ECED 3901

Laboratory 4

LiDARs and Simultaneous localization and mapping (SLAM)

Date: January 28, 2026

Report Due: none

Learning outcomes:

We will establish a virtual world with enclosed walls and features. Then we navigate the emulated robot within that world to create a map of the environment. Laboratory 4 will also focus on using ROS2 launch files for simplifying execution of the mapping nodes and the motion nodes. The objectives that will be accomplished in this lab are the following:

Prerequisites:

- Completion of Laboratories 1 to 3.
- Completion of Design Task 1.

Outcomes:

a) Explore ROS2's laserscan messages to gather the LiDAR data
b) Perform simultaneous localization and mapping (SLAM) and implement these algorithms with teleoperation.
c) Capture the resulting map from SLAM.

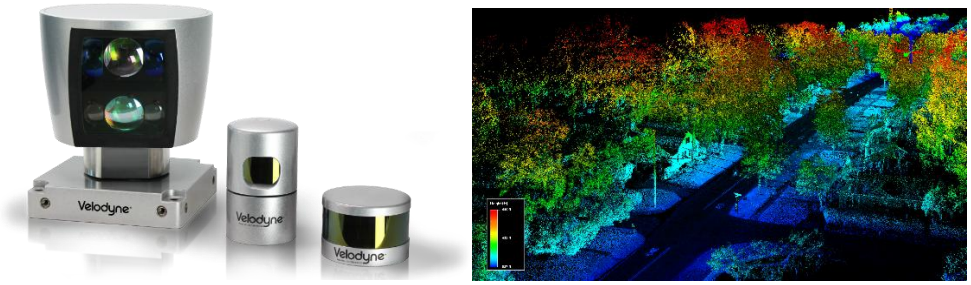
Part A – RViz to visualize LaserScan messages.

A1. Background:

- LiDAR stands for "Light Detection And Ranging" or "Laser Imaging, Detection And Ranging". They are range finding devices that launch pulses of light and illuminate a target. By measuring the elapsed time, or phase shift, or triangulation associated with the reflected return signal, they can determine distance.

- Commercially available LiDARs are used for autonomous vehicles and cars. For example, the Velodyne series below have multiple individual laser/detector combos (from 16 to 128) that rotate providing millions of data points per second. These can be used to generate depth maps like shown below. Youtube is filled with numerous videos of the Velodyne LiDARs for the curious student.

- 3D: 3-dimensional LiDARs can be expensive ranging from $5,000 to well over $100,000. 2D: 2-dimensional or "pencil scanning" LiDARs are much cheaper at $100-$1,000; for example, the RPLidar or YDLidar on the ECED3901 robot. 1D: 1-dimensional range finders are even more cost-effective at $15 each, e.g. VL53L1X.



Copyright © Velodyne and ZDNet

A2. Laser messages in ROS2:

ROS2 can handle most Laser range finder data. Laser range finding data is so fundamental to many robotics projects and algorithms, it was necessary to establish its own standardized message called "LaserScan". Please refer to the link below to inspect ROS' API on the LaserScan message details. While described for ROS version 1, the message format is the same in ROS2.

http://docs.ros.org/melodic/api/sensor_msgs/html/msg/LaserScan.html

Important information for our case (2D - YDLidar), is the "ranges" data array and the necessary parameters for understanding the array like angle_min, angle_max, angle_increment, scan_time, time_increment, etc.

A3. Simulation and the emulated LiDAR

You have your own physical LiDAR to experiment with, but running your robot takes setup time, consumes battery, and you may be limited in space and maps available. Fortunately, you have installed Ubuntu, ROS2 and the necessary packages that have an emulated LiDAR. The emulated LIDAR very closely resembles that on the ECED robots and your simulated code results will link very well to reality.

We start by spinning up a virtual world with a robot within.

1. Start your ECED3901 Robot or Virtual Machine.

2. Open a new terminal.

3. Bring up Gazebo and the lab4 world; run the following:

```
$ cd ~/ros2_ws/src/eced3901/worlds
$ gazebo eced3901_lab4.world
```

4. Open another terminal.

5. Run the following, to see verbose information on the topics:

```
$ ros2 topic list -v
```

   You should observe that the /scan topic is publishing a /LaserScan message from the *sensor_msgs* library. Similarly, /odom is publishing an /Odometry message from the *nav_msgs* library.

6. Run the following to view the /scan message, and the ranges data contained within.

```
$ ros2 topic echo /scan --field ranges
```

   Hard to visualize?

7. Stop the echo topic and Stop the Gazebo world. Press “ctrl + c” in both terminals to stop this section. Later, you can explore the LaserScan message elements by changing the field of “ranges”, to for instance “header”.

8. Let's load Gazebo and RViz using launch files. RViz2 will allow us to visualize range data in a manner that is more useful for debugging.

```
$ ros2 launch eced3901 lab4a.launch.py
```

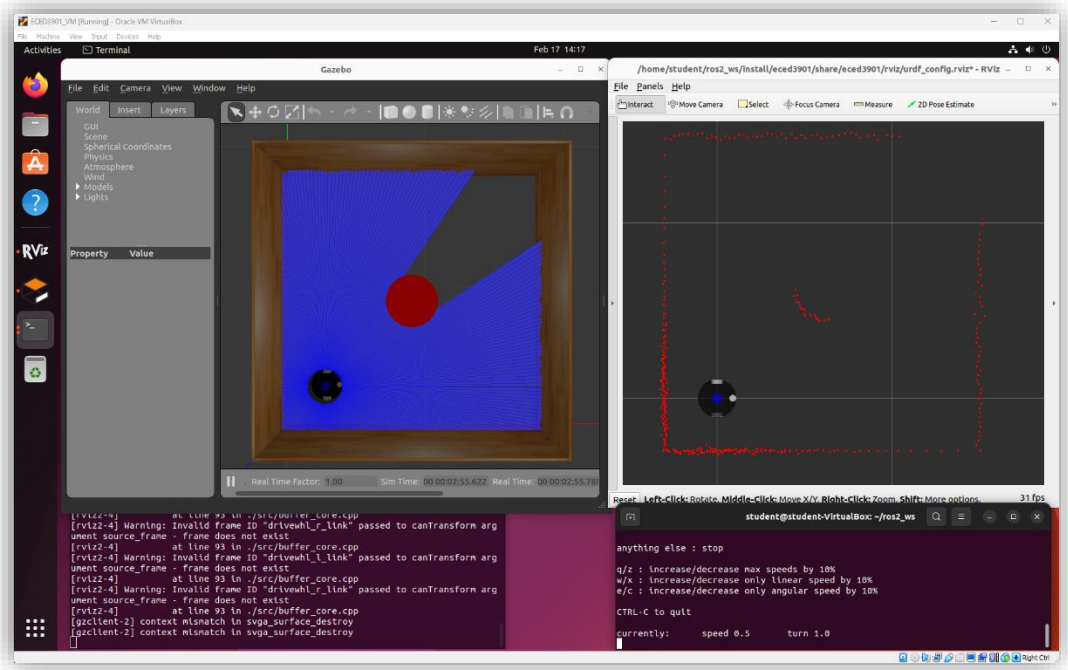
9. We have to add the LiDAR topic. In Rviz:
    a. Click the "Add" button.
    b. Move to the "By topic" tab.
    c. Click the "LaserScan" topic, then click OK.

    The LiDAR returns should reflect the WORLD environment loaded in Gazebo.

10. Open a new terminal and launch the teleop package.

```
$ ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

11. Drive around. Observe the range finder data, and how it behaves when interacting with the virtual world provided by Gazebo. Take note of the shadowing, spacing/density of return hits, etc. You should see something like below before you move on.



Part B – Making a map from LaserScan data.

Visualizing live sensor data is fine, and direct sensory data can be used for reactive robotic control systems, but how do you make a map from this ranging data to move toward a more deliberative approach? In other words, how does the robot know if the laser returns are from the walls/map or from an obstacle that it should avoid? At least for this course, we are going to help the robot by providing it with knowledge of the map. BUT… how do we make such a map?

The lectures introduced simultaneous localization and mapping, or SLAM, algorithms, which are used to create maps while also determining where the robot is in map at the same time, hence the name. Here we will put SLAM into practice in simulation. Your next Design Task will have you implement SLAM on the real robots. For now, let us begin with simulation.

1. Close all the windows, and halt all code execution running from Part A.

2. Make sure you have 3 terminals up and running. Terminal 1 will be used for launching Gazebo and your environment, terminal 2 will be used for running SLAM algorithms, and terminal 3 will be used to teleoperate the robot.

3. In a terminal, start Gazebo loaded with the Lab4 World by calling a different launch file. This launch file brings of "Nav2" and the gmapping SLAM algorithm.

```
$ ros2 launch eced3901 lab4b.launch.py slam:=True
```

4. In another terminal, start the teleop package.

```
$ ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

Start driving around and watch how the map is made! As you drive around the map, more areas will be covered by the laser rays, resulting in empty voxels or hits/returns. Therefore, the occupancy grid will become more complete as your robot travels, reflecting the actual physical environment as shown below.

5. When you're satisfied with the map, open a new terminal, and run the following command to save your map.

```
$ cd ~/ros2_ws/src/eced3901/maps

$ ros2 run nav2_map_server map_saver_cli -f lab4_map
```

6. Stop the launch file (Gazebo, RViz2, Nav2 stack) and the teleop terminals. In the maps folder, you should see “lab4_map.pgm” and “lab4_map.yaml”.

7. For the remainder of the Laboratory, apply these launch files on your Robot and **map a real environment** as you will need that capability for DT2.