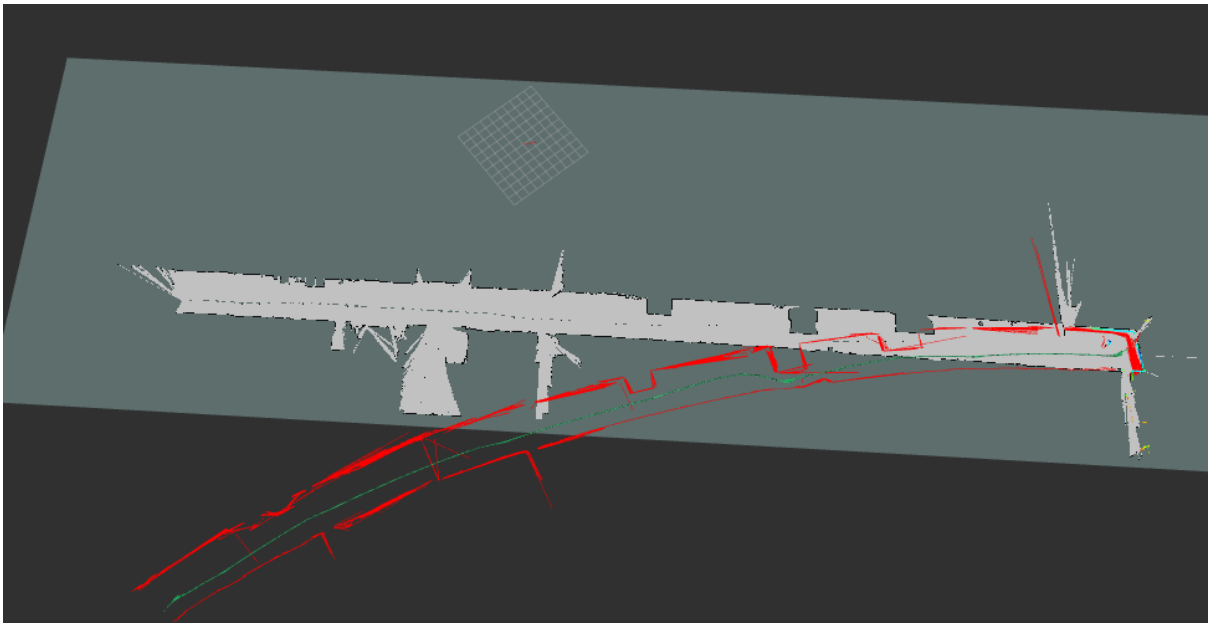

Figure. Maps created by gmapping (gray) and the line-fitting algorithm (red).

Appendix
Tutorial 1. Building Gmapping in your ROS workspace

This tutorial is intended for ROS2 on Ubuntu.

You need to create a ROS2 workspace if you do not have one yet.
Then, you just have to download this project to the src folder and build with colcon tools, as follows:

```
>> cd ros2_ws
>> cd src
>> git clone https://github.com/Project-MANAS/slam_gmapping.git
>> cd ..
>> colcon build --packages-select openslam_gmapping
>> colcon build --packages-select slam_gmapping
```

Run gmapping

```
>> source install/setup.bash
>> ros2 run slam_gmapping slam_gmapping scan:=/terrasentia/scan

in other terminal:
>> ros2 bag play lidar

Gmapping will publish a topic called /map, that you can visualize on rviz2
```