OVERHEAD OBSTACLE AVOIDANCE AND VERBAL COMMUNICATION FOR NAVIGATION IN ROBOTIC GUIDE DOG SYSTEMS

BY

JAKE JUETTNER

BA, Binghamton University, 2024

THESIS

Submitted in partial fulfillment of the requirements for
the degree of Master of Science in Computer Science
in the Graduate School of
Binghamton University
State University of New York
2025

© Copyright by Jake Juettner 2025

All Rights Reserved

Accepted in partial fulfillment of the requirements for
the degree of Master of Science in Computer Science
in the Graduate School of
Binghamton University
State University of New York
2025

December 11, 2025

Shiqi Zhang, Chair
School of Computing, Binghamton University

Yincheng Jin, Faculty Advisor
School of Computing, Binghamton University

Abstract

For blind and low-vision (BLV) people, a guide dog is a vital tool to navigate their daily environment. Despite this, guide dogs are rarely used by people who need them. This thesis presents several systems developed in service of creating an autonomous Robotic Guide Dog system that could provide another option for those seeking assistance. First, a comparative analysis of two obstacle detection systems using LiDAR sensors and RGB-D cameras on the detection of overhead obstacles. Both systems were implemented and their effectiveness was evaluated in real-world scenarios involving overhead obstacles. The results show that LiDAR offers a superior detection range, while RGB-D enables intuitive feedback through audio signaling within the context of robotic guide dogs. Our findings support the development of hybrid solutions for assistive robotics. These results have direct implications for autonomous mobility assistance, especially in environments where traditional guide dogs or wearable devices do not function. Second, a language navigation system interface was developed. This enables the robot to build a map of its surroundings and navigate through them using LiDAR. Additionally, this allowed guide dog users to give destinations to an AI voice agent and have the robot navigate to those locations autonomously. This was accomplished by creating a custom topic for the voice agent in ROS2 Foxy that could communicate with the navigation topic through navigation requests.

Acknowledgments

First, I would like to thank Professor Shiqi Zhang, who acted as my advisor. He is a great mentor who guided me towards success. He was always there when I needed help during the development and writing of this thesis. Next, Yi-Shan Wu, thank you for creating the LiDAR overhead obstacle detection system, running the comparative experiments, and helping to write the original report on which this thesis is based. Then, Chao Lin thank you for helping to program the obstacle detection for the RGB-D system. Kevin Yang, thank you for helping to research and implement ROS2 SLAMToolBox package developed by Andy Zhou. Finally, Nicholas DellAccio thank you for programming and developing the voice agent system in conjunction with the SLAM navigation. I could not have produced this thesis without all of your help and support.

I declare that no generative artificial intelligence (AI) tools or services were used in the research, writing, analysis, or any other aspect of this thesis.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

RGB-D Red Green Blue - Depth

LiDAR Light Detection and Ranging

SLAM Simultaneous Localization and Mapping

ROS Robot Operating System

VNC Virtual Network Computing

RANSAC Random Sample Consensus

DBSCAN Density-Based Spatial Clustering of Applications with Noise

LLM Large Language Model

1 Introduction

Guide dogs are a daily occurrence for many people in the Blind and Low Vision (BLV) community. To safely transport the handler to their destination, guide dogs are trained to navigate unfamiliar environments and avoid obstacles present in modern life. However, for all the feats guide dogs are capable of, there are areas in which they fall short, leading to unfortunate mishaps. Guide dogs tend to miss obstacles at human head height, even though they are trained to avoid obstacles within view. This could be low hanging signs, improperly cut foliage, construction, or small doorframes, to name a few. Guide dog trainers have trained their dogs to look up when assessing a scene for potential obstacles, but this can tire the dog out if done for a prolonged period of time. To account for this, I created two modules of a system, namely an Overhead Obstacle Detector and Voice Navigation Agent, that can be used to improve navigation capabilities for BLV individuals without taking away from what many like about guide dogs.

2 Background

This thesis focuses on developing an Overhead Obstacle Detection and Voice Navigation system for robotic guide dogs. Researchers have explored similar research topics for robotic guide dog navigation that I will highlight here. One work placed a map within the environment for a robot dog to scan and use for navigation [6]. This allows the robot to use autonomous navigation to move the robot around the map. Another set of authors consider physical communication between the human and the robot through external pulls on the back to guide its movement [5]. In addition to navigation, researchers have also considered different gaits for robot navigation that could enhance robot locomotion. Creating systems to simulate and learn these gaits through reinforcement learning [4].

The remainder of the chapter will focus on technologies and techniques that were widely used throughout the thesis. Providing a brief baseline for each of their capabilities and common implementations.

2.1 RGB-D Camera

RGB refers to the red, green, and blue color channels used within a standard camera. The RGB-D camera's differ by introducing a new depth channel. This channel maps depth information to each captured pixel, allowing the camera to determine the distances of objects within the frame. Depending on the camera, depth information can be calculated in three different ways [8]. The first is structured light, which projects an inferred pattern onto anything in front of the camera. When the light pattern hits the object, it deforms based on the shape of the object in front of it. The camera compares the deformed pattern to the original pattern to determine the depth [2]. The second is Time-of-Flight, in which the camera sends out an infrared laser to bounce off

objects in the environment. The amount of time it takes for the light to return to the camera determines the distance the object is from the camera [18]. The third method is stereo vision, which uses two separate cameras that are spaced apart. The images of two cameras can be compared based on their viewing angle to determine the depth of the object in the scene [8].

2.2 LiDAR

Light Detection and Ranging (LiDAR) sensors are spinning sensors that utilize laser light to create a 3D point cloud. For each point within a point cloud, a laser is emitted from the sensor. Depending on the amount of time it takes the laser emitted from the sensor to return, the sensor can calculate the distance it traveled before it was reflected back to the sensor [17]. This is because the speed of light within a medium is constant, so knowing the total travel time allows the total distance traveled to be calculated. Then that distance can be divided in half to determine the distance from which the laser was reflected. LiDAR also uses GPS and IMU sensors to determine coordinates x, y, z, and the orientation of the hit object [17].

2.3 SLAM

Simultaneous Localization and Mapping (SLAM) is the process of creating maps of unfamiliar environments and calculating the position of the robot within that environment [14]. Depending on the robots capabilities and the current environment, SLAM is implemented differently, however, the underline process is still the same. The robot utilized throughout this thesis is the Unitree Go2, a quadruped robot equipped with a LiDAR sensor. SLAM takes 3D point cloud data generated by the LiDAR and extracts key features within the environment. This information is then passed to an Extended Kalman Filter to determine the uncertainty of the robots and the key features positions [14]. As the robot moves around the environment and collects new features, the robot's position is recalculated based on the uncertainty of its current position, the

position of the new features and the updated positions of old features [14]. Overtime, as the features are saved, a map will be generated that can be used by the robot later for navigation.

3 Overhead Obstacle Detection For Robotic Guide Dogs

3.1 Overview

This thesis presents a comparative analysis of RGB-D and LiDAR sensors in robotic visual perception. RGB-D cameras estimate depth using structured light or time-of-flight, enabling spatial perception with two- or three-point localization. LiDAR, on the other hand, generates accurate point-cloud data that effectively captures distance and dynamic motion. Both systems were implemented and their effectiveness was evaluated in real-world scenarios involving suspended obstacles. The results show that LiDAR offers a superior detection range, while RGB-D enables intuitive feedback through audio signaling. These findings support the development of hybrid solutions for assistive robotics, and the results have direct implications for autonomous mobility assistance. Especially in environments where traditional guide dogs or wearable devices fall short.

Robotic perception is crucial for autonomous navigation, object recognition, and interaction. Two key sensing technologies are RGB-D cameras and LiDAR. RGB-D cameras capture color and depth, enabling spatial estimation through two- or three-point localization [13]. LiDAR, in contrast, produces high-precision 3D point clouds for accurate distance measurement and dynamic scene understanding [16, 19]. A demonstration video of the obstacle detection system is available here. [1]

Independent mobility for visually impaired people often relies on traditional guide dogs or white canes, which are limited to awareness of obstacles at ground level. Suspended or overhead

[1] https://drive.google.com/drive/folders/1MK4aS23N3qrotZ3ixH5X_VcudIx4EqFj?usp=sharing

hazards, such as signs, low-hanging bars, or tree branches, pose a serious risk, as they are often undetectable without advanced detection. To address this challenge, robotic platforms must interpret 3D spatial geometry with sufficient vertical resolution, low latency, and robustness to real-world environmental factors.

This thesis compares the performance of RGB-D and LiDAR in various environments, analyzing their strengths and limitations for robotic applications.

Guide robots play a crucial role in helping visually impaired individuals navigate autonomously in dynamic environments. Existing robots efficiently detect obstacles at ground-level but struggle with high obstacles such as hanging signs and tree branches [1]. This limitation poses a safety risk. Since the robot can continue to move forward, it assumes that the person it is guiding can do the same. Inflicting possible injuries that could have been avoided. This project aims to prevent this by using various vision techniques and technologies to detect objects above the robot before the following person has the opportunity to interact with it [7]. If successful, a robot that could detect overhead obstacles could compensate for a major flaw that normal guide dogs have, leading to a better user experience for visually impaired individuals.

Assess the status of the obstacle in a three-dimensional space to provide a warning or change course if necessary, following a similar approach as demonstrated [3]. Depending on the assessment, the user could be warned of an obstacle and begin to change course if the object cannot be avoided. In the case where the object is too large or dangerous, the path with the overhead obstacle will be treated as a blocked path. Signal to the robot that the route it has planned is not feasible and that it must update its assumption. This process would use SLAM algorithms to reroute and reposition both the robot and the person to the desired location without interacting with the overhanging object. By incorporating adaptive path-planning techniques, the system ensures that navigation remains smooth and efficient without unnecessary stops or delays.

This thesis aims to address this issue by integrating advanced perception techniques, including 3D-LiDAR, and RGB-D cameras. Using these technologies, I propose an improved system

that allows guide robots to detect and avoid high obstacles, ensuring safer navigation. The anticipated results include improved safety, efficient sensor integration, and improved usability for visually impaired individuals [10]. Using data from both technologies, the user would be warned of an incoming obstacle by a frequency emitted within the handle of the guide dog. Changing depending on the distance from the obstacle above the guide dog. The frequency will be based on a frequency commonly used for visually impaired individuals in other apparatuses, such as smart canes. This will hopefully ensure an easy adoption of our purposed solution by members of the visually impaired community and a seamless transition from other commonly used items that they have used in their daily lives.

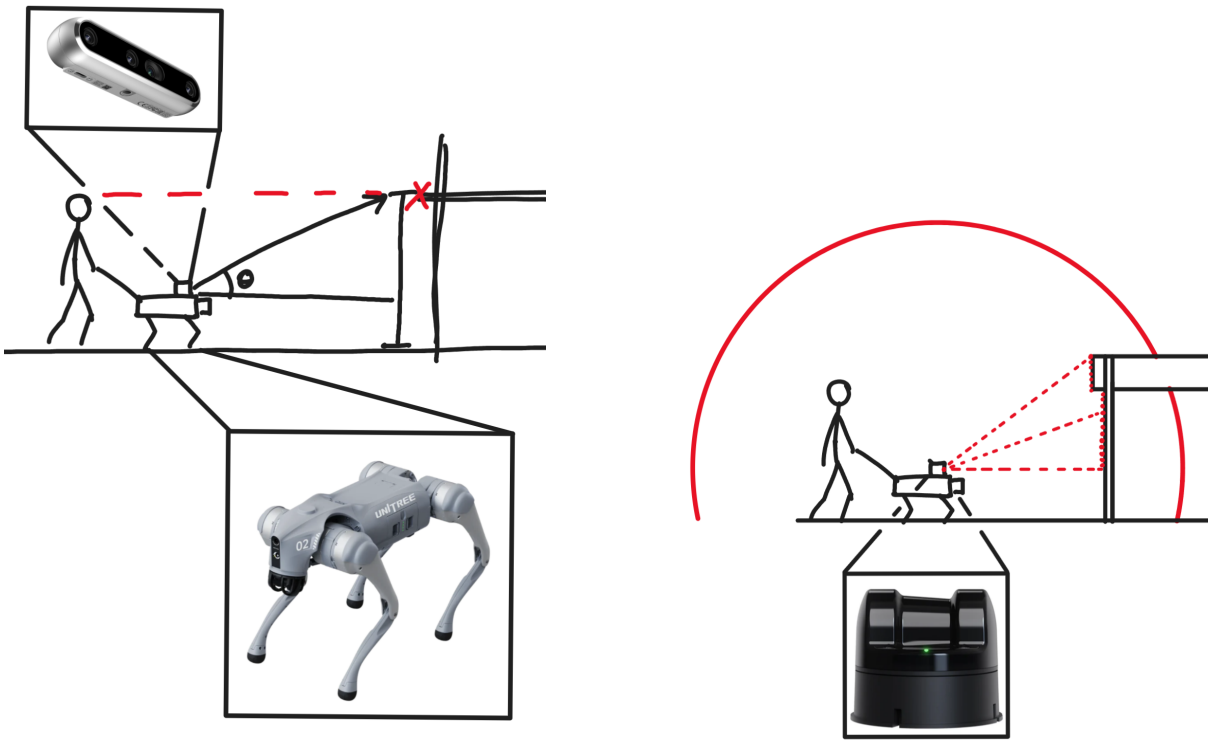

Figure 3.1: Problem scenario: Traditional guide dogs and canes cannot detect hanging or suspended obstacles. The RGB-D and LiDAR configurations provide complementary coverage.

In addition, the thesis evaluates efficiency, speed, and user experience to ensure a balance between rapid obstacle detection and user comfort. This assessment will involve testing different detection algorithms, sensor responsiveness, and real-time processing capabilities. Both 3D LiDAR and RGB-D cameras allow for accurate object detection but approach it in completely different ways. Comparing each method against one another to see which method performs

best in a variety of scenarios and tests can illuminate the strengths and weaknesses of both approaches.

3.2 Research Objectives

The increasing reliance on robotic vision systems requires an in-depth evaluation of sensor modalities such as RGB-D cameras and LiDAR. Each technology offers distinct advantages, making it crucial to determine their effectiveness in various robotic applications. This thesis aims to analyze their performance under controlled scenarios and provide insight into their respective strengths and limitations.

Specifically, this research aims to:

- **Compare the accuracy of spatial perception using RGB-D and LiDAR [9].** Depth estimation is a critical factor in robotic navigation and object recognition. RGB-D cameras generate depth maps using structured light or time-of-flight, while LiDAR produces high-resolution 3D point clouds. This thesis will assess the accuracy of both technologies in detecting object distances, shapes, and spatial relationships.
- **Assess the computational efficiency of both modalities in real-time robotic applications [11].** Efficient processing is essential for autonomous systems operating in dynamic environments. Although LiDAR data are highly precise, they may require extensive computational power. RGB-D cameras, on the other hand, provide depth data alongside visual information, potentially reducing processing complexity. This research will evaluate the trade-offs between computational cost and real-time performance.
- **Evaluate the robustness of each method under different environmental conditions.** Sensors often encounter challenges such as varying light, occlusions, and reflective surfaces. This thesis will examine how RGB-D and LiDAR perform in diverse indoor and outdoor settings, assessing their reliability in detecting objects under different conditions.

- **Provide insights into the advantages and limitations of RGB-D and LiDAR for robotic sensing.** Understanding the strengths and weaknesses of each technology will help inform sensor selection for specific robotic applications. This research aims to offer practical recommendations for the integration of RGB-D and LiDAR in vision-based robotic systems.

By addressing these objectives, this thesis will contribute to the ongoing development of advanced perception systems for autonomous robots, enhancing their ability to interact with and navigate complex environments.

3.3 Challenges

This thesis required software and hardware integration between the Unitree robot and the two different vision apparatuses. Due to the various differences in how the RGB-D camera and the LiDAR collect data from the environment, they both required different hardware setup on the robot and a different software back-end to make them function correctly. As such configuring both systems each brought along its own trials and tribulations, as I needed to learn the best way to allow the hardware and software to communicate seamlessly.

RGB-D images and depth maps will be collected using an RGB-D sensor, while LiDAR point cloud data will be acquired using a LiDAR sensor [12]. The RGB-D sensor captures both visual and depth information, whereas the LiDAR sensor generates high-precision 3D point clouds. The collected data will be preprocessed to ensure consistency and accuracy before further analysis.

To assess sensor performance, the experiments will involve multiple controlled environments, including:

- **Indoor and outdoor settings:** Evaluation of how environmental factors such as confined spaces, open areas, and reflective surfaces affect depth sensing and object recognition.

- **Different lighting conditions:** Testing under varying illumination levels, including bright sunlight, dim lighting, and complete darkness, to examine sensor robustness in challenging visibility conditions.
- **Varying levels of occlusion:** Introducing partial and full occlusions to observe how each sensor handles object recognition and spatial mapping in cluttered environments.

The collected data will be analyzed using both conventional and deep learning-based methods. LiDAR point cloud data will be processed using point-based deep learning techniques [16, 19], while the RGB-D data will be analyzed with image processing techniques [13, 15]. The following metrics will be used for evaluation:

- Accuracy
- Processing time
- Robustness to noise

3.4 System Architecture

The tilt angles chosen were empirically determined to balance near-field coverage with forward-looking detection. The Raspberry Pi's USB 3.0 port ensured sufficient bandwidth for depth stream processing, while Wi-Fi connectivity enabled remote monitoring and audio feedback triggering from an external compute unit. Power consumption was optimized using a portable 5V battery bank, which supports more than two hours of untethered operation.

The depth camera was configured to the Raspberry Pi using the Intel RealSense Python library, which allowed simultaneous access to RGB and depth data streams. The Raspberry Pi served as the on-board processing unit and was flashed with the latest version of the Raspberry Pi OS. To enable convenient monitoring and development, VNC and SSH access was configured. Data acquisition was performed on the Pi, while live visualization and post-processing were handled

on a remote laptop via a local Wi-Fi network. The Pi transmitted serialized data (depth values and projected height estimates) using a socket-based protocol. Feedback control, including audio alerts, was also handled locally, allowing a low-latency response independent of the main workstation. Additionally, a separate laptop was wirelessly connected to the Raspberry Pi using the TigerVNC client, enabling real-time monitoring of the RGB-D data stream and execution feedback during experiments. This separation of sensing and visualization minimized the load on the robot while enabling real-time feedback for the user.

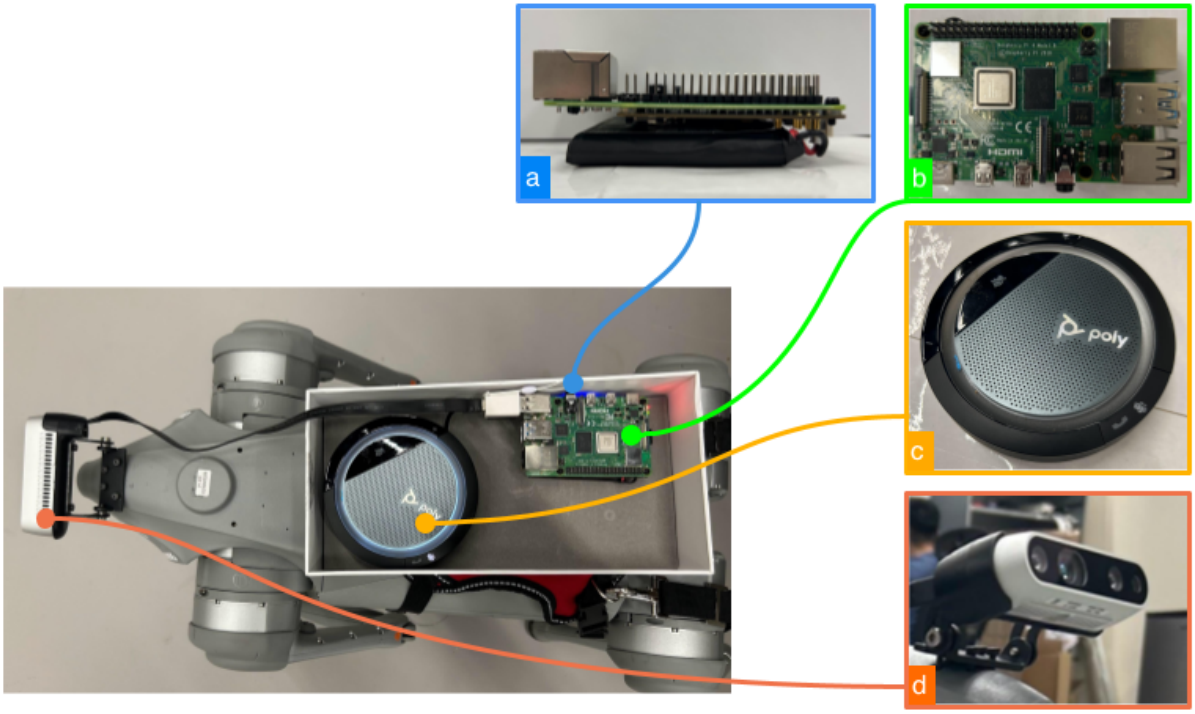

Figure 3.2: RGB-D hardware system mounted on the robot. The camera is angled upward, connected to a Raspberry Pi 4B with speaker and vibration motor support. The system contains (A) a PiSugar battery pack that supplies the Raspberry Pi with power without needing to be connected to a wired port. (B) A Raspberry Pi 4b with 8GB of RAM, this was chosen since it has 4 usb ports that are necessary to control both the camera and speaker. Subsequently 8GB were selected to allow the raspberry pi to process the image data as fast as possible to ensure that the system could quickly detect and report threats to the user. (C) A Poly USB speaker play an audio file to alert the user to threats. (D) Intel Realsense i435 Depth Camera to take in RGB and depth data to allow the system to determine wether an object is within range of colliding with the user.

The Unitree L1 LiDAR provides 360° horizontal and ±15° vertical scanning, delivering point cloud densities sufficient to resolve objects as small as 5 cm across in 2–5 meters. Point-cloud preprocessing includes downsampling, ground-plane removal, and normal estimation to facilitate segmentation. RViz2 was used to overlay bounding boxes on real-time data for visualization and

debugging.

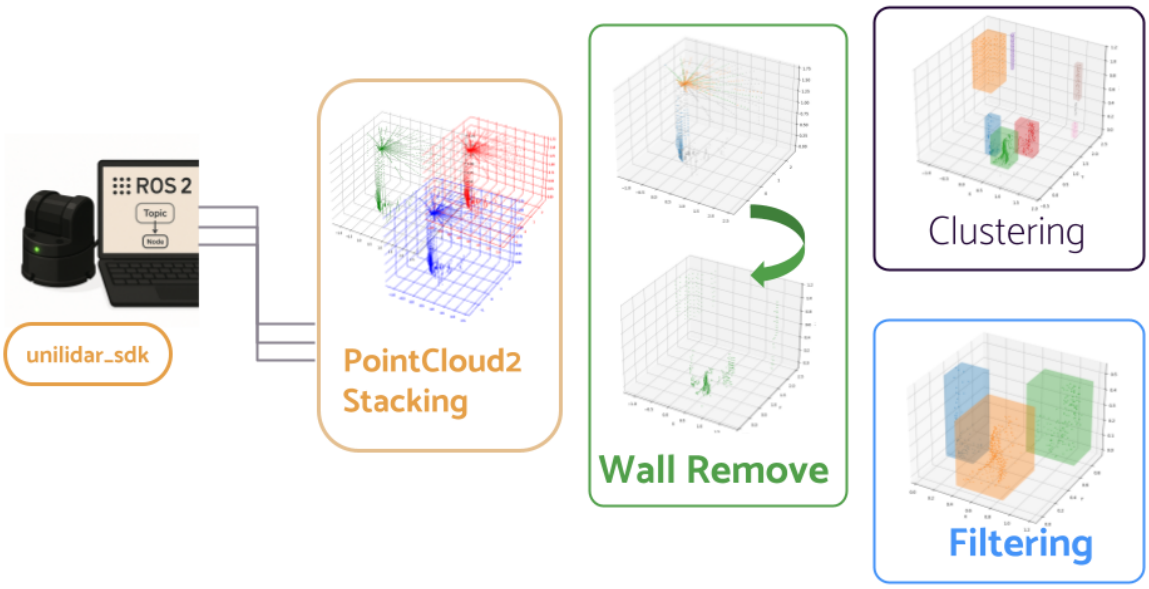

Figure 3.3: LiDAR data processing pipeline consisting of multi-frame stacking for temporal consistency, RANSAC-based plane removal to eliminate ground and wall surfaces, DBSCAN clustering to segment spatially coherent point groups, and height/distance filtering to isolate overhead obstacles for real-time detection.

The system adopts a modular ROS 2 architecture with separate nodes for perception, filtering, clustering, and visualization. RGB-D data are published via `/camera/depth/points`, while LiDAR nodes publish to `/unilidar/cloud`. An obstacle detection node subscribes to both streams, synchronizes temporal data, and publishes the resulting obstacle positions as bounding boxes in the `/obstacles` topic. All nodes support ROS 2 lifecycle management and parameter updates.

Each RGB-D depth pixel is projected in 3D coordinates. A fixed tilt angle θ is used to calculate the vertical component of the depth, defined as:

$$z_{adjusted} = d \cdot \cos(\theta)$$

Objects that intersect with a virtual threshold below 1.5 m from the ground are treated as hazards. Alerts are sent to the user through audio feedback. Depending on the distance of the closest pixel below the threshold, one of three audio files will play varying in pitch from low to

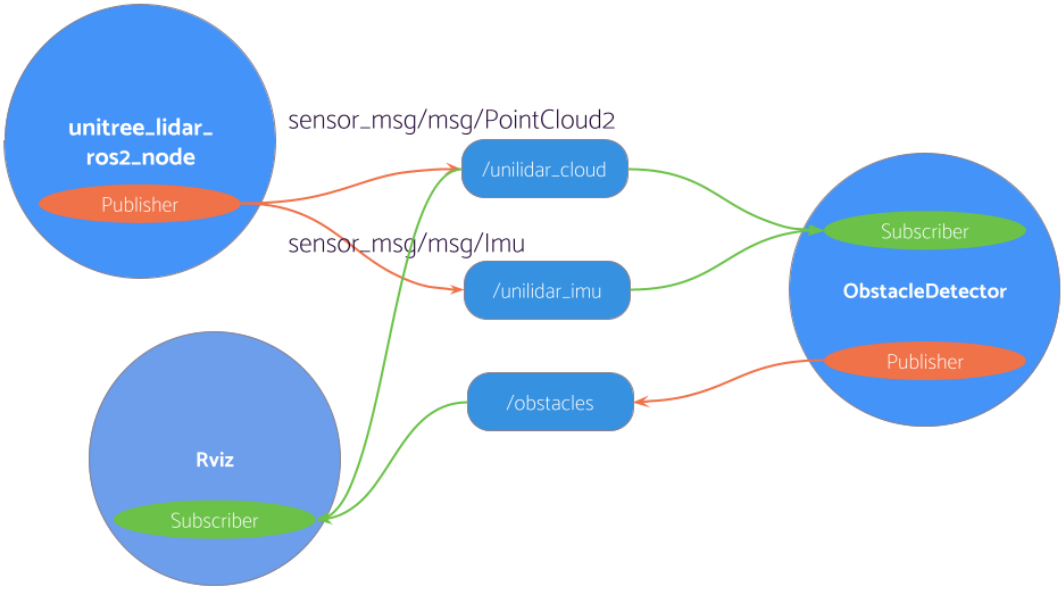

Figure 3.4: Overview of system integration using ROS 2. RGB-D and LiDAR data are processed in parallel pipelines and published for visualization and response.

high. The higher pitch file is played when the robot is closer to the detected obstacle, while the lower pitch file is played when the object is first detected within 1.5 m.

Point clouds are accumulated in 10 consecutive frames. RANSAC filters dominant planes using a 0.01 m threshold and removes segments with more than 300 points aligned to floor/wall normals. The remaining points are clustered using DBSCAN ($\varepsilon = 0.12$, minPts=10) and filtered by height (<2 m), proximity (<2.5 m), and density (>150 points/m^3).

3.5 Experimental Scenarios

To evaluate robustness, tests were conducted under varying conditions:

The performance of each system was compared in both indoor lighting and sunlight. Both the LiDAR and the RGB-D systems were able to perform consistently in each environment. The lighting seems to not affect either the accuracy or performance of the system.

The objects used were tree branches, a hanging broom, and a humanoid robot harness handle. LiDAR failed to detect the handle because of its smaller size, while the broom experienced noise

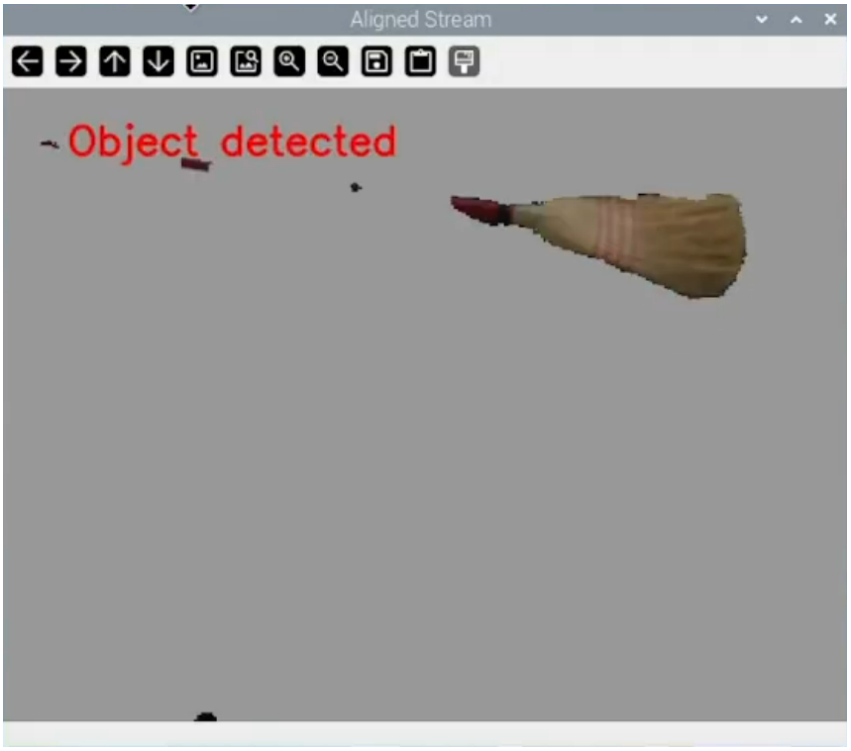

Figure 3.5: Segmentation of obstacle from objects outside the threshold. When 5 percent of pixels are within the 1.5 meter threshold the object is.

as a result of high reflectivity from surrounding obstacles, while RGB-D missed the same object when moving towards them at lower angles.

For dynamic testing, a hanging object was swung slightly during detection. The depth map of the RGB-D system fluctuated, whereas LiDAR preserved object shape due to point-cloud aggregation.

3.6 Results and Discussion

Due to a lack of time, both systems were evaluated by their First Detection Distances from the front of the Unitree Go2. This distance is determined by the distance the capturing device is horizontally from the obstacle. This was chosen since the user's body is in line with the obstacle, meaning that I could not measure the true diagonal distance from the capturing device. This method allowed us to determine which system was able to alert the user in the most time to respond to the current obstacle.

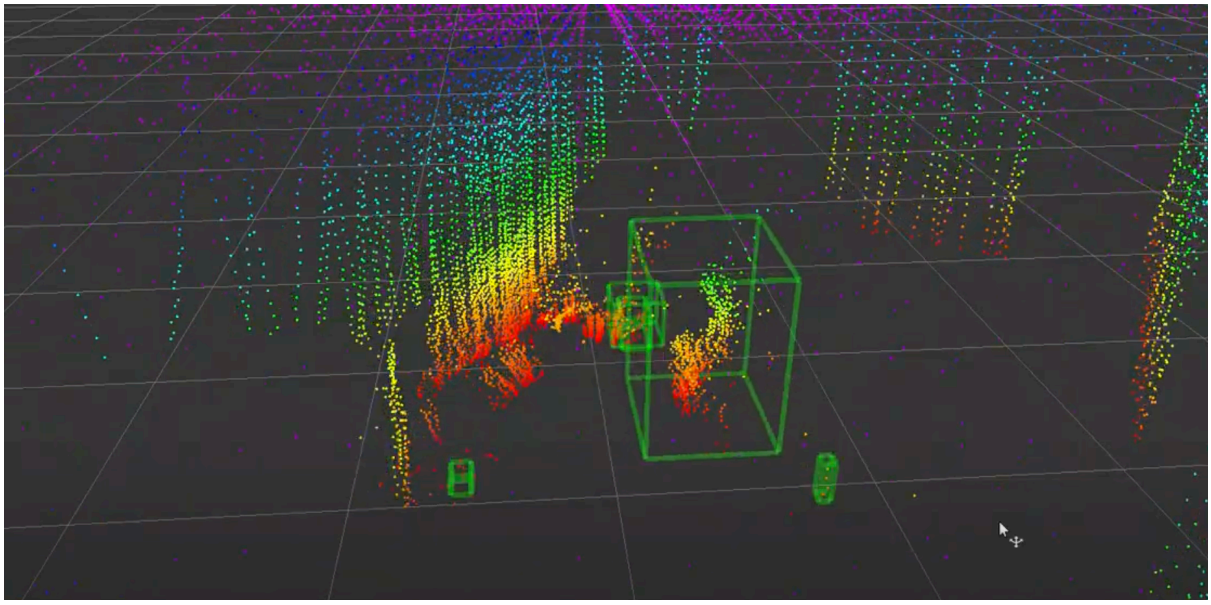

Figure 3.6: LiDAR-based point cloud segmentation visualized in RViz2. Raw data is processed through plane removal (RANSAC), DBSCAN clustering, and height-based filtering. Resulting obstacle clusters are enclosed in bounding boxes for real-time visualization.

The evaluation was carried out in an indoor hallway using hanging objects (brooms, hanging handles) and measured the detection range, responsiveness, and visualization output. Data were captured using the ROS 2 pipeline and Open3D tools.

The RGB-D system reliably detected overhead obstacles up to 3.9 feet at 45° tilt, while LiDAR achieved 5 feet with greater consistency. RGB-D struggled in low light, whereas LiDAR remained unaffected.

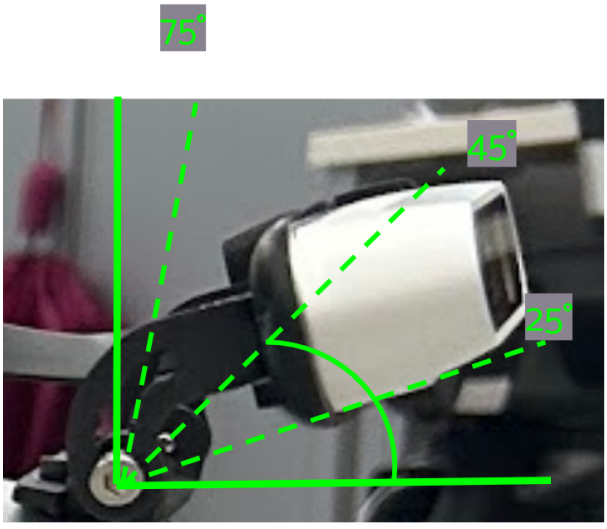

Figure 3.7: Approximate angles above the Unitree Go2's head selected for testing the RGB-D systems effectiveness

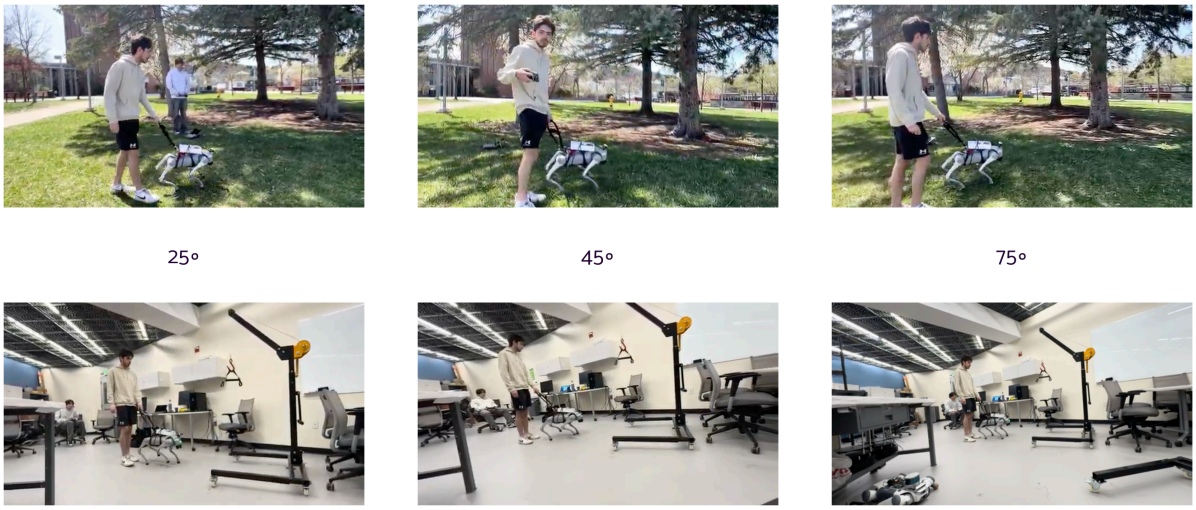

Figure 3.8: Depth map visualization from RGB-D camera under outdoor lighting. Depth distortions increased with distance and angle.

The boundaries of the LiDAR clusters were clearly visualized. The RGB-D system generated discrete warnings through a mounted speaker. Multi-object scenes were successfully handled by the LiDAR setup, while RGB-D occasionally missed higher objects due to field-of-view limits.

In addition, the detection delay between the appearance of the obstacle and the system response was measured. The RGB-D system responded in 0.4 seconds, suitable for audio feedback. The LiDAR pipeline required 0.8 seconds to accumulate and process frames, but offered more stable results over time.

Furthermore, false positives were observed when the RGB-D camera pointed too steeply upward, causing ceiling detection. Adding a narrow region of interest (ROI) reduced these errors. For LiDAR, side reflections from reflective surfaces (such as a TV or whiteboard) sometimes formed small phantom clusters, which were filtered out based on size and density.

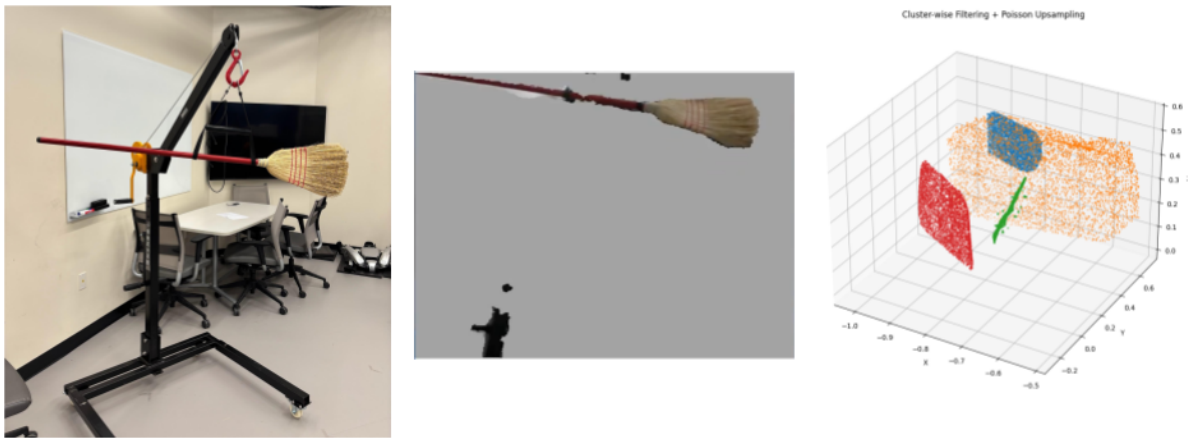

Figure 3.9: Suspended broom used as an overhead obstacle during indoor trials. The obstacle height was approximately 4 ft 2 in.

Table 3.1: Comparison of RGB-D and LiDAR Approaches

Metric	**RGB-D**	**LiDAR**
Range	3.9 ft	5 ft
Lighting Dependence	High	Low
Precision	Medium	High
Feedback Support	Yes	No
Cost	Low	High
Integration Effort	Low	Moderate

3.7 Limitations and Future Integration Discussion

The RGB-D camera system is well suited for short-range, lightweight applications, especially those requiring immediate user feedback. LiDAR offers robustness and range, but requires higher processing and cost. Future designs should combine both systems for optimal performance.

Although promising, both systems have limitations. The effective range of RGB-D and the dependence on ambient light restrict its use in outdoor or dim corridors. Its low frame rate (6–15 fps) under depth load also limits the real-time mapping.

LiDAR offers spatial fidelity, but lacks semantic context. A hybrid architecture can exploit RGB for classification and LiDAR for depth and shape. For instance, projected RGB features onto point cloud surfaces could allow for material-aware segmentation.

Future Investigations Include:

- Depth-temporal filtering (e.g., Kalman smoothing for RGB-D)
- Low-power onboard Jetson or EdgeTPU deployment
- ROS 2 lifecycle node modularization

4 Autonomous Quadruped Navigation through Voice Commands

The goal was to create a completely autonomous system that is capable of moving and responding to external stimuli on its own. Accepting navigation goals by translating human speech into target locations on a previously loaded map.

4.1 SLAM Setup

Since the robot already had ROS2 Foxy loaded onto the system, I searched for a SLAM navigation package on GitHub that could generate maps that work with Nav2. The package was developed for UnitreeGo2 and uses ROS2 Foxy and included modules for perception, navigation, and a plugin for the use of SLAM known as SLAMToolBox [20]. All of these modules were connected to a central launch file that could run all of them simultaneously after being built. When run they will open a program called RViz which is a visualization tool for ROS2, allowing us to see the current map as it is being built.

4.2 Wireless Setup

Originally, the UnitreeGo2 jetson could not connect to outside WiFi connections, limiting our ability to make api requests to LLMs. To enable this, it was necessary to enable WiFi and a wireless connection to the robot. In addition, ensuring that SLAM navigation can be done from any distance within range of the WiFi network. When in communication with the UnitreeGo2, it

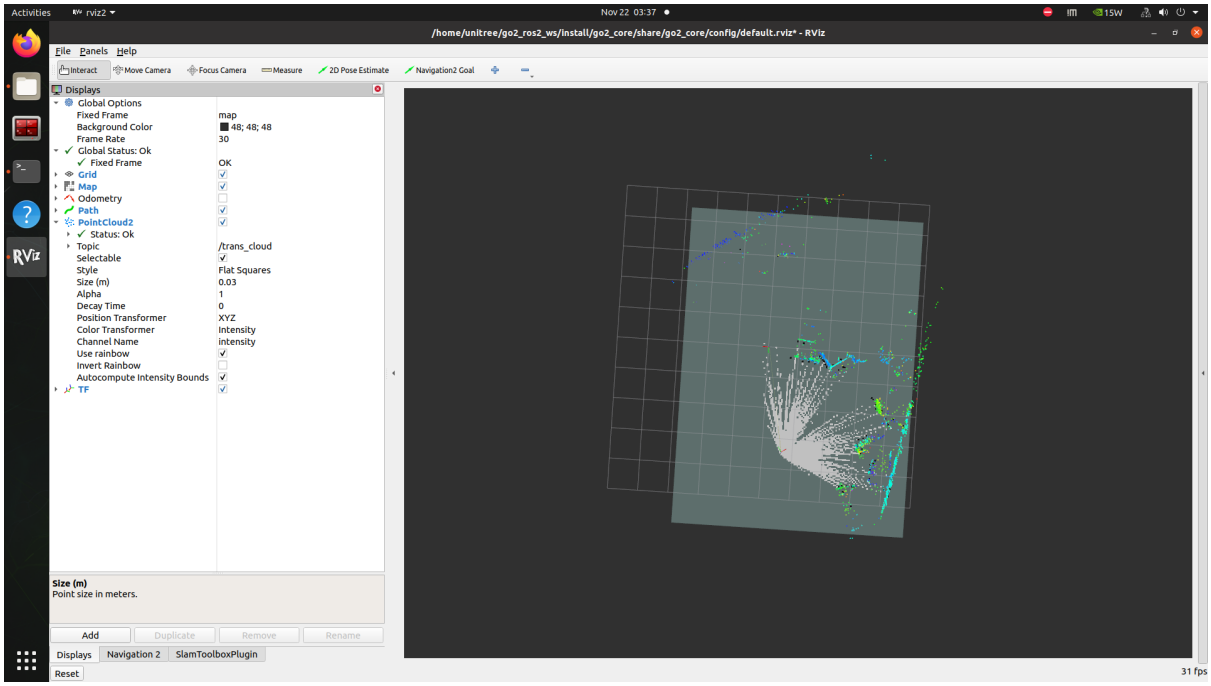

Figure 4.1: The Displays tab denotes what is viewable in the curernt scene in RViz. For the purposes of running SLAM I add a point cloud 2 display to visualize the LiDAR data and a the TF display to see the robots current assumed position. This works in conjunction with the 2D Pose Estimate and Navgation2 Goal buttons. 2D Pose Estimate: allows the robots position to be manually set on the graph which can be used using deserialization, Navigation2 Goal: creates a path from the robots current estimated position to the position clicked on the map and moves the robot along that path

is possible to connect to the jetson computer over an Ethernet cable. This was how the majority of the initial tests were performed. The first step was to be able to connect the robot wirelessly to avoid any limitations associated when using a finite length cable. Through testing, I found that the UnitreeGo2 and jetson are connected through an Ethernet cable, and when that connection is interfered with, it causes the jetson to be unable to get data from any ROS2 topics. When trying to access the internet on the jetson through Ethernet, the connection to the robot would be replaced. Since, as stated previously, the jetson does not have a built in WiFi adapter, I used a Netgear WiFi adapter and a mobile hotspot to allow it to have access to the internet in addition to its Ethernet connection. Next, since I need to be able to control the jetson computer, I installed NoMachine a Virtual Network Computing (VNC) viewer onto both the jetson and my laptop. This enables me to connect over WiFi using the NX protocol and control the jetson as if I had connected to it directly.

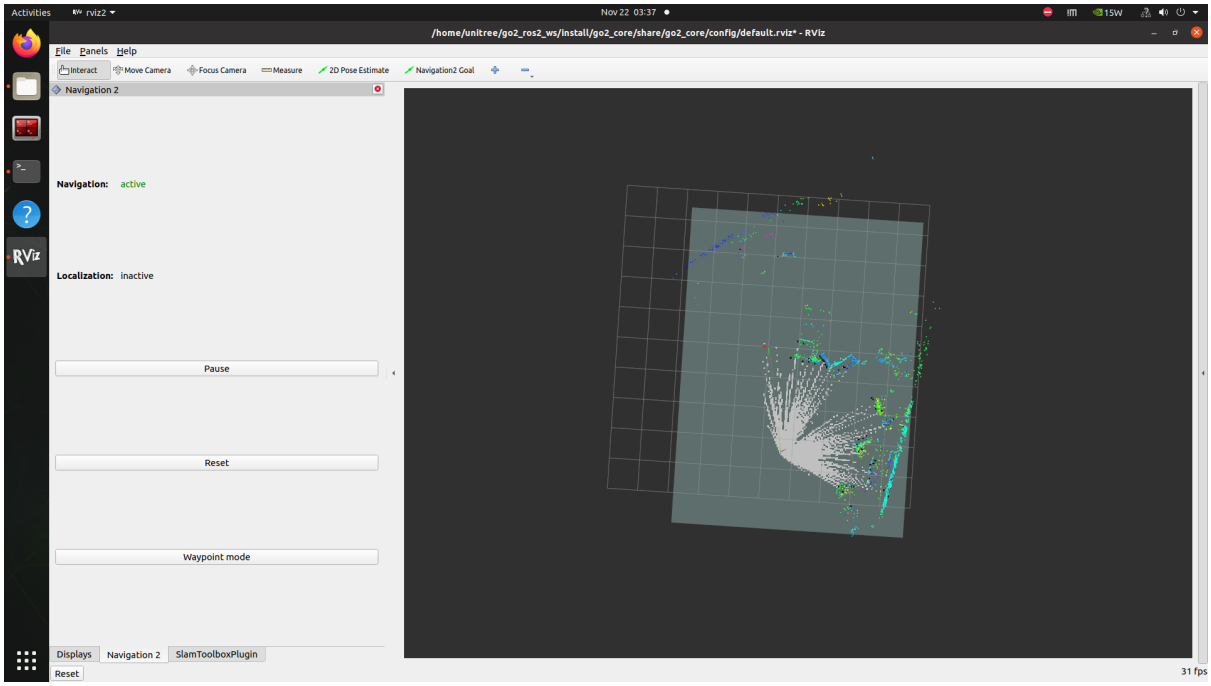

Figure 4.2: The Navigation section allows paths current navigation goals to be paused or reset. Additionally waypoints can be manually set to create complex paths.

4.3 Map Generation

To generate the maps used for navigation, I built the SLAM packages and walked the robot around the floor of the building our lab is on using LiDAR to generate the maps. Using the SLAM plugin to serialize a couple of different maps of varying sizes. Some of these maps were built before the WiFi adapter and NoMachine was implemented, so they only map the lab. Once I was able to connect to the robot wirelessly, I controlled the robot to map the entire floor resulting in the map shown in Figure 4.4.

4.4 Voice Agent

To enable handler control of robot navigation, I developed a voice agent to process voice commands and translate them into coordinates on the loaded map. This allowed the user to speak into a microphone connected to the robot and ask to be taken to the bathroom. The robot will then ask for confirmation from the handler; if conformation is provided by the handler, the

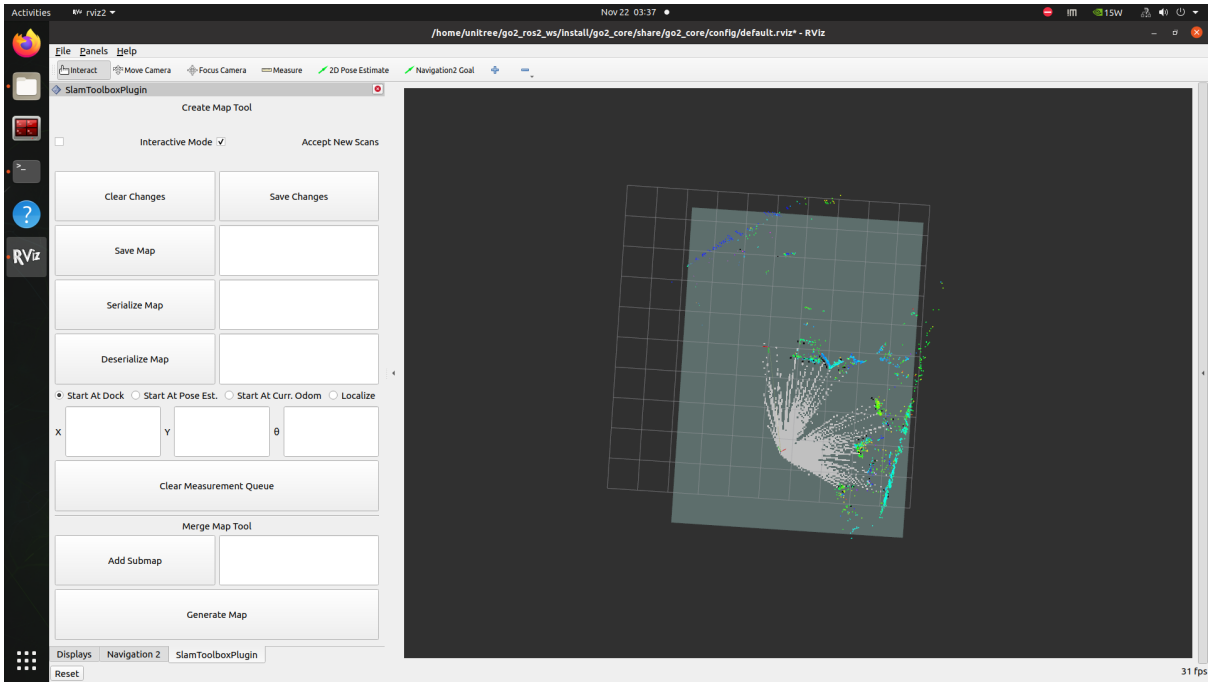

Figure 4.3: The SLAM Plugin allows the robot to save maps that are created by LiDAR and loaded back into the environment to be used to set navigation goals. Save Map: saves a pgm file of the name placed within the box, Serialize Map: creates a posegraph that can be used to load the map during deserialization, Deserialize Map: Loads serialized map of the name provided at the current estimated pose, the dock location, the current odometry or it will try to localize

robot will send a navigation goal to the designated target and begin navigating the handler. To accomplish this, the robot utilized WiFi and OpenAi's API to prompt ChatGPT, providing it with the context that it is a guide dog within a building with the goal of guiding the user to a designated location. The prompt also limits the types of responses and locations that can be generated to two predefined lists, one labeled Targets (for possible destinations) and the other labeled Sentences (for responses). The responses range from an initial welcome statement explaining the robots role, to asking for conformation on a navigation location. Each response is represented as a json format, which enabled easy processing of the LLM output.

To integrate with the SLAM toolbox, the Voice Agent was designed as a custom ROS2 node, represented as a python class, that is spun up after the map has been loaded. The agent is split into two major components, the listener and the goal sender. The listener takes in audio input from the user and converts that into text that can be processed by the ChatGPT. This is accomplished by using Google's speech recognizer python library to convert audio to text. First, the recognizer adjusts itself to the current level of ambient noise in the environment before

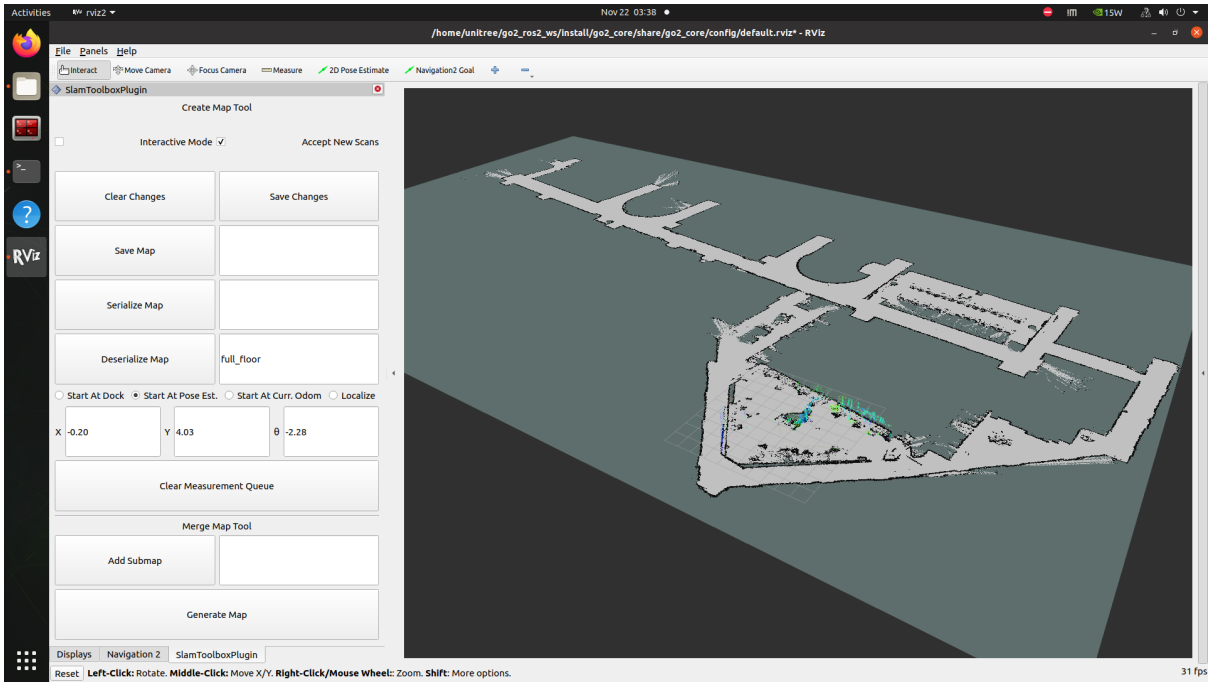

Figure 4.4: Fully mapped floor of the lab building using LiDAR, utilized during testing

taking external audio as input. If no input is provided, the listener will start the loop again. When speech is detected, its converted to text and appended to the initial prompt as a message from the user. Then the model is prompted and its response is decomposed into the json format described previously to get the models spoken response and target. This process is repeated until the user confirms the target location to which they would like to travel. Once the selection has been made, the target is sent to the goal sender through the send goal function.

The goal sender is an ROS2 node that integrates with Nav2. Using the same command used by the Navigate2Pose button within RViz to send location goals. Each target location is saved in a python dictionary where the target name is the key and the (x,y,z) coordinates are saved as the value. After a target is selected by the voice agent, a second thread is started to run the goal sender concurrently with the voice agent. This allows navigation to continue without interrupting communication between the handler and the robot. This allows the goal to be handled asynchronously using a callback function to log if the goal was reached or not. If the returned status was a success, the voice agent will tell the handler that they have reached the goal.

4.5 Limitations

The navigation tool and the voice agent together were capable of navigating around a predefined map with little to no difficulty or error. Through human robot interactions the Voice Navigation agent is capable of generating navigation routes completely autonomously. However, there are limitations with the current implementation of this combined system. Firstly, the SLAM navigation tool aborts prematurely when faced with a new obstacle placed in its path. This was strange since the robot was capable of avoiding obstacles that were placed in front of initially becoming prevalent after introducing the voice agent. This poses a problem, as the system should be able to avoid small local obstacles in the environment without ending the current global path. The navigation tool also has an issue with generating paths through walls. This was due to the maps containing small holes that were not filled properly during map generation. Causing the robot to change course rapidly towards a wall before realizing that the new path is not a suitable route. Stopping for a moment to recalculate a new path fairly similar to the original. This can lead to disorientation and confusion when following behind the robot dog. In addition to this, the Voice Agent has trouble recognizing words spoken to it taking a few attempts to pick up what the handler is trying to say. This is most likely due to a hardware issue, as the microphone is located on the back of the robot dog, which makes a lot of noise. As a result of this, the handler must lift the microphone to their mouth or have to reach down and speak directly into it. Making using the voice agent inconvenient and slow in its current form.

5 Conclusions and Future Work

This thesis provided a comparative framework for evaluating RGB-D and LiDAR sensors and developed a voice activated navigation system for robotic guide dog applications. Displaying the process in developing the hardware and software required to create systems for both real and robotic guide dogs. Performing extensive experimentation to determine the strengths and weaknesses of both RGB-D cameras and LiDAR for detecting obstacles both in and outdoors. The results of the comparison show clear trade-offs between precision, usability, and deployment between both LiDAR and RGB-D detection systems.

The current systems are capable of detecting overhead obstacles within 2 meters, but are completely disconnected from the navigation and voice agent. Using a separate voice script to alert the handler of incoming obstacles. For future work, I want to integrate the RGB-D overhead obstacle detector into the voice navigation system. The RGB-D is capable of working in outdoor environments, allowing the LiDAR to focus on indoor SLAM. Additionally, the current voice navigation system can only use one pre-loaded map. In future work I will enable map switching capabilities, increasing the number of potential locations, and improve the voice agents ability to pick up the handlers voice.

Bibliography

[1] Marziyeh Bamdad, Davide Scaramuzza, and Alireza Darvishy. Slam for visually impaired people: a survey. https://arxiv.org/abs/2212.04745, 2024.

[2] Tyler Bell, Beiwen Li, and Song Zhang. Structured light techniques and applications. https://onlinelibrary.wiley.com/doi/full/10.1002/047134608X.W8298, 2016.

[3] L. Brown and T. Green. Multi-sensory guidance system using orb-slam and yolo. https://www.mdpi.com/2078-2489/13/7/343, 2022. Accessed: 2024-02-10.

[4] David DeFazio, Yohei Hayamizu, and Shiqi Zhang. Learning quadruped locomotion policies using logical rules, 2024.

[5] David DeFazio, Eisuke Hirota, and Shiqi Zhang. Seeing-eye quadruped navigation with force responsive locomotion control, 2023.

[6] David DeFazio, Hrudayangam Mehta, Meng Wang, Ping Yang, Jeremy Blackburn, and Shiqi Zhang. Vision language models can parse floor plan maps, 2025.

[7] F. Gao, J. Li, and X. Zhang. Autonomous navigation for assistive robots: A lidar-based approach, 2024.

[8] GeeksForGeeks. Rgb-d images: A comprehensive overview. https://www.geeksforgeeks.org/computer-vision/rgb-d-images-a-comprehensive-overview/, 2024.

[9] Jordan S. K. Hu, Tianshu Kuai, and Steven L. Waslander. Point density-aware voxels for lidar 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page to appear, 2022. Available: *https://arxiv.org/abs/2203.05662*.

[10] H Kim and R Patel. Advanced perception in guide robots using 3d lidar and sensor fusion. https://ieeexplore.ieee.org/document/12345678, 2023. Accessed: 2024-02-10.

[11] Feiya Li, Chunyun Fu, Dongye Sun, Jian Li, and Jianwen Wang. Sd-slam: A semantic slam approach for dynamic scenes based on lidar point clouds. *arXiv preprint*, 2024. Available: *https://arxiv.org/abs/2402.18318*.

[12] The Drone Life. What is a drone lidar point cloud? (.las/.laz). https://thedronelifenj.com/drone-lidar-point-cloud/, 2023.

[13] Microdrones. Lidar point cloud quality control: Automating accuracy and precision testing. https://www.microdrones.com/en/content/lidar-point-cloud-quality-control-automating-accuracy-and-precision-testing/, 2023.

[14] Søren Riisgaard and Morten Rufus Blas. Slam for dummies. https://dspace.mit.edu/bitstream/handle/1721.1/119149/16-412j-spring-2005/contents/projects/1aslam_blas_repo.pdf, 2005.

[15] SmartOne.ai. A complete guide to 3d lidar point cloud data. https://smartone.ai/blog/a-complete-guide-to-3d-lidar-point-cloud-data/, 2023.

[16] Guangming Wang, Xinrui Wu, Shuyang Jiang, Zhe Liu, and Hesheng Wang. Efficient 3d deep lidar odometry. *arXiv preprint*, 2021. Available: *https://arxiv.org/abs/2111.02135*.

[17] Leah A. Wasser. The basics of lidar - light detection and ranging - remote sensing. https://www.neonscience.org/resources/learning-hub/tutorials/lidar-basics#:~:text=LiDAR%20is%20an%20active%20remote,%2C%20pitch%2C%20and%20yaw)., 2024.

[18] Yida. What is a time of flight sensor and how does a tof sensor work? https://www.seeedstudio.com/blog/2020/01/08/what-is-a-time-of-flight-sensor-and-how-does-a-tof-sensor-work/, 2019.

[19] Xingguang Zhong, Yue Pan, Cyrill Stachniss, and Jens Behley. 3d lidar mapping in dynamic environments using a 4d implicit neural representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page to appear, 2024. Available: *https://arxiv.org/abs/2405.03388*.

[20] Andy Zhuo. Go2 ros2 toolbox. https://github.com/andy-zhuo-02/go2_ros2_toolbox, 2025.