Generating a Map of the Environment With a GMapping and Hector Mapping SLAM Robot

Bill Wang[1,2], Yancheng Zhu[2], Sean Andersson[2]

Bergen County Technical Schools - Teterboro, 504 US-46, Teterboro, NJ[1]; Boston University, Boston, MA[2]

Introduction

To navigate through an environment, a robot needs to recognize its surroundings and know its location. In an unknown environment, the robot does not have either, and needs to generate a map of its surroundings while simultaneously determining its location. However, to generate the map, the robot requires its location, but to acquire its location, the robot needs the map. Simultaneous Localization And Mapping (SLAM) algorithms tackle this problem by approximating the solution, allowing robots to navigate through new environments more easily. This project focuses on the GMapping and Hector Mapping algorithm.

The goal is to create a robot capable of generating a map using SLAM algorithms. The GMapping algorithm requires laser scan and odometry data to generate a 2D grid map of the environment and use that to approximate the robot's new location. Laser scan data is a 2D plane of points that represent the distance of the sensor's surroundings. Odometry is an estimate of a robot's change in position over time, obtained by wheel encoders, accelerometers, or in this case, motion tracking. However, odometry data tends to drift over time, hence why SLAM is required to more precisely determine the robot's position. Hector Mapping, on the other hand, only requires laser scan data.

Robot Operating System (ROS) is an open source software framework that serves as a middleman between a computer and the robot. It is built on nodes that can publish and receive data and process it, such as a SLAM node receiving sensor data and publishing a map.

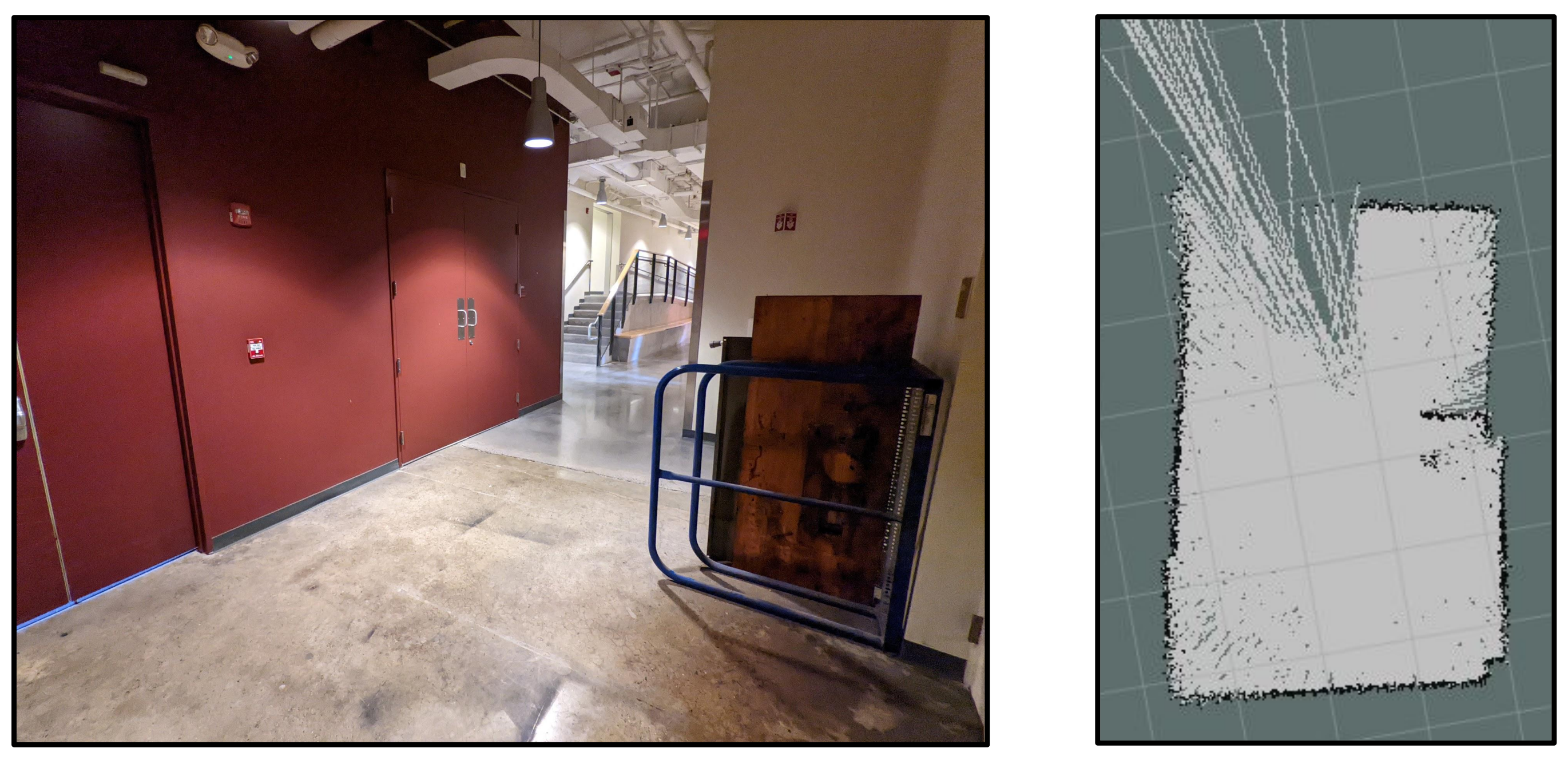


Figure 2: A Hector map of a room compared to ground truth, mapped by manually moving the LIDAR around

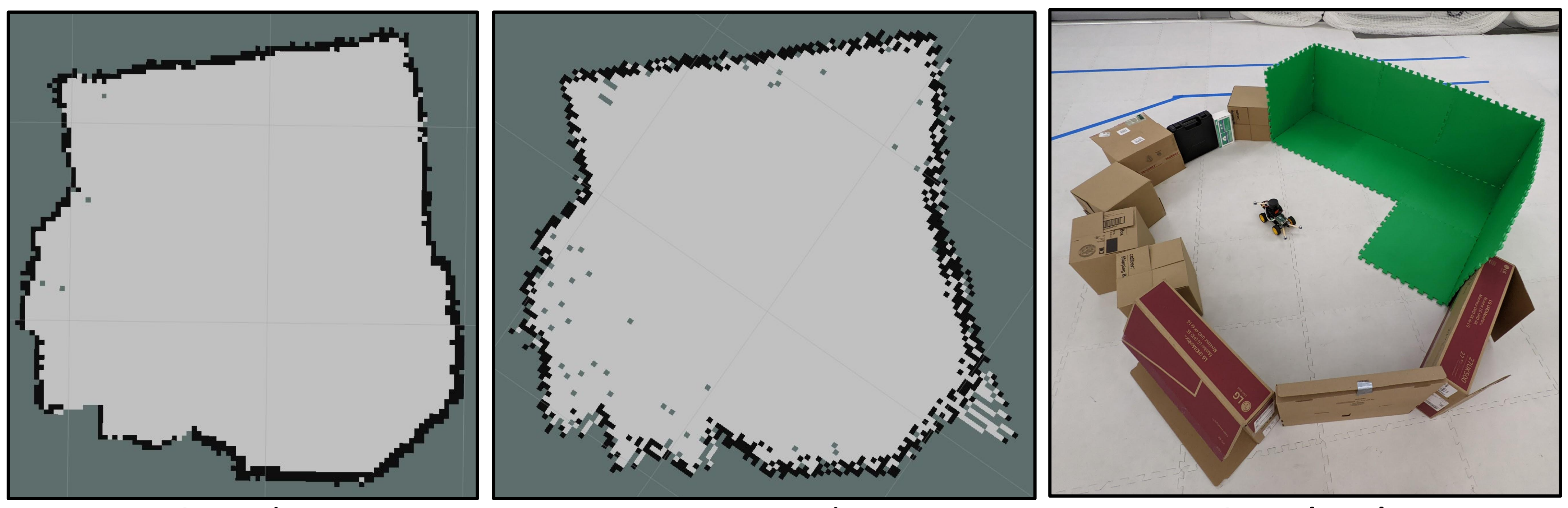


Figure 1: Comparison of GMapping and Hector mapping outputs

Methods

ROS is used to run the SLAM algorithms. A laptop and the Raspberry Pi were connected via Husarnet, a ROS compatible peer-to-peer VPN, which allowed one ROS session to run across both computers. The Freenove car body has four motion capture markers, two batteries, and the Raspberry Pi attached to the frame. The Raspberry Pi ran a server that allowed a smartphone to remotely control the car and a node to run the sensor. The laptop ran the SLAM algorithms, a few nodes to allow compatibility, and the VRPN client, which communicates with OptiTrack (motion capture). The compatibility nodes included ones that converted pointcloud data from the LIDAR to laser scan data, converted the pose output from OptiTrack to odometry data, and published transform information. See figure 3 for the software setup and figure 4 for the robot setup.

Software

- Robot Operating System (Kinetic Kame Distribution)
- Ubuntu 16.04 Mate (Raspberry Pi)
- Ubuntu 16.04 (Laptop)
- Freenove Smart Car library
- Motive (OptiTrack software)
- Teraranger ROS library
- Sweep ROS library
- Hector Mapping
- GMapping
- VRPN Client
- Husarnet

Hardware

- Laptop
- Smartphone
- Scanse Sweep LIDAR
- Freenove 4WD Smart Car for Raspberry Pi
- Raspberry Pi 3 Model B
- OptiTrack Motion capture system

Figure 3: Software Setup

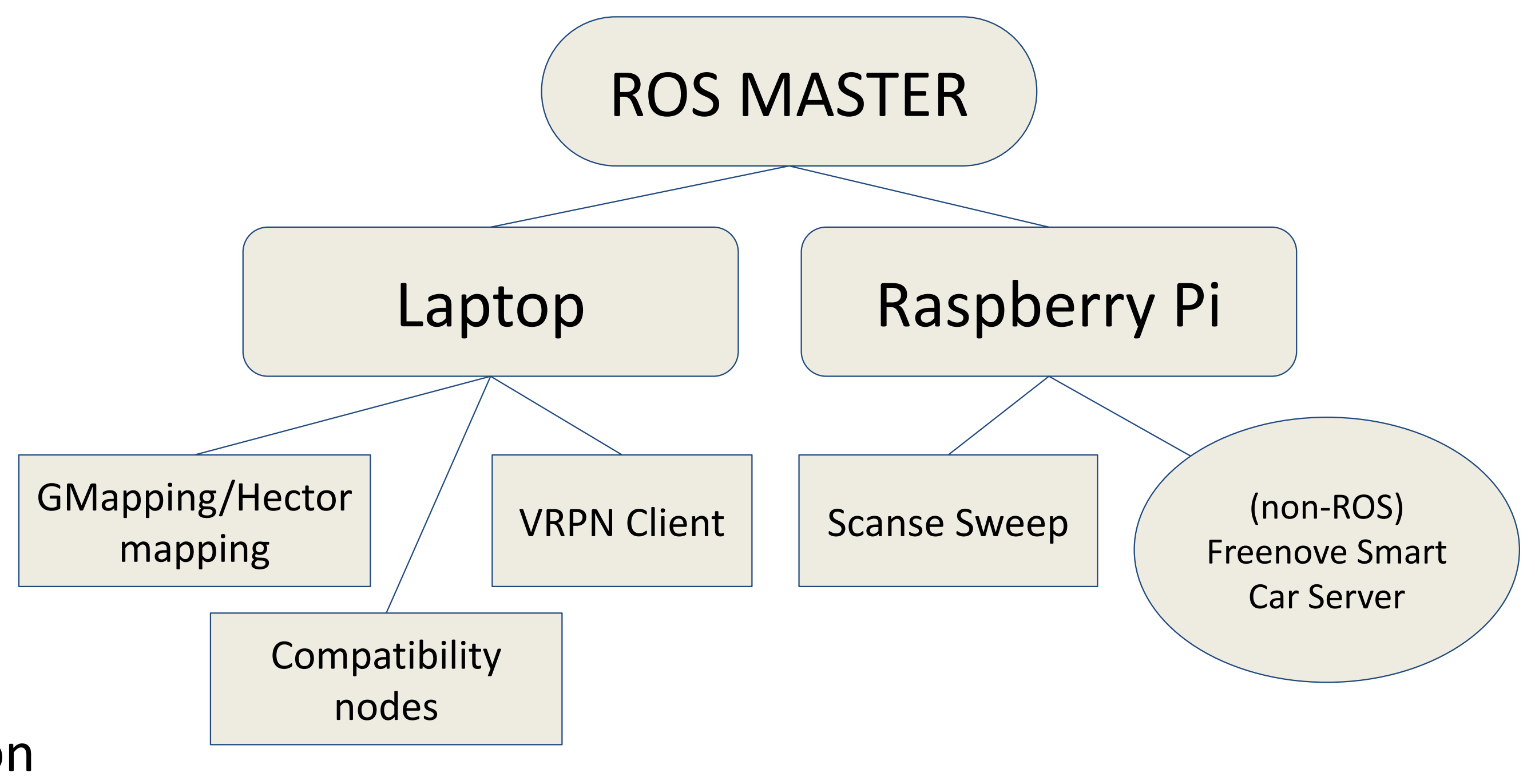
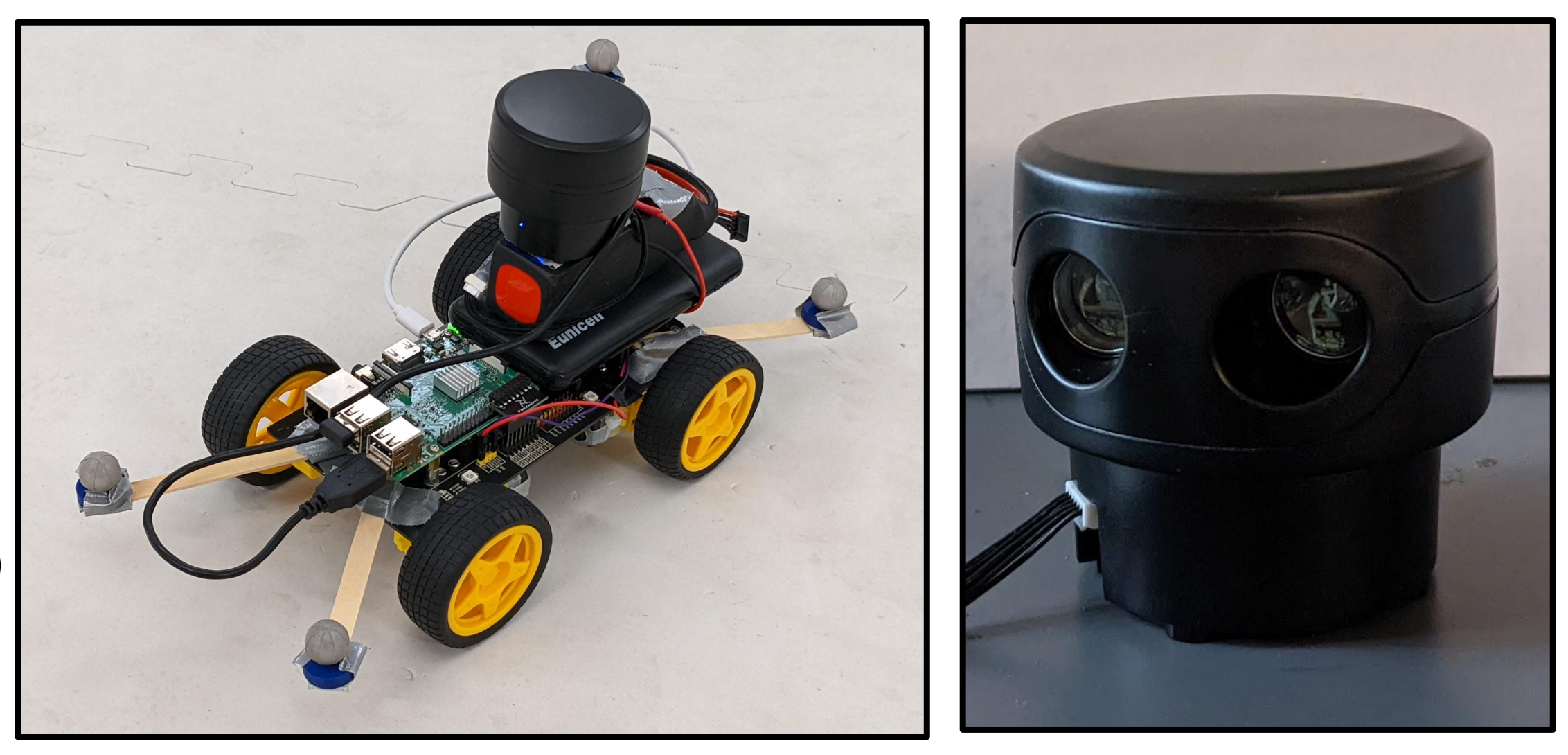


Figure 4: Freenove 4WD Smart Car with a Raspberry Pi 3 Model B, two batteries, motion capture markers, and a Scanse Sweep LIDAR (left) Scanse Sweep LIDAR (right)



Results

Maps were generated using both Hector Mapping and GMapping SLAM algorithms. See figures 1 and 2. Hector mapping, because it only requires laser scan data, was used to test the scanner to ensure it was sufficient to use in a SLAM algorithm. The LIDAR could be held and moved around manually to map the environment. GMapping required odometry data, so a handheld test was not possible. While both the GMapping and Hector Mapping algorithms were able to capture the shape of the environment, the results highlighted differences between the two, as seen in figure 1. GMapping requires more data, but the resulting map is more crisp. The Hector map's edges were less precise in comparison, and there is also an artifact at the lower right corner of figure 1. Hector Mapping proved much easier to implement, as it did not require the motion capture system in order to generate a map. However, if precision is more important, then GMapping is preferable.

Discussion/Conclusion

Further tests could compare other SLAM algorithms, or determine how much data is required to sufficiently map an area. With the robot unmoving in an enclosed environment, it was able to generate a map with the data virtually unchanging. This suggests that by just using a single scan, a SLAM algorithm may be able to generate a nearly complete map of the surroundings within its line of sight.

A limitation of this implementation, specifically the connection between the computer and Raspberry Pi with the Scanse Sweep sensor, was the delay. The Scanse Sweep had a noticeable delay between laser scans, which were problematic when the smart car moved too fast. This issue was exacerbated by the delay of the signal traveling from the Raspberry Pi to the laptop, which confused the SLAM algorithms into interpreting the laser scan as being from another location than it actually was, causing offset overlapping maps to be generated.

References

Freenove 4WD Smart Car - https://github.com/Freenove/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi
GMapping - https://github.com/ros-perception/slam_gmapping
Hector Mapping - https://github.com/tu-darmstadt-ros-pkg/hector_slam
Pointcloud to Laserscan - https://github.com/ros-perception/pointcloud_to_laserscan
Robot Operating System - Stanford Artificial Intelligence Laboratory et al. (2018). Robotic Operating System. Retrieved from https://www.ros.org
Scanse Sweep - https://github.com/scanse/sweep-ros
VRPN Client - https://github.com/ros-drivers/vrpn_client_ros
Ponnu G.; George J. Realtime ROSberryPi SLAM Robot. Cornell University, 2016.

Acknowledgements

I would like to thank the wonderful people at the lab for always answering my questions, giving support, and providing the necessary materials. I would also like to thank the RISE program for making all of this possible.