# Indoor Robot Navigation with 3D LiDAR

Stephen Sun
Advisor: Prof. Xinming Huang
Brown University

May 20, 2022

## Abstract

This project aims to investigate the problem of indoor robot navigation and people detection with a 3D LiDAR. The overall goal was to develop a robot that can navigate autonomously throughout a previously mapped indoor environment. During navigation, the robot should automatically detect and avoid colliding with any moving pedestrians. A TurtleBot 2 running ROS on board a Nvidia Jetson Xavier was used, together with a Velodyne VLP-16 (”Puck”) LiDAR.

## Acknowledgements

I would like to thank Professor Xinming Huang, my advisor for all his guidance throughout the course of the semester. Without his help, this project would not have been possible. I would also like to thank Kohmei Kadoya, Xiao Zhang, Lile Zhang and Mahdi Elhousni for help in setting up the robot and troubleshooting ROS.

## 1 Introduction

Robots are becoming increasingly common in a multitude of use cases in public environments. Specifically, in the last decade, there has been much development and deployment of autonomous robots. This includes autonomous vehicles, which are now relatively prevalent on city streets and highways. Apart from autonomous vehicles, there has also been growth in the use of autonomous robots indoors. Examples include warehouse robots used by e-commerce companies like Amazon, robot attendants at restaurants and other service venues, and delivery drones that drive or fly to their destinations.

For all these autonomous robots, the perennial problem that needs to be resolved by robot designers is that of autonomous navigation. Specifically, in indoor environment, there are additional unique challenges. When a robot navigates inside a building, it must interact with pedestrians in a socially cooperative manner. It needs to be able to detect and track pedestrians, and then modify its navigation path to avoid these moving pedestrians. This poses a significant difficulty, as the robot will need to perform this in real time while it is moving.

There are a few layers involved in allowing for autonomous navigation in a robot. First, there needs to be technology to create a 2D or 3D map of the robot’s environment. The purpose of creating a map is to enable localization, for the robot to know its position for navigation. We will discuss each of these layers in further details in sections 5,6 and 7 respectively.

## 2 Background

### 2.1 Related Work

There has been much development of various navigation and pedestrian detection algorithms using various sensors.

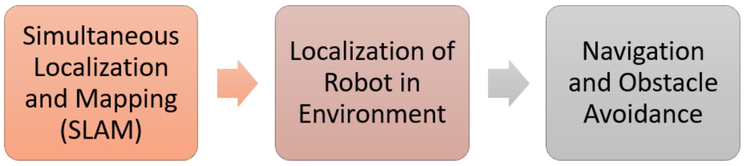
Simultaneous Localization and Mapping (SLAM)
Localization of Robot in Environment
Navigation and Obstacle Avoidance

Figure 1: Typical workflow of preparing a robot for autonomous navigation

The GMapping [1] and amcl [2] packages provide a mapping and localization solution for use with 2D sensors like a 2D LiDAR. These packages are able to work out of the box for many robot kits compatible with ROS today. ROS also provides a navigation stack [3] that contains GMapping and amcl, together with a costmap generator and path planner. This navigation stack allows for the autonomous navigation of a robot through a 2D mapped environment.

There has also been projects aimed at using a visible light camera such as the Intel RealSense camera for pedestrian detection, and then transmitting the coordinates of any detected pedestrians to the navigation stack such that the robot can plan a path that avoids the pedestrian. [4]

#### 2.1.1 LiDAR

LiDAR stands for Light Detection and Ranging. It uses light in the form of a pulsed laser to measure distances to objects. The time taken for laser pulses to bounce off objects and return to the sensor is measured, allowing a real-time 3D map of the surroundings to be created [5]. It is similar in operation to Radar apart from the fact that it uses light (specifically usually Infrared beams) waves, hence the similar name.

LiDAR sensors are widely used for geodesy and computer vision applications. Specifically relevant to us is the fact that LiDAR is able to create a precise map of ever-changing surroundings quickly and regardless of weather and visibility conditions. Thus LiDAR is commonly a key component for autonomous robots, being one of the sensors that the robot uses for safe navigation.

### 2.2 Motivation

We can see that there is currently no comprehensive package that uses only a 3D LiDAR for pedestrian detection and avoidance. While 2D navigation works perfectly well for indoor robots, the major disadvantage of using 2D sensors (for instance a 2D 1-line LiDAR) for pedestrian detection is that it is very difficult to differentiate a pedestrian from any regular obstacle. A pedestrian on a 1-line LiDAR will simply appear as a series of a few points that are closer to the robot than the wall. This means that it will be impossible to track the pedestrian for obstacle avoidance.

Thus, we aim to create a autonomous robot that performs mapping, localization and pedestrian detection and avoidance using only a 3D LiDAR.

## 3 Robot Component Overview

### 3.1 Robot Operating System (ROS)

The Robot Operating System (ROS) is a open source robotics middleware suite. Contrary to its name, it is not an operating system, but rather a software platform for robotics software development [6]. ROS provides a common foundation for developers in different areas of robotics, and has become the industry standard for many robots.

ROS comes with deep integrations with Linux Operating Systems, and has major explicit integrations with Ubuntu. ROS comes in many different versions and releases, and is usually directly tied to a certain Ubuntu release. There are 2 main versions of ROS now, ROS 1 and ROS 2. ROS 2 is the newer version that has been redeveloped from the ground up, and is functionally quite different from ROS 1. In this project, we used ROS 1 Melodic Morenia installed on Ubuntu 18.04.

A common tool that we used throughout this project is Rviz. Rviz is a visualization tool built for ROS. It is specifically designed to interpret and visualize ROS messages sent through ROS topics, which are essentially packets of data sent through a stream from other programs.

### 3.2 TurtleBot 2

The TurtleBot is a robot kit designed to integrate with ROS. The TurtleBot is designed to be modular and be easy to set up for educational applications. It is made up of a Kuboki base, and is designed to be connected to a laptop or an embedded computer. TurtleBot is supplied with drivers and software for ROS, so that it is easy to connect it with other ROS programs.

### 3.3 Nvidia Jetson

The computer that we used to power the TurtleBot and run ROS is the Nvidia Jetson Xavier. The Nvidia Jetson is a series of embedded computing boards designed for embedded AI applications. Jetson boards contain a Tegra CPU and full GPU on board, making them suitable for our application. A Jetson board can either be connected to a monitor, or be run as a remote desktop and be connected to via VNC.

### 3.4 Velodyne VLP-16 LiDAR

The Velodyne VLP-16 LiDAR we used is a 16-line 3D LiDAR. It typically scans at 10hz, and can be integrated with ROS to send its scanned data as pointcloud messages. It has a 100m range, and uses 905nm infrared lasers for scanning its surroundings [7]. With its 16 lines, it is able to scan up to 300,000 points per second. It has a 30° vertical $(+/-15°)$ and 360° horizontal field of view. With an IP67 rating, it is resilient for challenging environmental conditions.

## 4 Setup of TurtleBot

### 4.1 Software Setup

The Jetson had Ubuntu 18.04 with ROS Melodic installed, so we proceeded to set it up for a remote connection. There was some difficulty in getting the remote server working, due to the need to register any embedded device on the Brown network before it can be used.

### 4.2 Hardware Setup

The TurtleBot 2 was supplied disassembled, so we had to put it together at first. Also, due to the field of view of the LiDAR, it needed to be mounted at a slightly higher position than the top frame of the TurtleBot allowed for. Thus, it was necessary to print out a base for the LiDAR. A 3D model was found online on Thingiverse, a website for sharing 3D printable models [8]. This was 3D printed at the Brown Design Workshop.

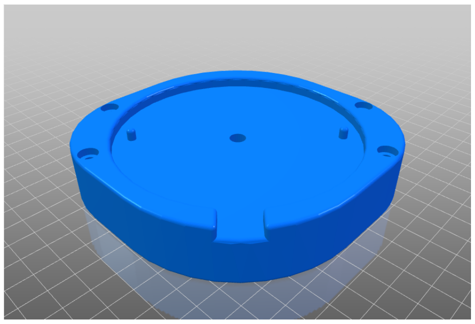

Figure 2: A 3D view of the VLP-16 LiDAR mount for TurtleBot.

The TurtleBot is made up of platforms with mounting holes at various heights. Upon assembly of the TurtleBot, the mount was attached via screws to the highest platform to maximize the LiDAR's field of view. The LiDAR is connected to the Jetson via ethernet, and the TurtleBot is connected via USB.

To power all components of the system while keeping the robot mobile, we had 2 options. First, we could run everything off the internal battery of the TurtleBot. The drawback of doing that is the fact that the TurtleBot's internal battery will be depleted very rapidly. Hence, we chose to power components with an external battery pack. The external battery pack has a 12V and a 9V outlet. The 12V outlet is used to power the Jetson and the 9V outlet is used to power the LiDAR.

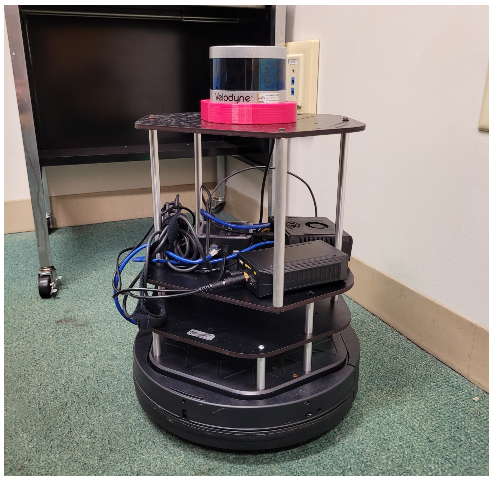

Figure 3: The final assembled TurtleBot.

## 5 SLAM

SLAM refers to Simultaneous Localization and Mapping. It is a method that allows a robot to build a map and localize itself within that map at the same time[9]. SLAM algorithms allow robots to map out unknown environments and then carry out other tasks. Robots running SLAM algorithms are generally able to localize themselves only with onboard sensory apparatus. There are two common types of sensors used: cameras (visual SLAM) and LiDAR (LiDAR SLAM).

Many different SLAM algorithms are available today. Researchers at the University of Zagreb [10] have looked into different SLAM algorithms. They came up with 3 main candidates: LOAM [11], Google's Cartographer [12], and hdl_graph_slam [13]. We chose to look into Hdl_graph_slam in more detail, as we initially desired to run all aspects of the workflow (mapping, localization and pedestrian detection) in 3D.

In the SLAM mapping process, we would manually drive the robot around by running turtlebot_teleop. The teleoperation package allows the TurtleBot to be driven via the keyboard remotely.

### 5.1 Hdl_graph_slam

Hdl_graph_slam was a package developed by researchers from AIST to work with a separate 3D localization as well as 3D people detection and tracking package. These three packages were meant to monitor and track people while the system is carried in a backpack. It was meant to record people behaviors in an indoor environment. Because of the similarity in working environment, as well as the fact that there are packages designed for other parts of our workflow, we chose to test out this package. The way that these packages integrate is shown in the figure below.

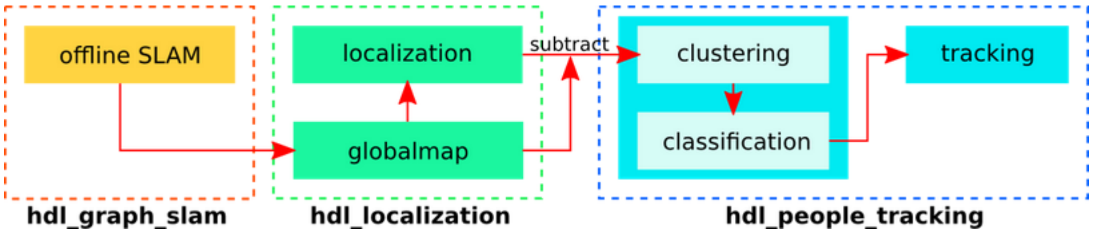

Figure 4: Diagram of components of how Hdl_graph_slam integrates with other packages from AIST. [13]

The Hdl_graph_slam generates a map in the form of point clouds, and can be saved in the form of a pcd file.

### 5.2 GMapping

Upon determining that the Hdl navigation stack was unsuitable for our application due to poor pedestrian detection performance, we looked into using other packages for navigation. GMapping is a SLAM algorithm that performs 2D mapping. To use this package with our 3D LiDAR, we would choose 1 of the 16 lines to act as input data for the algorithm.

This map that this produced works well. The algorithm produces a 2D occupancy grid map, which directly represents whether an arbitrary point is occupied or not. This map can be visualized in Rviz. An image of the map together with the localization can be seen in figures 6 and 7.

## 6 Localization

### 6.1 Hdl_localization

After successfully obtaining a pcd file from Hdl_graph_slam, this can be fed into the Hdl_localization package. This localization package is able to perform localization globally with or without an initial guess, by matching up current sensor data with the points from the loaded map. These points that have been matched and aligned are published to the /aligned_points ROS topic.

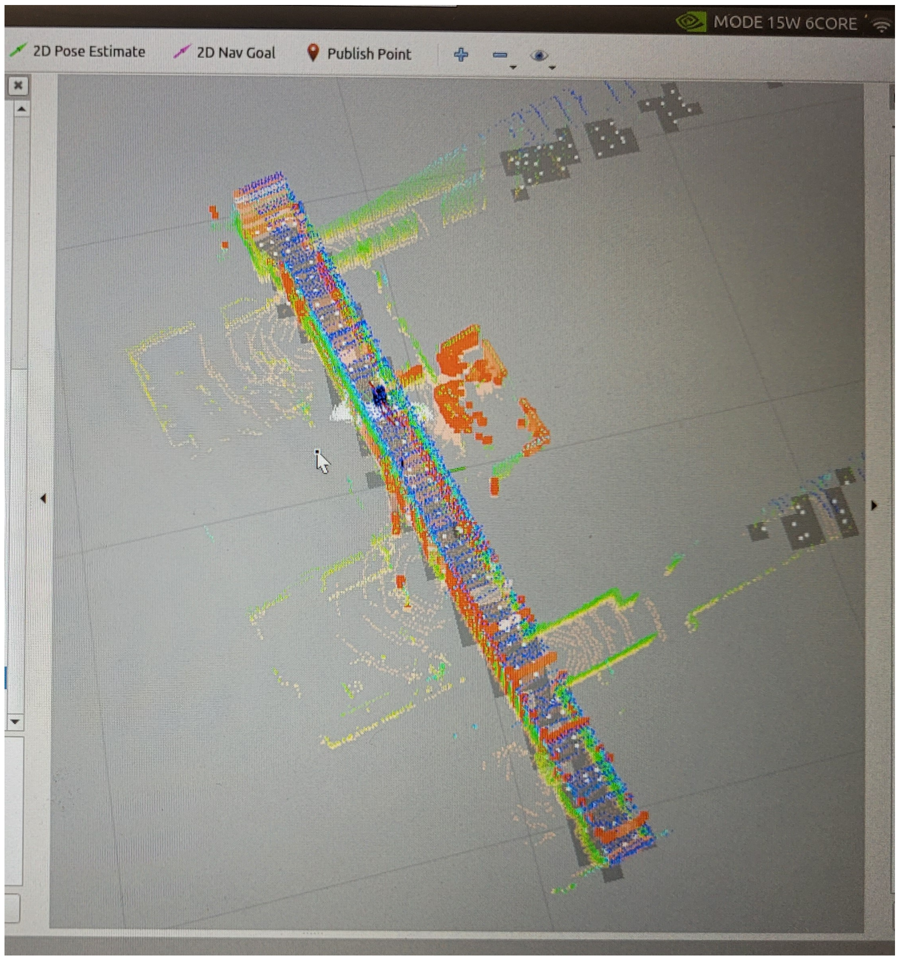

Figure 5: A successful mapping and localization of the Brown Center For Information Technology 5th floor corridor.

As we can see in figure 5, the points that are matched up are marked with orange squares, while the rest of the map scanned by Hdl_graph_slam are of various colors. Thus we can say that localization works successfully with this package. The

## 6.2 Amcl

Amcl is a localization package that is able to localize a robot within a 2D map. The package is a part of the ROS navigation stack package, and is designed to work with GMapping.

By passing it a map file that we made with GMapping, Amcl was able to determine the robot's position and orientation after being given an initial guess in Rviz.

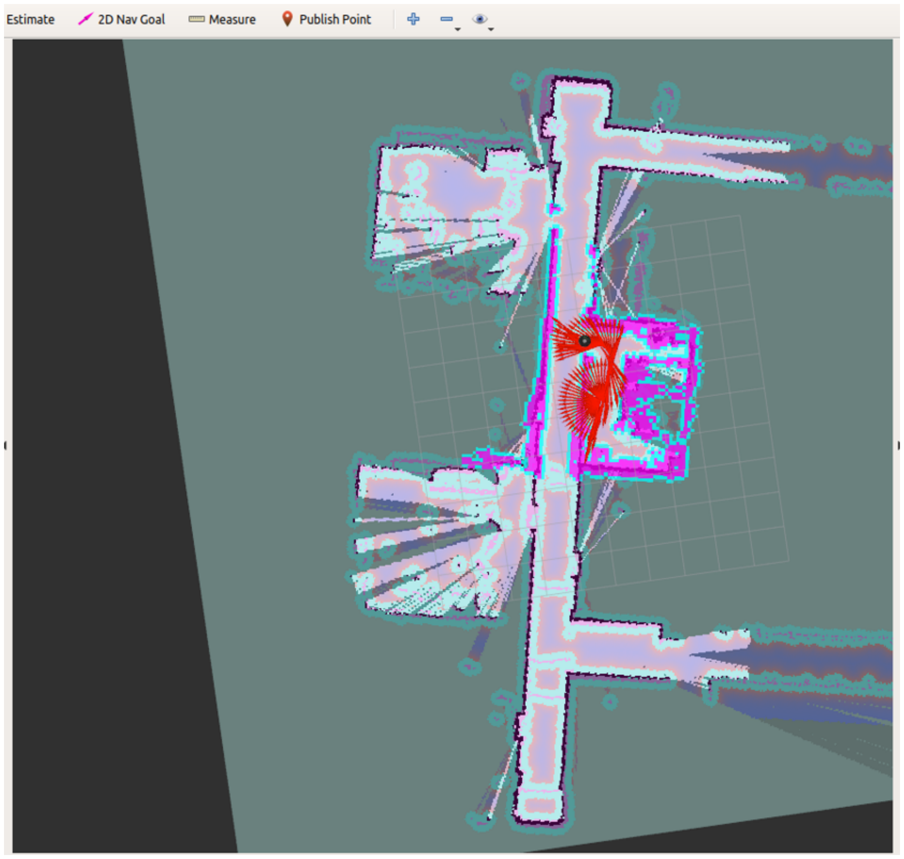

Figure 6: Amcl working well on the 2D map generated by GMapping.

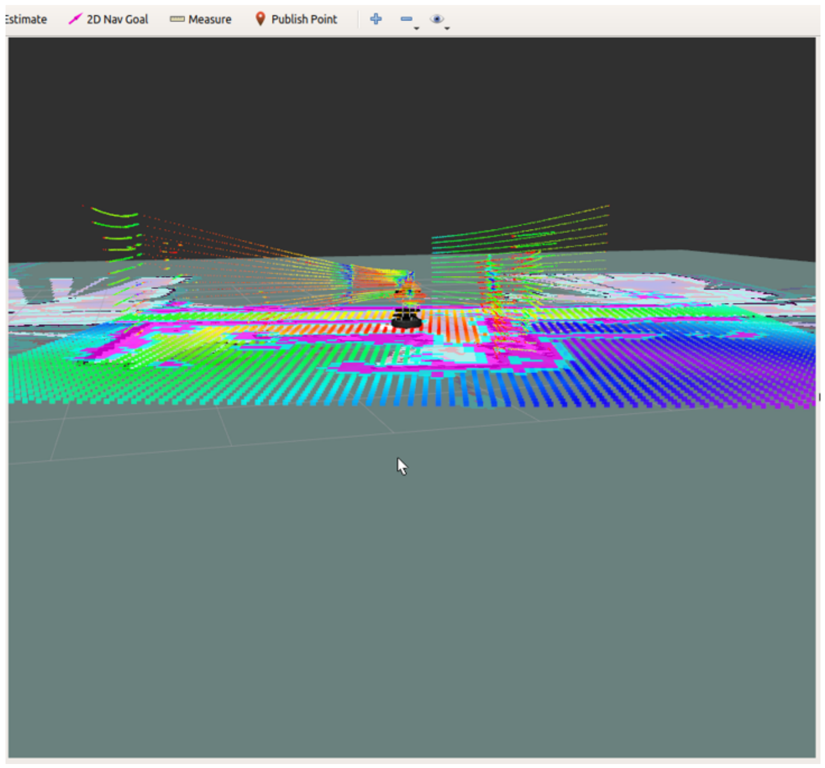

Figure 7: The 2D map with the 3D LiDAR's scans overlaying it and the robot's current position.

From figure 6, we can see the robot's position and orientation marked out by the robot model in the center, and the red arrows to indicate its orientation. In figure 7, we can see that the 2D map and localization can be overlapped with the 3D LiDAR's current scan data, and thus it can be designed to work wtih a 3D person detection algorithm. We can simply use navigation in 2D for our application because an indoor robot is only able to move in 2 dimensions. Hence, there is no need for height information to be recorded in the map.

## 7 People Detection

### 7.1 Hdl_people_tracking

This package first performs Haselich's clustering technique to detect human candidate clusters, and then applies Kidono's person classifier to eliminate false detections. People will be marked in Rviz as colored cubes.

Unfortunately, in our testing, this package worked extremely poorly. As we can see in figure 6, the package detected many false positives, which were marked out by the different colored cuboids on the map. Furthermore, the package completely failed to detect the actual person walking along the corridor in our tests. The combination of many false positives, and a false negative for the actual person led us to look at other people detection packages.

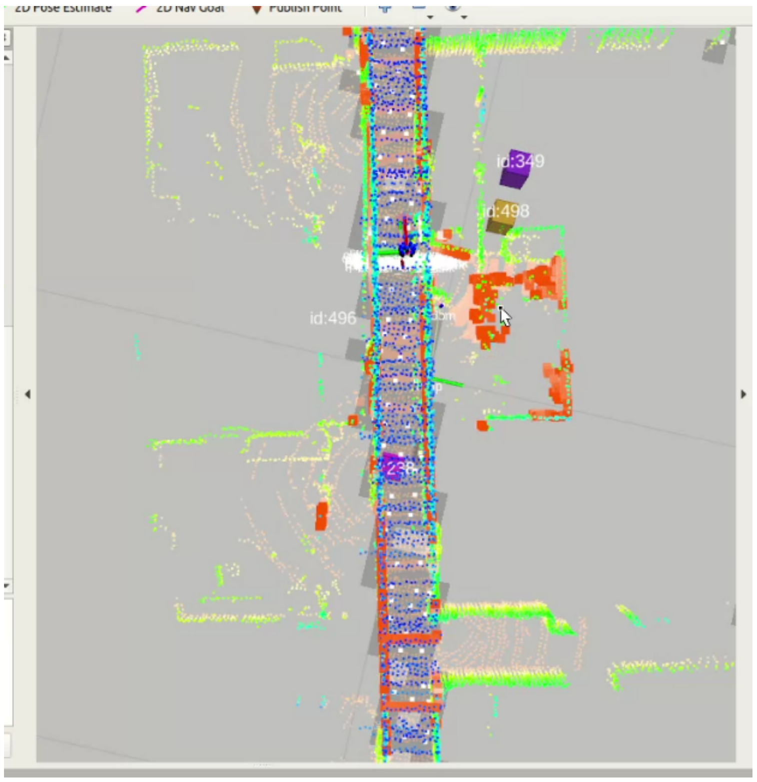

Figure 8: Substandard performance by Hdl_people_tracking which contained many false positives.

## 7.2 Online_learning

Another package we investigated was the online_learning package by [14]. This package is an online learning framework for people classification in 3D LiDAR scans. It uses a support vector machine (SVM), specifically the libsvm package to classify people. It comes in an offline and online version, where the online version trains itself on new data constantly to improve predictions. Any pedestrian candidates are marked with a blue cuboid bounding box, and after performing classification, any pedestrians that are detected are marked with green bounding boxes. The detection algorithm runs at 10 hz.

Unfortunately again, the ability of the package to detect people was poor. As can be seen in figure 9, the package fails to detect any pedestrians and mark them out with green bounding boxes. Even the actual person walking around (annotated with a red arrow in the figure) is not detected, and moreover not even marked as a candidate with a blue bounding box. Instead, we can see that a couple of random spots are marked as candidates (annotated with blue arrows).

There were occasions where the person walking around is detected as a candidate, as shown in figure 10 but the issue is that there is no persistence in the bounding box, and it disappears immediately in the next iteration of the detection algorithm.

The developers of this package also directed us to a GUI tool that they made [15]. This tool is supposed to be able to annotate any 3D LiDAR data (in the form of per-frame pcd files), and should be used to indicate the locations of pedestrians. The tool can then train the SVM, and make the model more accurate. However, due to the fact that the tool was developed for use in Ubuntu 14.04, we were unable to get the tool to work properly on our Jetson.

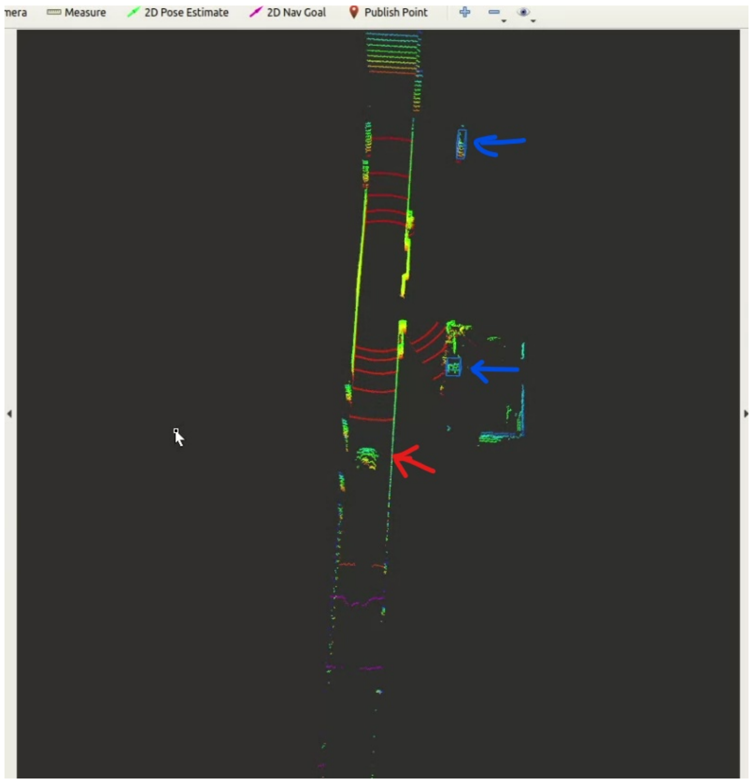

Figure 9: Performance by the online_learning package was also poor.

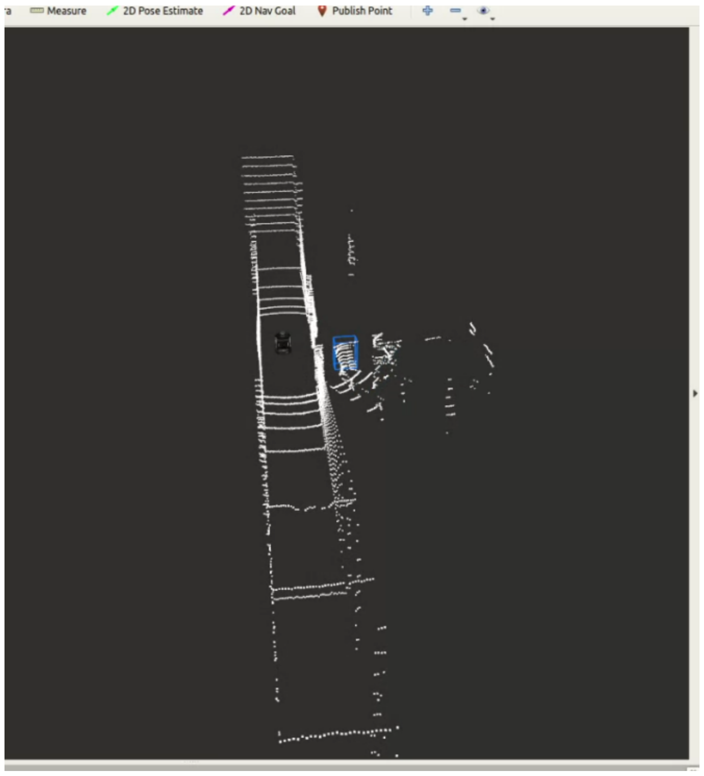

Figure 10: A one-off detection of a walking pedestrian.

## 8 Conclusions

Overall, while we were not successful in our original goal of developing an integrated autonomous navigation stack that performs pedestrian detection and avoidance with a 3D LiDAR, we have managed to explore many different packages and algorithms for the different parts of the stack.

We can see that there are many packages that fulfill individual components of the navigation workflow, but they are unable to work well together. Integrating and ensuring that data is properly passed between different nodes of the navigation workflow is key to ensuring that autonomous navigation works effectively.

### 8.1 Future Work

In the future, more work can be put into training the machine learning models, or perhaps writing a custom model for pedestrian detection. There can also be more research into the navigation and path planning algorithms that the robot will use. During the course of the project, we also discovered that Matlab contains many toolboxes such as the ROS Toolbox, LiDAR toolbox, and Navigation Toolbox. These toolboxes are more substantially documented as compared to the Github repositories that were tested in this project, and also provide significantly more example code. Using Matlab may help to provide a more straightforward integration of the various parts of the navigation workflow.

## References

[1] G. Grisetti, C. Stachniss, and W. Burgard. [Online]. Available: https://openslam-org.github.io/gmapping.html

[2] B. P. Gerkey, "Amcl package." [Online]. Available: http://wiki.ros.org/amcl

[3] Ros-Planning, "Ros-planning/navigation: Ros navigation stack." [Online]. Available: https://github.com/ros-planning/navigation

[4] J. Song, L. Zhang, Z. Xu, and Y. Xu, "Pedestrian avoidance for indoor robots," *Worcester Polytechnic Institute*, August 2020. [Online]. Available: https://digital.wpi.edu/pdfviewer/nv9355467

[5] "What is lidar? learn how lidar works," Apr 2022. [Online]. Available: https://velodynelidar.com/what-is-lidar/

[6] "Robot operating system," 2022. [Online]. Available: https://www.ros.org/

[7] "Puck lidar sensor, high-value surround lidar," May 2022. [Online]. Available: https://velodynelidar.com/products/puck/

[8] MIRT_Turtle, "Turtlebot 2 velodyne laserscanner vlp-16 (puck) mount by mirt_turtle," Aug 2016. [Online]. Available: https://www.thingiverse.com/thing:1699962

[9] "What is slam (simultaneous localization and mapping)." [Online]. Available: https://www.mathworks.com/discovery/slam.html

[10] R. Milijas, L. Markovic, A. Ivanovic, F. Petric, and S. Bogdan, "A comparison of LiDAR-based SLAM systems for control of unmanned aerial vehicles," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2021. [Online]. Available: https://doi.org/10.1109%2Ficuas51884.2021.9476802

[11] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, no. 9. Berkeley, CA, 2014, pp. 1–9.

[12] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.

[13] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional lidar-based system for long-term and wide-area people behavior measurement," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419841532, 2019. [Online]. Available: https://doi.org/10.1177/1729881419841532

[14] Z. Yan, T. Duckett, and N. Bellotto, "Online learning for human classification in 3D LiDAR-based tracking," in *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Canada, September 2017, pp. 864–871.

[15] ——, "Online learning for 3d lidar-based human detection: Experimental analysis of point cloud clustering and classification methods," *Autonomous Robots*, 2019.