Summer Research: SLAM Algorithm Comparison and Ouster LiDAR Setup

Ilana Zane, Ming-Hung Yen
Stevens Institute of Technology
Hoboken, NJ

September 17, 2022

Abstract

Our summer work consisted of two parts: learning about path planning and navigation through ROS and gaining experience in using robotics hardware. This summer we learned how to use RViz and Gazebo for robot simulation and visualization, and worked on setting up an Ouster Lidar, with an NVIDIA Jetson AGX Xavier to collect metadata and create a point cloud of its surroundings.

1 Introduction

In recent years, mobile robotics have become increasingly more popular and researchers are exploring new applications of robots in increasingly more complex environments. A fundamental problem in robotics is the process of having the robot acquire a map of its environment while simultaneously localizing itself relative to this map, i.e. SLAM (Simultaneous Localization And Mapping). SLAM is an important step in making a robot autonomous, and there are currently several different methods of performing SLAM. With many different methods available, it is worth exploring them and figuring out which ones may perform better or worse.

2 Software Setup

2.1 ROS Setup

Preliminary steps that we had to take before starting our project included updating the Ubuntu software and downloading the ROS setup on our desktops. In order to use ROS Noetic, we first had to update our Ubuntu software from Ubuntu 18.0 to 20.0. Following instructions from the ROS wiki site[1], we downloaded the full desktop version of Noetic. Once we configured our setup and confirmed that it was downloaded and working, we continued to follow the ROS wiki tutorials in order to familiarize ourselves with nodes, topics, subscribers etc., and the file layout system.

2.2 Navigation and Simulation Setup

Before testing different SLAM algorithms we tested the movebase node from ROS in order to learn the functions and become familiar with the entire navigation stack in ROS. We consistently tested the algorithms with the TurtleBot3 package [2], specifically using the TurtleBot3 Burger model. In order to test the different SLAM algorithms we used the provided TurtleBot3 house simulation that consisted of multiple walls and static obstacles throughout, such as tables. A snapshot of the simulation world can be seen in Figure 1.

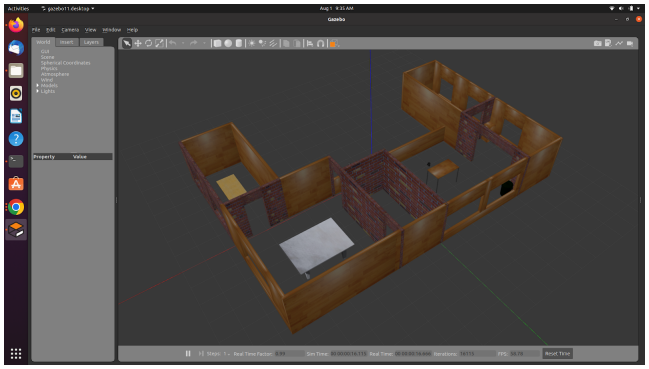

Figure 1: House Simulation in Gazebo

The TurtleBot3 Burger has a size of 138mm x 178mm x 192mm (LxWxH), a maximum translational velocity of 0.22 m/s, and a maximum rotational velocity of 2.84 rad/s (162.72 deg/s) [2]. A diagram of the model is provided in Figure 2.

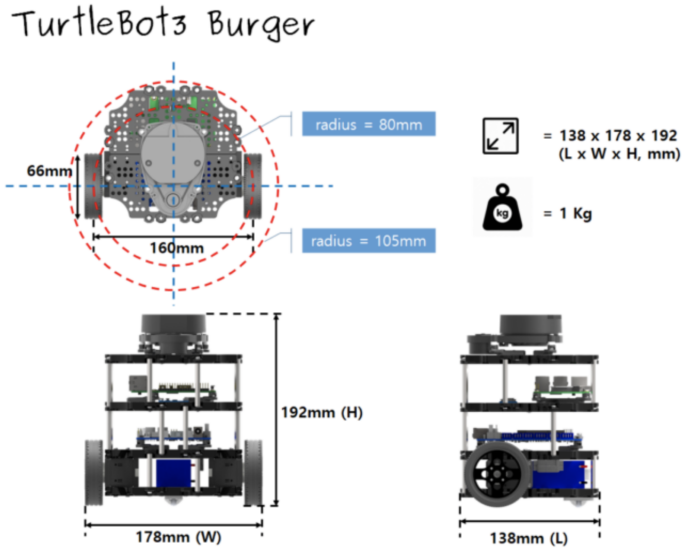

Figure 2: TurtleBot3 Burger Specifications

2.3 SLAM Experiment

In order to start running the SLAM algorithms, we downloaded necessary code from the official Turtlebot3 Github that included all simulation and launch files [3]. We used the following commands (in two separate terminals) to launch the simulation world in Gazebo and then RViz:

```
roslaunch turtlebot3_gazebo house.launch
roslaunch turtlebot3_slam turtlbot3_slam.launch slam_methods:=gmapping
```

Where **gmapping** can be replaced with either **hector** or **karto** in order to use different SLAM methods. Once we were able to see the simulation world in Gazebo and the robots perception of its environment in RViz, we were ready to create a map. To operate the robot around the environment we used the following command to launch the teleop node:

```
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

The teleop node allows the user to control the movements of the robot around its environmment using the "w,a,s,d,x" keys on their keyboard as seen in Figure 3. The w and x keys increase and decrease, respectively, the linear velocity of the robot, while the a and$d$ keys increase and decrease, respectively, the rotational velocity of the robot. The s key stops robot movement. With these combinations we were able to move around the room and create the following maps for each SLAM algorithm.

```
$ export TURTLEBOT3_MODEL=burger
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

 Control Your TurtleBot3!
 ---------------------------
 Moving around:
        w
   a    s    d
        x

 w/x : increase/decrease linear velocity
 a/d : increase/decrease angular velocity
 space key, s : force stop

 CTRL-C to quit
```

Figure 3: Teleop Node [2]

2.3.1 Gmapping

Gmapping is an algorithm created to increase the performance of Rao-Blackwellized particle fillers to solve the SLAM problem [4]. We also explore mapping with Karto-SLAM[5], which uses "sparse pose adjustment to solve the problem of matrix direct solution in nonlinear optimization"[6] and Hector SLAM[7] which uses the "Gauss-Newton method to solve the problem of scanning matching ... high precision LiDAR is required"[6].

2.4 SLAM Results

Testing Gmapping, Karto and Hector SLAM produced the following maps as shown in Figure 4. The first image shows the results from Gmapping, second is Karto, and third is Hector SLAM. Based on these maps, we can observe that Gmapping produced a map that was closest to the actual layout of the simulated house. Karto produced similarly successful results, but with some noise outside of the walls. Hector produced a map that was sub-optimal and did not accurately present the initial configuration of the house simulation. We think this is because Hector SLAM relies solely on LiDAR information and the defualt configurations of the simulation TurtleBot3 might not have been optimal enough. Karto SLAM produces decent results with slight noise in certain areas of the house. Overall, Gmapping produced the best results. Perhaps, it is best to use a SLAM algorithm that relies more on odometry data than LiDAR data for areas with narrow hallways and several obstacles.

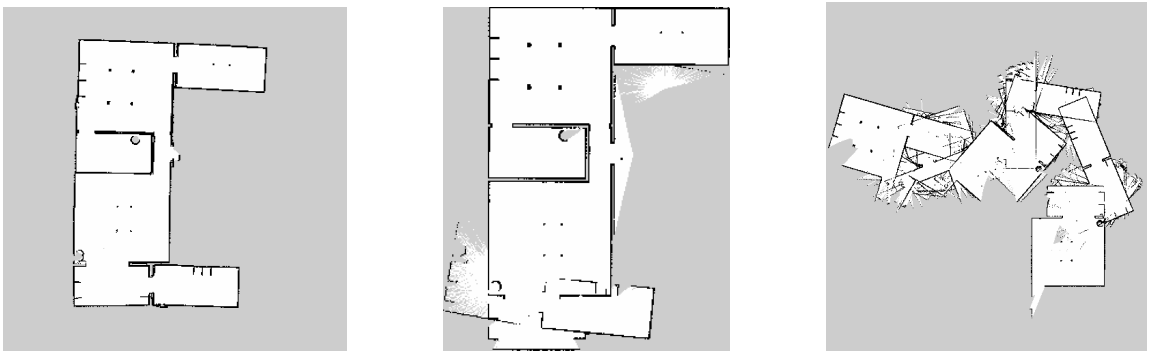

Figure 4: SLAM Mapping Results

3 Ouster Lidar Experiment

The second part of our project consisted of learning how to configure the Lidar sensor to read data from its surroundings and create a point cloud map to visualize this data. We used an Ouster Lidar with hardware version: Gen1 PN:840-101XXX and firmware version: v2.3.x provided by the Ouster Company [8]. After downloading the appropriate firmware to be compatible with our operating system, we had to connect the Lidar to the NVIDIA Jetson AGX Xavier. This process included closely following the hardware and API manuals [9][10] to connect the two entities. Once a connection was made between the Lidar and the Xavier Computer, we were able to open a local host page (**os1-991910000147.local**) consisting of the hardware version and unique serial number that showed a dashboard consisting of the Lidar configuration information and diagnostic information. Following the Ouster in ROS tutorial[11] we were able to launch the Lidar point cloud node in RViz with the following command in one terminal window:

```
roslaunch ouster_ros ouster.launch sensor_hostname:=os1-991910000147.local
metadata:=/home/usr/catkin_ws/myworkspace/src/ouster_example/ouster_ros/metadata.json viz:=true
```

Once the Lidar node was launched it produced the following point cloud in RViz, seen in Figure 5. The image seen is from the lab room, where we can observer two walls in the top right corner as well as the chairs and desk in the room.

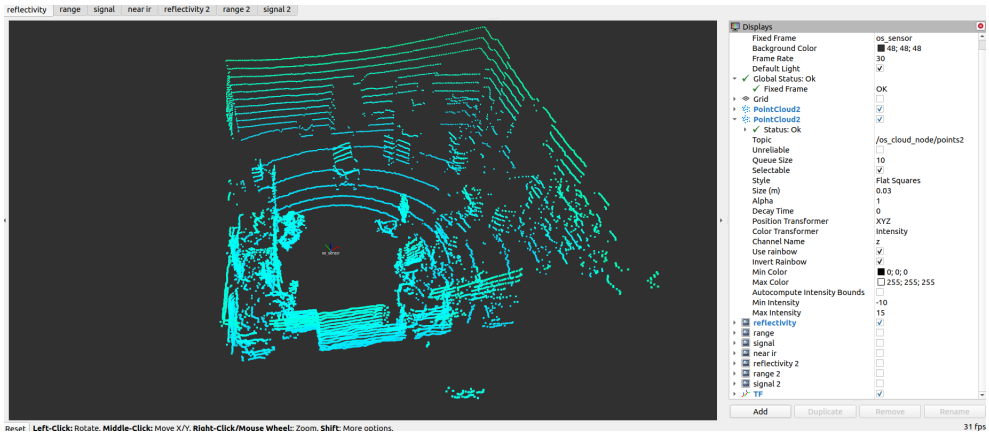

Figure 5: Output from Lidar Sensor

References

[1] "Noetic/installation/ubuntu - ros wiki." (), [Online]. Available: `https://wiki.ros.org/noetic/Installation/Ubuntu` (visited on 08/01/2022).

[2] "Robotis e-manual." (), [Online]. Available: `https://emanual.robotis.com/docs/en/platform/turtlebot3/basic_operation/#basic-operation` (visited on 08/01/2022).

[3] "Robotis-git/turtlebot3$_s$*imulations*." (), [Online]. Available: `https://github.com/ROBOTIS-GIT/turtlebot3_simulations` (visited on 08/01/2022).

[4] W. B. Giorgio Grisetti Cyrill Stachniss, "Improved techniques for grid mapping with rao-blackwellized particle filters," vol. Volume 23, 2007, pp. 34–36. DOI: `10.1109/TRO.2006.889486`.

[5] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2d mapping," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 22–29. DOI: `10.1109/IROS.2010.5649043`.

[6] G. Jiang, L. Yin, S. Jin, C. Tian, X. Ma, and Y. Ou, "A simultaneous localization and mapping (slam) framework for 2.5d map building based on low-cost lidar and vision fusion," *Applied Sciences*, vol. 9, no. 10, 2019, ISSN: 2076-3417. DOI: 10.3390/app9102105. [Online]. Available: https://www.mdpi.com/2076-3417/9/10/2105.

[7] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2011, pp. 155–160. DOI: 10.1109/SSRR.2011.6106777.

[8] "Ouster download center." (), [Online]. Available: https://ouster.com/downloads/ (visited on 08/01/2022).

[9] "Ouster download center." (), [Online]. Available: https://data.ouster.io/downloads/hardware-user-manual/hardware-user-manual-gen1-os1.pdf (visited on 08/01/2022).

[10] "Ouster download center." (), [Online]. Available: https://data.ouster.io/downloads/tcp-http-api-manual/api-manual-v2.3.0.pdf (visited on 08/01/2022).

[11] "Ouster sensor sdk." (), [Online]. Available: https://static.ouster.dev/sdk-docs/ros/index.html#building-ros-driver (visited on 08/01/2022).