ERDC TR-23-17

US Army Corps
of Engineers®
Engineer Research and
Development Center

Tactical Geospatial Information Capabilities

Low Size, Weight, Power, and Cost (SWaP-C) Payload for Autonomous Navigation and Mapping on an Unmanned Ground Vehicle

Osama Ennasr, Brandon Dodd, Michael Paquette,
Charles Ellison, and Garry Glaspell

September 2023

Engineer Research and Development Center

The US Army Engineer Research and Development Center (ERDC) solves the nation's toughest engineering and environmental challenges. ERDC develops innovative solutions in civil and military engineering, geospatial sciences, water resources, and environmental sciences for the Army, the Department of Defense, civilian agencies, and our nation's public good. Find out more at www.erdc.usace.army.mil.

To search for other technical reports published by ERDC, visit the ERDC online library at https://erdclibrary.on.worldcat.org/discovery.

Low Size, Weight, Power and Cost (SWaP-C) Payload for Autonomous Navigation and Mapping on an Unmanned Ground Vehicle

Osama Ennasr, Brandon Dodd, Michael Paquette, and Garry Glaspell

US Army Engineer Research and Development Center (ERDC)
Geospatial Research Laboratory (GRL)
7701 Telegraph Road
Alexandria, VA 22315-3864

Charles Ellison

US Army Engineer Research and Development Center (ERDC)
Information Technology Laboratory (ITL)
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

Final Technical Report (TR)

Prepared for Headquarters, US Army Corps of Engineers
Washington, DC 20314-1000

Under PE 0602146A / Project AT9 / Task 01, "Tactical Geospatial Information Capabilities"

Abstract

Autonomous navigation and unknown environment exploration with an unmanned ground vehicle (UGV) is extremely challenging. This report investigates a mapping and exploration solution utilizing low size, weight, power, and cost payloads. The platform presented here leverages simultaneous localization and mapping to efficiently explore unknown areas by finding navigable routes. The solution utilizes a diverse sensor payload that includes wheel encoders, 3D lidar, and red-green-blue and depth cameras. The main goal of this effort is to leverage path planning and navigation for mapping and exploration with a UGV to produce an accurate 3D map. The solution provided also leverages the Robot Operating System.

Contents

Figures, Tables, and Listings

Figures

Tables

Listings

Preface

This study was conducted for the US Army Corps of Engineers under Project Element 0602146A, Project AT9, Task 01, "Tactical Geospatial Information Capabilities." The technical monitor was Dr. Jean Nelson.

The work was performed by the Data Representation Branch of the Topography Imagery and Geospatial Research Division, US Army Engineer Research and Development Center (ERDC), Geospatial Research Laboratory (GRL), with assistance from personnel at the ERDC Information Technology Laboratory (ITL). At the time of publication, Mr. Vineet Gupta was branch chief; Mr. Jeff Murphy was division chief; and Dr. Austin Davis was the technical director. The deputy director of ERDC-GRL was Ms. Valerie L. Carney, and the director was Mr. David R. Hibner. The deputy director of ERDC-ITL was Dr. Jackie S. Pettway, and the director was Dr. David A. Horner.

The authors would like to acknowledge the following individuals for their contributions to this project: Mr. Steven Bunkley, Mr. Chuck Dickerson, and Mr. Richard Brown. The authors would also like to acknowledge Mr. Richard Curran and Mr. Phillip Devine for their help in collecting and processing the point clouds from handheld survey lidar systems for comparison.

The commander of ERDC was COL Christian Patterson, and the director was Dr. David W. Pittman.

1 Introduction

1.1 Background

Active use of mobile robots is increasing rapidly in a variety of different fields, including mapping and exploration of environments denied GPS (Zakiev et al. 2019). To maximize applicability, the mobile robot must be able to perform simultaneous localization and mapping (SLAM) using its onboard sensors and processor while continuously path planning based on the collected data about its dynamic environment.

Autonomous navigation in the Robot Operating System (ROS) typically works in tandem with the map generated by SLAM (Castellanos et al. 2001; Wang et al. 2016; Jacobson et al. 2018). The most common type of map used for path planning and navigation is a 2D costmap that represents the world as a 2D grid and then attaches a cost to each grid cell depending on sensor data. While each cell in the costmap can have one of 255 different cost values, the underlying planning structure that it uses is capable of representing only 3. Specifically, each cell in this structure can be either free, occupied, or unknown.[1] Two costmaps are typically maintained: a global (more static) costmap used for computing a long-term path plan based on overall map and a local (more real-time) costmap used to navigate around dynamic obstacles not available in the global map. To accomplish this, ROS leverages a layered approach. At the bottom of the stack is static layer (map layer), which is the 2D map maintained by the robot. The second layer (obstacle layer) comprises obstacles currently detected by the sensors and is overlaid onto the static layer. The third layer (inflation layer) generates a buffer around the obstacles as a safety measure to keep the robot from running into them. Finally, the combination of all layers refers to a costmap.

For autonomous exploration, common approaches implement greedy frontier-based exploration methods (Keidar et al. 2014; Wang et al. 2011). In these methods, the boundary between known (free or occupied) and unknown points (referred to as a frontier) becomes a possible navigation

[1] http://wiki.ros.org/costmap 2d

goal. Associated with each frontier is a cost for clearing the frontier that includes factors such as distance, orientation, and size. The exploration node greedily selects and publishes the lowest-cost frontier as a navigation goal. The path-planning node subscribes to the navigation goals and attempts to steer the robot toward goals while keeping the cost at a minimum. This process repeats for all existing frontiers and with the addition of new frontiers.

Therefore, successful path planning and navigation for mapping and exploration of unknown environments in ROS require three main components. First, accurate odometry and mapping (frequently achieved through SLAM). Second, an efficient path-planner is required to avoid static and dynamic obstacles in the environment. Finally, for exploration, a method for outputting navigation goals to the path-planner must be implemented.

This report builds on the previous work in developing and testing a low-size, weight, power, and cost (SWaP-C) robotic platform for mapping and exploration of unknown interiors and subterranean (SubT) environments. Specifically, we present the hardware and software improvements to our robotic mapping solution in Glaspell et al. (2020) and apply them to autonomous navigation and exploration (Christie et al. 2021). Furthermore, the team identified and fused ROS packages that address each requirement to ensure successful path planning and navigation for mapping and exploration of unknown environments. The combination of these packages results in a practical solution that allows autonomous navigation and mapping exploration of 2D environments.

1.2 Objective

This report addresses the Army Multi-Domain Intelligence FY21-22 Science and Technology Focus Areas. Specifically in the sensors section, we feel this report correlates with the following need: “Novel combinations of sensors and robotic platforms that can not only move across terrain, but maneuver to sense.” We also feel this work addresses the statement, “Wars will be fought at hyper speed and scale, dominated by technologies such as robotics and autonomous systems, machine learning (ML), and artificial intelligence (AI) capabilities, which are widely available, packaged, and ready for use” (Department of the Army 2021).

1.3 Approach and Scope

The goal is to provide the warfighter a cost-effective solution that provides situational awareness, in near real time, of the interiors of buildings and SubT environments. To address the gaps listed above, it will be necessary to successfully merge a mapping technology with a robotic platform capable of navigating through a dynamic environment with limited computational and power resources. This report focuses on the development of a robot that can navigate and map unknown 2D environments by planning routes to specific points of interest or using an automated exploration strategy. This is achieved by leveraging ROS Melodic[2] and its available packages.

Improvements from the previous work to the sensor payload and software package were implemented, and experimental studies were conducted to characterize the performance and accuracy of our SLAM approach. Additionally, we introduced new approaches to path planning and navigation to facilitate operation through waypoint navigation. Finally, frontier-based exploration was implemented in order to perform autonomous exploration of unknown 2D environments.

The mapping solution was tested in multiple locations to generate accurate maps including: (1) the bunker at the Army Geospatial Center (hereafter "AGC bunker"), (2) a conduit in the Mojave River Dam, and (3) the experimental shoot house in a hangar at Fort Belvoir. The robotic platform was also presented in the Maneuver Support, Sustainment and Protection Integration eXperiments (MSSPIX) 2022 hosted by the Army's Maneuver Support and Sustainment Capability Development Integration Directorates. At this event, soldiers used three different modes of operation: teleoperation with joystick, waypoint navigation, and autonomous exploration. During these tests, the value of waypoint navigation and autonomous exploration was highlighted, as they greatly facilitated exploration of unknown environments.

[2] http://wiki.ros.org/melodic

The main payoff is the development of a low-cost, low-power mobile survey and mapping sensor suite capable of autonomously navigating dynamic 2D environments. The platform is capable of providing real-time, or near real-time, situational awareness of the indoor or SubT environments being explored.[3]

The lidar data collected can provide input to terrain processing and analysis tools and mission planning applications to inform situational awareness, operational security, and maneuverability. Finally, the collected data could be utilized in Army training and virtual reality (modeling and simulations) applications.

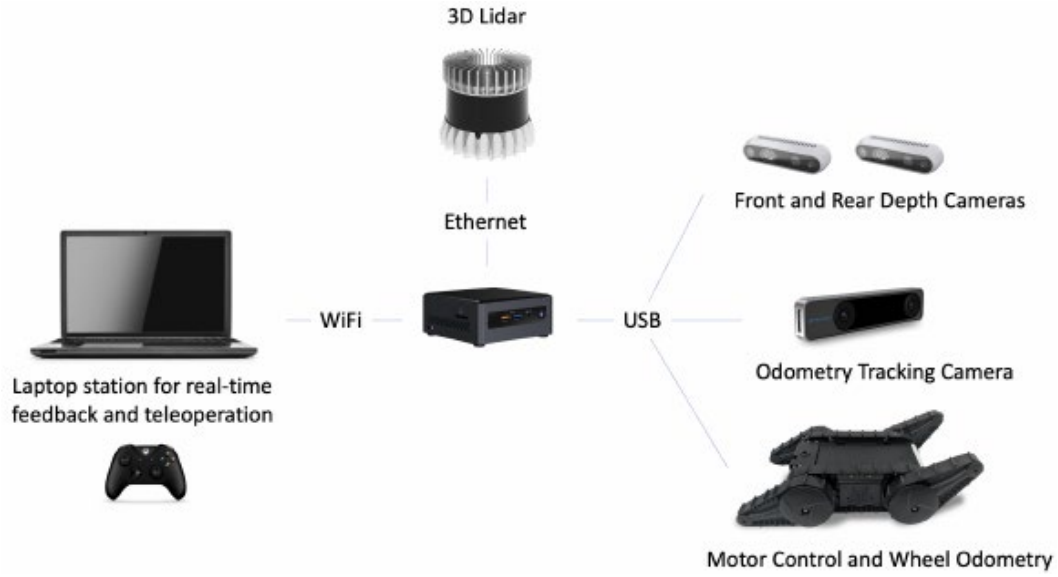
[3] In cases where there is limited range or bandwidth, near real-time results can be obtained within a few minutes.

2 Hardware Setup

2.1 Overview

This chapter describes the details of the hardware (HW) for our robotic solution, including architecture, platform, processor, and sensors. Figure 1 shows all HW components (including processor and sensors with connections) for our GPS-denied mapping system.[4] The major HW components of our solution include the unmanned ground vehicle (UGV) robotic platform (RoverRobotics flipper robot), the onboard processor for navigation, mapping, and communication (Intel NUC7i5), 32-channel 360° lidar (Ouster OS0-32), a front-facing odometry tracking camera (Intel Realsense T265), and forward-facing and backward-facing red-green-blue and depth (RGB-D) cameras (Intel Realsense D435) (Figure 1).

Figure 1. Hardware setup.

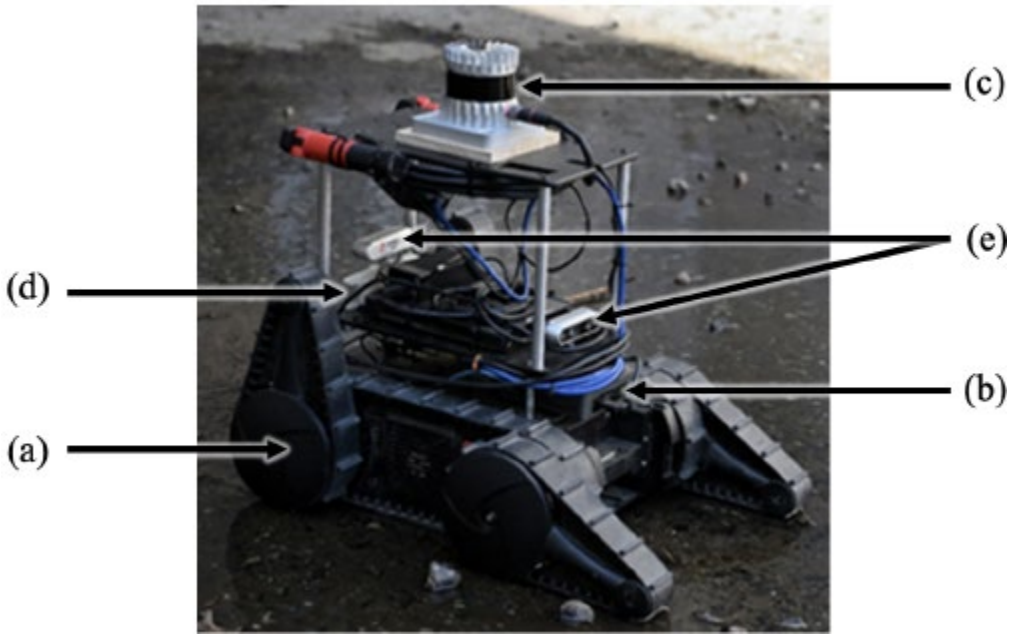


The robot was configured to broadcast a Wi-Fi network on startup using the embedded Wi-Fi chip on our processor. This allows the operator to have direct control of the robot and guide it through the remote joystick or by placing navigation waypoints for the robot using the laptop station. Moreover, it offers a connection for real-time feedback over the Wi-Fi

[4] Component stock photos are taken from manufacturer websites (i.e., Intel, Ouster, and RoverRobotics).

range to offer better situational awareness. However, under the autonomous exploration mode, this laptop station is not needed, and near real-time (within a few minutes) 3D point clouds can be obtained after the robot completes its exploration mission successfully and returns to its starting position. Figure 2 shows the finalized robot with all HW attached to it. For low-light environments, we mounted flashlights to help illuminate the environment and reduce drift from the visual odometry tracking camera. In the following subsections, we summarize the details of each major HW component.

Figure 2. Final robot.



2.2 Platform

The RoverRobotics with treads and active flippers was selected as a test platform for this project as it fit the low SWaP-C profile we were looking for. We acknowledge that this specific robotic platform does not offer the same robustness that can be found in bigger platforms or legged robotics, limiting the types of environment that can be explored.

However, it was a good option to evaluate our SLAM, path-planning, and navigation approaches. While this robot cannot carry the same payload and battery capacity as the one reported in Glaspell et al. (2020), it allowed us to reduce the overall footprint, weight, and cost of our autonomous robotic mapping solution. Table 1 shows the specifications for this robotic platform.

Table 1. Unmanned ground vehicle (UGV) specifications.

Description	Value
Dimensions	62 × 39 × 25.4 (cm)
Weight	11 kg
Max speed	2.5 m/s
Payload capacity	100 kg
Ground clearance	8 cm
IP rating	IP67

These robots come equipped with a wheel odometry sensor that aids in robot localization. The robot's control board is connected to the processor over USB and allows for reading the wheel odometry reading as well as controlling the robot's motors. For accurate wheel odometry, the friction coefficient and weight parameters for the robot's ROS driver need to be accurately tuned depending on the type of surface the robot is traveling over (tile, carpet, grass, etc.). This poses a challenge when relying solely on the robot's wheel odometry for localization, as significant drift can occur when the driving conditions change from calibrated values. Therefore, our system fuses these wheel odometry with additional sensor information, resulting in a more robust odometry solution.

2.3 Processor

An Intel NUC7i5 mini PC was selected to handle all the mapping, navigation, exploration, and communication tasks. This unit was ideal for our low SWaP-C solution as it offered good computational capabilities with low power consumption and a small form factor. Table 2 lists the specifications of the processor used for our solution.

Table 2. Processor specifications for the NUC7i5 computer used for this work.

Description	Value
Dimensions	11.4 × 3.5 × 11.1 cm
Weight	0.9 kg
Processor	7th Generation Intel Dual-Core i5-7260U 2.2 GHz
RAM	8 GB
Storage	128 GB solid-state drive
Connectivity	1× Thunderbolt 3, 4× USB 3.0, Bluetooth, Wi-Fi, Gigabit ethernet

The robot's processor was set up to host additional devices (e.g., laptop, tablet, etc.) for real-time monitoring and remote operation purposes. However, as we discuss in the following chapters, all computations for mapping and autonomous exploration operations are performed locally on the robot, allowing for operation in areas with limited connectivity or bandwidth.

2.4 3D Lidar

The Ouster OS0-32 unit is a high-resolution multibeam flash lidar suited for indoor and outdoor use. We rely heavily on lidar in our solution as it offers a long range and accurate measure that is used for localization, navigation, and mapping. Table 3 shows the specifications of the lidar unit used in our solution. The lidar was connected to the processor over gigabit ethernet to ensure adequate bandwidth for transferring the point clouds generated by the sensor.

Table 3. Specifications for the Ouster OS0-32 3D lidar sensor used on the robot.

Description	Value
Dimensions	8.7 × 8.7 × 7.4 cm
Weight	0.45 kg
Vertical resolution	32 channels
Horizontal resolution	2,048
Range	50 m
Vertical FOV	90°
Precision	±1.5 cm–5 cm
Rate	10 Hz

By default, the Ouster sensor requires a wired connection hosted by the onboard processor for data to be utilized. To minimize setup time of the robot, we added a new service that runs on boot to establish the connection with the sensor. Please see Appendix A for complete details.

2.5 Stereo Tracking Camera

We utilize a visual-inertial odometry sensor to aid in localization and local path planning. Specifically, the Intel Realsense T265 sensor was selected.

This sensor consists of stereo fisheye cameras and an inertial measurement unit (IMU), and it fits the low SWaP-C requirement for our project. Table 4 lists the specifications for this sensor.

Table 4. Specifications for the Intel Realsense T265 odometry tracking camera.

Description	Value
Dimensions	10.8 × 2.5 × 1.3 cm
Weight	0.06 kg
Operating FOV	163°

2.6 Forward- and Backward-Facing Red-Green-Blue and Depth (RGB-D) Cameras

Two RGB-D cameras were selected for visual-based odometry and for short-range (0–2 m) obstacle avoidance and navigation. We used the Intel Realsense D435 camera. To maximize coverage and better aid in remote operation of the robot, we mounted the cameras to cover the front and back of the robot. Table 5 shows the specifications for this camera.

Table 5. Intel Realsense D435 red-green-blue and depth (RGB-D) camera specifications.

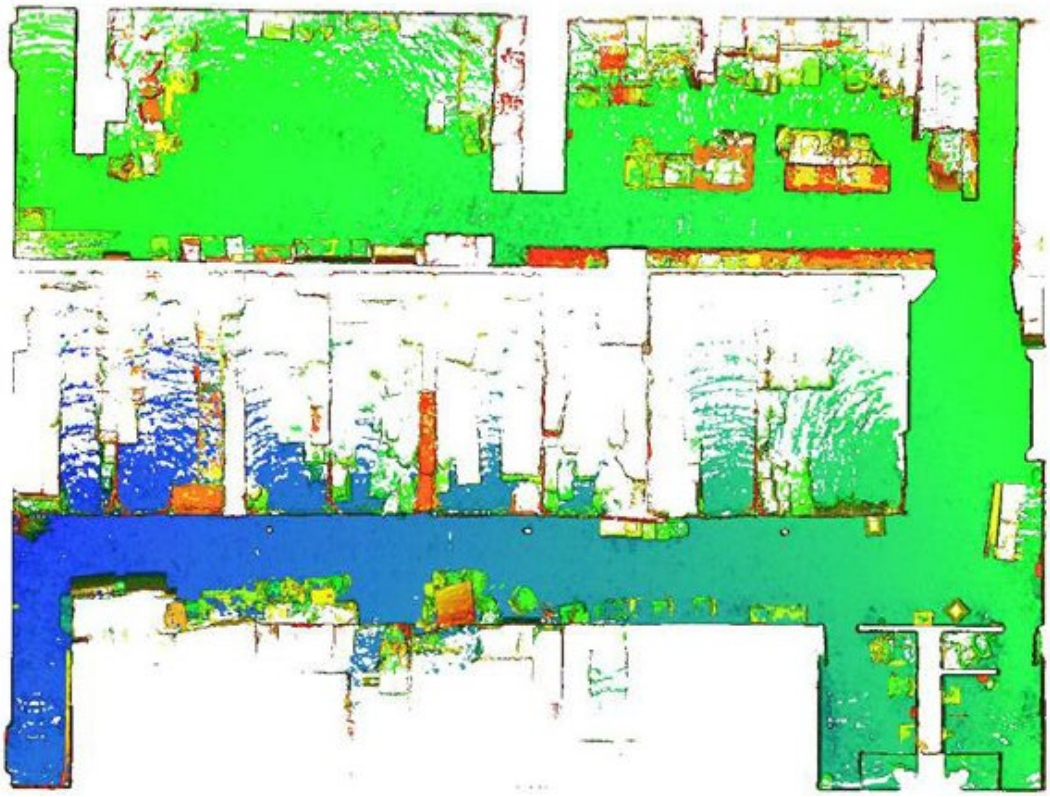
Description	Value
Dimensions	9 × 2.5 × 2.5 cm
Weight	0.02 kg
Operating FOV	69° × 42°
Max range	2 m
Depth error	<2% at 2 m

3 Accurate Odometry and Mapping

3.1 Overview

In addition to providing 2D occupancy grids that present a simple abstraction of the environment for online path planning and navigation, stored data can be post-processed in a few minutes to generate accurate 3D point clouds. These dense point clouds, an example of which is shown in Figure 3, improve situational awareness and can guide decision-making.

Figure 3. Collected point cloud.

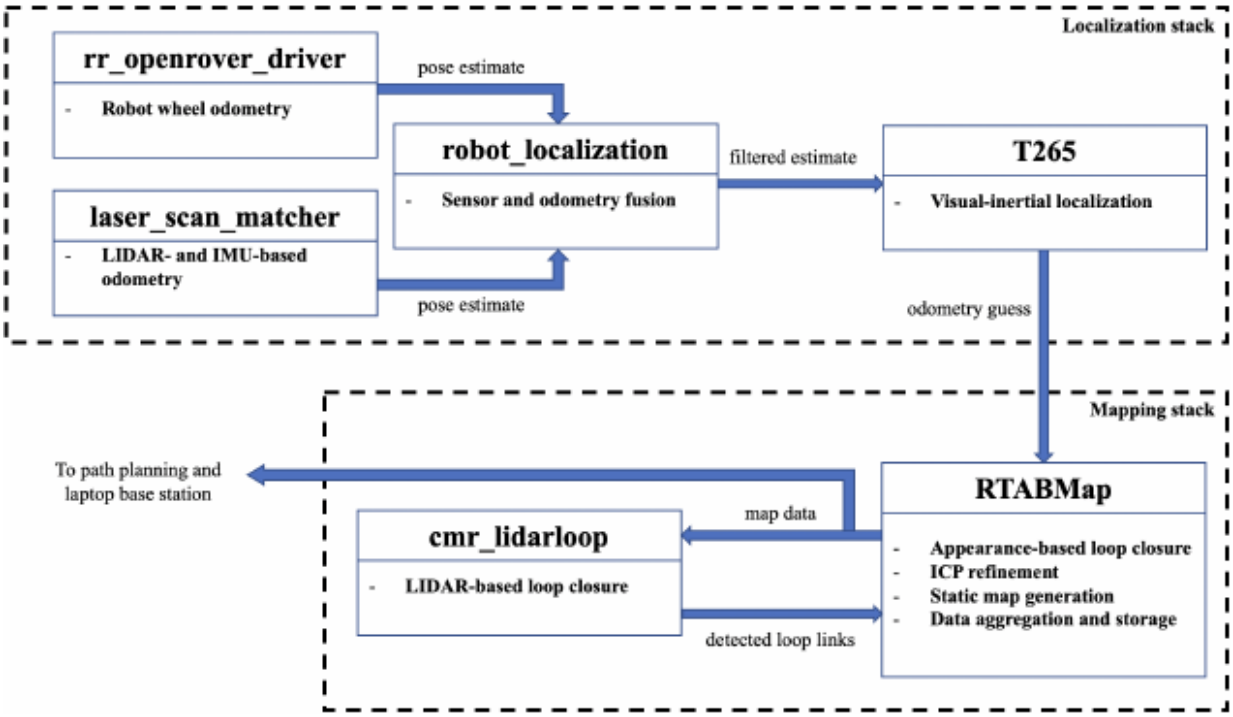


The SLAM approach relies on two major components and is shown in Figure 4. First, for localization and accurate odometry, an initial guess frame is obtained from the T265 tracking camera. To minimize drift and increase robustness of localization, we leveraged the `laser_scan_matcher` package to generate a pose estimate that was fed into our T265 sensor to produce the initial guess. This package is an incremental laser scan registration tool that implements a point-to-line iterative closest point (ICP) algorithm to match between consecutive laser scan messages and publish the estimated position of the laser (Censi 2008). A voxel-filtered version of the lidar

cloud is fed to the `laser_scan_matcher` node along with the IMU data from the T265 camera and the wheel odometry message from the robot to generate a pose estimate.

To convert this message into one that the T265 can use, we leveraged the `robot_localization` package, which took in the input of the `laser_scan_matcher` node and outputted an odometry message that is then forwarded to the T265 to use and minimize drift.

Figure 4. Mapping system.



Second, the final guess frame generated by the T265 camera is fed into the Real-Time Appearance-Based Mapping library (RTAB-Map), which is an RGB-D, stereo, and lidar graph–based SLAM approach based on an incremental appearance-based loop closure detector (Labbé et al. 2019). As the robot moves in the environment and collects new images, a loop closure detector determines how likely a new image comes from a previous location or a new location. When a loop closure hypothesis is accepted, a new constraint is added to the map graph, and then a graph optimizer minimizes the errors in the map. A memory management approach is used to limit the number of locations used for loop closure detection and graph optimization so that real-time constraints on large-scale environments are always respected.

Because RTAB-Map uses primarily visual data, places and loop closures can rarely be recognized under dynamic lighting conditions. On the other hand, laser scans are more resilient to such conditions which motivated us to make more extensive use of this data. For that reason, we complement the appearance-based loop detection of RTAB-Map with a lidar-based loop detector by incorporating the `cmr_lidarloop` package. This allows us to add constraints to our graph-based SLAM based on features obtained from 3D scans (Habich et al. 2021). Appendix B details the launch files used for our localization and mapping approach.

3.2 Mapping Accuracy

To quantify the accuracy of our mapping platform, we compared the point clouds collected at the AGC bunker and a building during MSSPIX 2022 to point clouds from a handheld survey lidar system. The two point clouds where then aligned and the root-mean-square error (RMSE) for each point was calculated using the nearest neighbor distance.[5]

For the AGC bunker, we compared the ground-truth point cloud collected using the Leica C10 lidar to the one collected using the robotic platform, and calculated an average RMSE of 4.7 cm. Figure 5 shows the comparison between the two point clouds and confirms the high accuracy of the map generated with our SLAM approach. We also compared some of the point cloud collected during MSSPIX. Specifically, the point cloud of an interior of a building was compared to the one collected from a ZEB Revo lidar, and calculated an average RMSE of 3.8 cm. Figure 6 shows the close overlap between the two point clouds.

While indeed there are small deviations in the point clouds collected with our robot (e.g., the bottom hallway in Figure 3 having a different height compared to the top hallway in the AGC bunker), the difference is rather small compared to the ground-truth point cloud. It is worth noting that the sensor parameters (e.g., camera image resolutions, lidar points per scan, publish rates) can be throttled down to reduce the computational load and reduce power consumption if lower accuracy can be tolerated.

[5] Nearest neighbor distance: for each point of the compared cloud, find the nearest point in the reference cloud and compute their (euclidean) distance.

Figure 5. Point cloud comparison between Leica C10 lidar and the robotic platform.

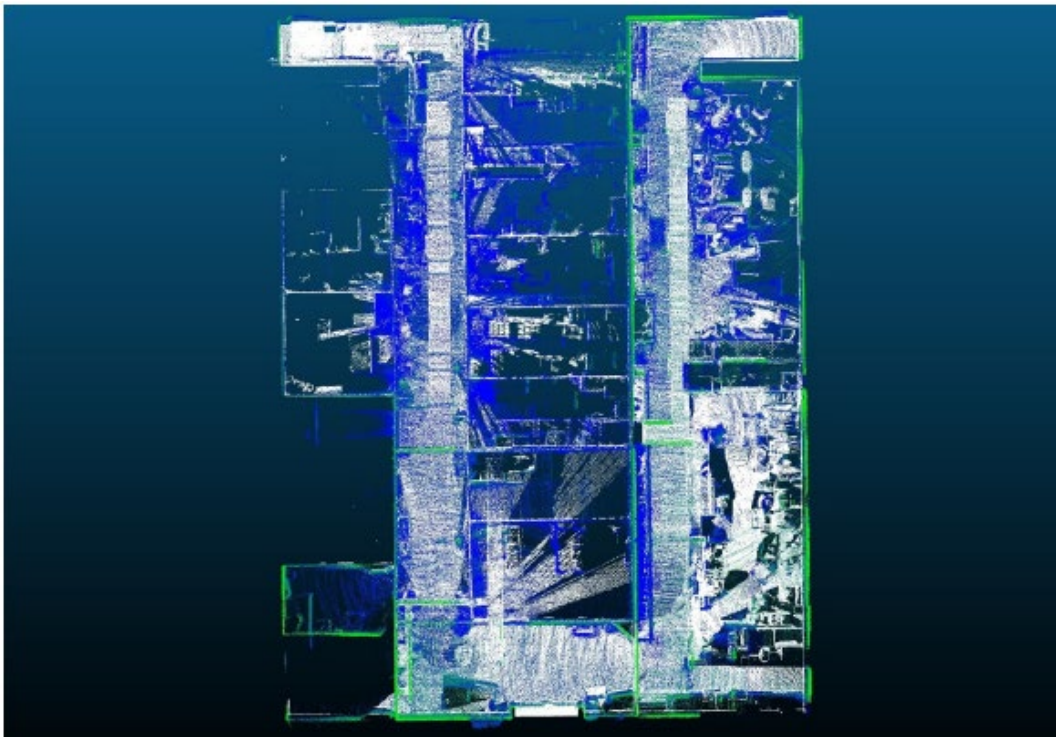
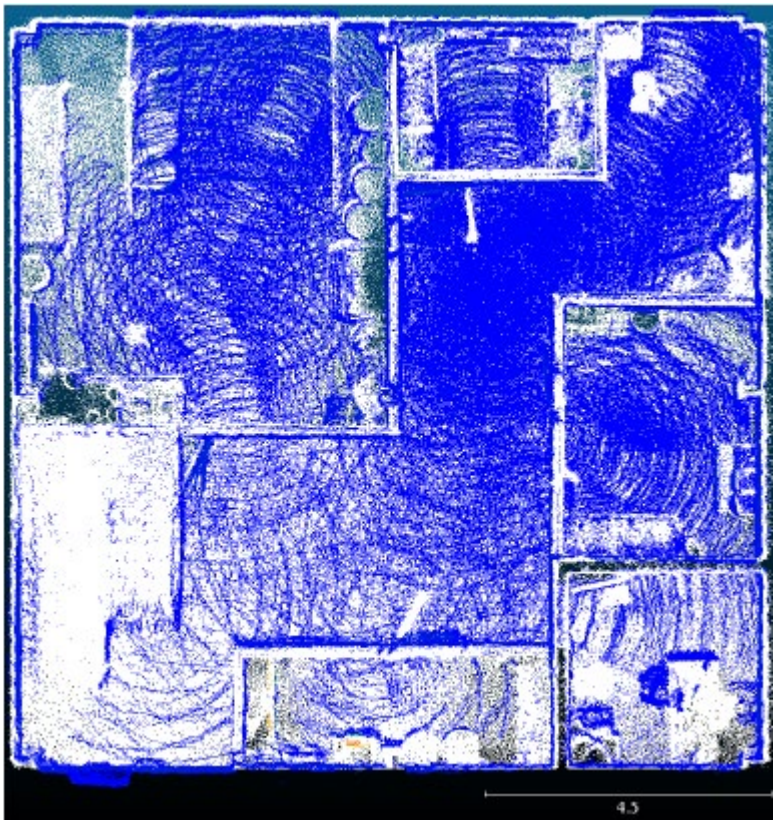


Figure 6. Point cloud of a building interior compared with ZEB Revo lidar.



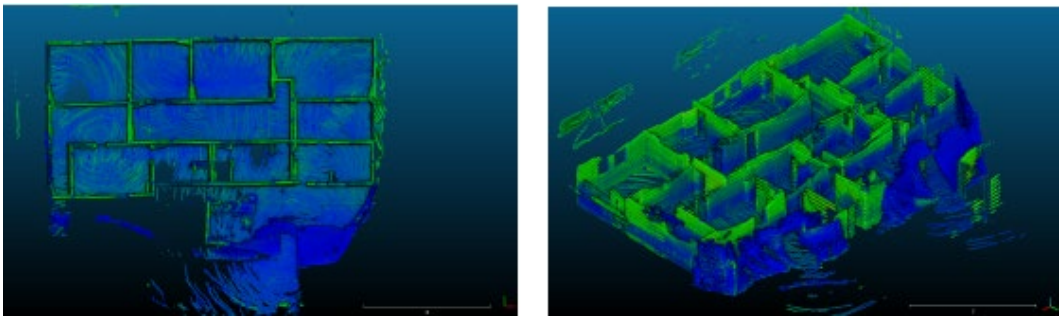
As RTAB-Map continuously collects visual information while constructing the map, it is also possible to overlay the RGB information onto the collected point cloud. Figure 7 shows the result of applying this information onto a mesh generated from the point cloud, which further aids in understanding the interior environment. However, this is very sensitive to misalignment between sensors, and fine alignment is necessary to accurately project the RGB values onto the point cloud.

Figure 7. Color information.

3.3 Additional Mapping Results

Our mapping platform was tested in an experimental shoot house in a hangar at Fort Belvoir to see the robot's performance in tighter spaces. The test area had very minimal visual cues (all similar white walls) with many turns needed to travel through the house, which generally amplifies the drift issues in odometry. Figure 8 shows the top and front views of the collected point cloud, respectively, which confirmed that our mapping solution was capable of addressing these types of environment.

Figure 8. Shoot house collect.



The robotic mapping platform was also used to map the interior of a conduit in the Mojave River Dam. This environment was particularly challenging due to the lack of visual features and the similarity of the collected 3D scans throughout the conduit, as well as the lack of adequate lighting inside the conduit. Figure 9 shows the top view of the collected point cloud, which confirms the minimal drift of odometry during the generation of the map. Initially, point-to-plane ICP refinement was performed to compensate for the robot drift, but the challenging environment still caused significant drift and for the final result to be warped. However, after switching to a point-to-point ICP refinement, the final result showed very minimal drift.

Finally, the robotic platform was demonstrated during MSSPIX 2022 at Fort Leonard Wood and was used by soldiers to map an interior of a building and a small underground tunnel. These results are shown in Figure 10 and Figure 11, respectively, and they demonstrated that our robotic platform is indeed capable of providing accurate mapping in near real time to improve situational awareness.

Figure 9. Mojave River Dam collect.

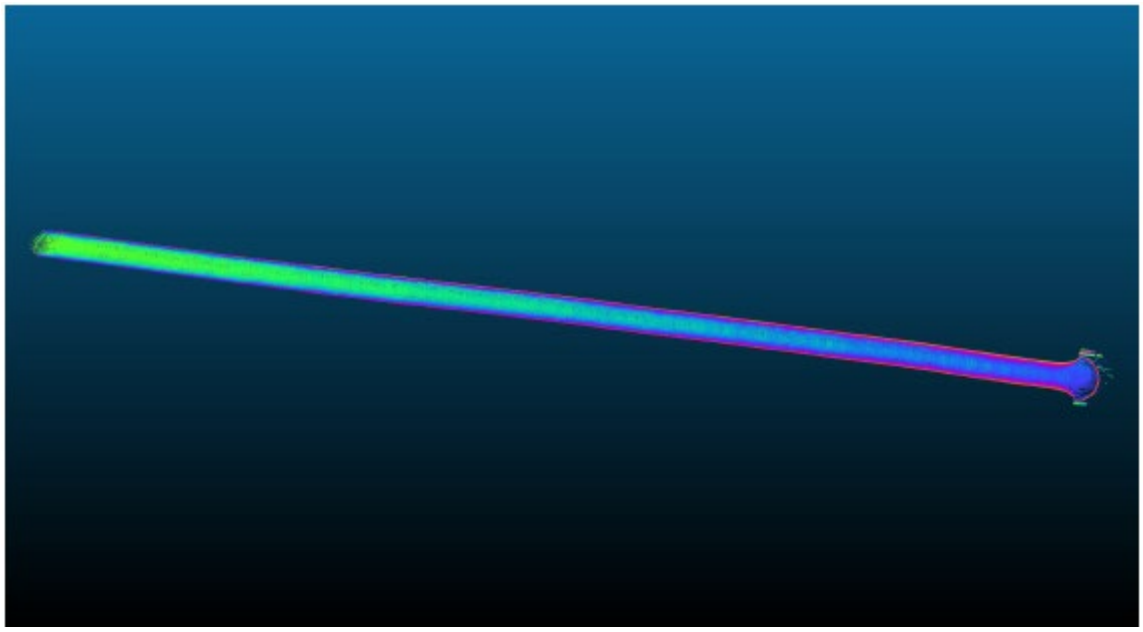


Figure 10. Maneuver Support, Sustainment and Protection Integration eXperiments (MSSPIX) building interior collect.

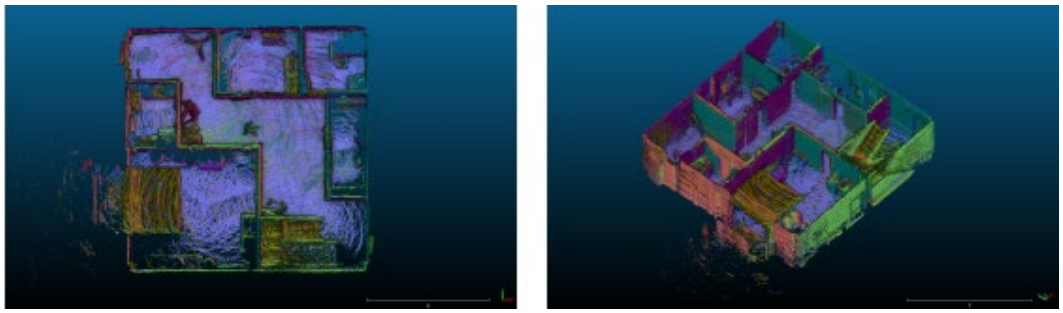
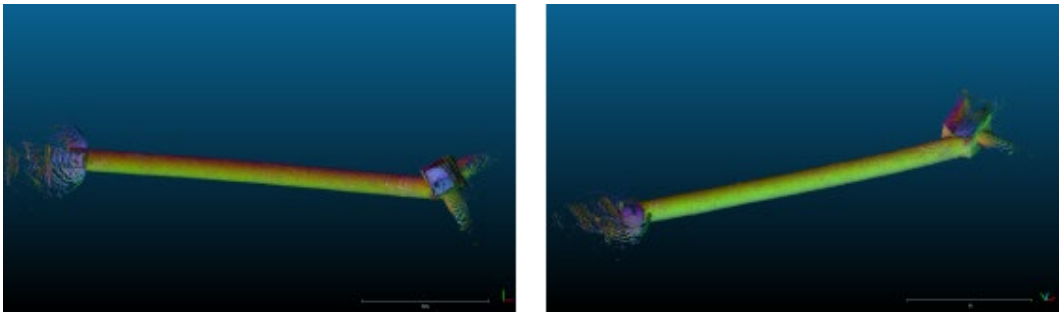


Figure 11. MSSPIX tunnel collect.



4 Navigation and Path Planning

4.1 Overview

The final version of the robot has three modes of operation: manual driving using a wireless controller connected to the laptop, waypoint placement where the robot autonomously navigates to a desired goal placed by the operator, and frontier-based exploration where the robot continues to navigate to available frontiers until all frontiers are visited.

To ensure that the robot does not run into walls during manual driving, we utilized the `erdc_safe_teleop` collision velocity filter, which slows the robot down or completely stops it as it approaches an obstacle. Regarding autonomous navigation and path planning, both for waypoint navigation and frontier-based exploration, the team relied on the `move_base` package, which allows the robot to navigate to its intended goal. Two costmaps are maintained for planning and obstacle avoidance: a global costmap used to calculate a long-term path plan based on the general map and a local costmap used to navigate around dynamic obstacles not available on the global map. To generate these costmaps, ROS leverages a layered approach. For waypoint navigation, we make use of three layers. The first layer is a static layer (`map_layer`) and is produced by the SLAM approach. The second layer (`obstacle_layer`) comprises obstacles detected by the sensors and is overlaid onto the static layer. The third layer (`inflation_layer`) generates a buffer around the obstacles as a safety measure to keep the robot from running into them. Finally, the combination of all layers results in a costmap. As we describe in the next section, additional layers can be included to further customize the path planning strategies.

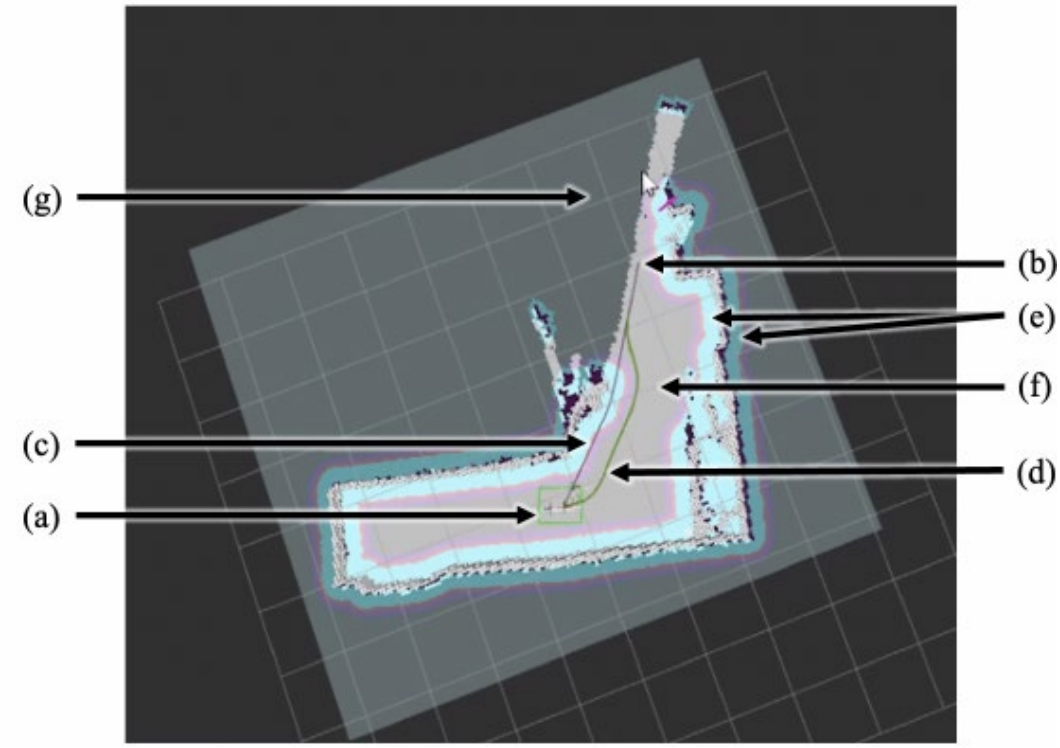
Global and local path planners utilize their respective costmaps to generate feasible trajectories for reaching their target while avoiding obstacles. For this project, we used `global_planner` and `teb_local_planner` for global and local path planning, respectively. The global planner is a versatile package as it allows for using A* or Dijkstra's algorithms for global path planning. For fast-moving vehicles, A* may be preferable. However, Dijkstra was chosen for this project, as it maximized coverage when determining the optimal path. The `global_planner` package also implements an orientation filter. Hypothetically, if the robot was equipped with a single camera, it would be possible to drive down a hallway and then drive back-

ward, using the orientation filter, on the return trip, to ensure loop closures occurred in order to minimize drift. However, a dual camera approach was used; as a result, the orientation filter was not modified. Finally, the team enabled the `cost_factor` (3.0), `neutral_cost` (50), and `lethal_cost` (253).

For local path planning, the `teb_local_planner` was selected ahead of the `eband_local_planner` reported in (Christie et al. 2021) since development for the latter was stopped after ROS Kinetic. This package implements the Timed Elastic Band (TEB) approach (Rösmann et al. 2017) and presents an online optimal local trajectory planner for navigation and control of our robot. The initial trajectory generated by a global planner is optimized during runtime with respect to minimizing the trajectory execution time (time-optimal objective), separation from obstacles, and compliance with kinodynamic constraints such as satisfying maximum velocities and accelerations.

Figure 12 shows a snapshot of the path planning procedure the robot performs to navigate to its desired destination and along the way.

Figure 12. Path Planning.



Appendix C details the launch files and parameters used for our path planning and navigation stack.

Waypoint navigation was tested in the AGC bunker and during MSSPIX 2022, where it was found to be much more useful than manual teleoperation as the user did not need to worry about obstacles while relying on a limited field of view from the cameras. This is especially important because no noticeable differences were observed between the point clouds generated using this method and teleoperation presented in Chapter 3. Figure 13 shows exploration of the AGC bunker using waypoint navigation, while Figure 14 shows the final grid map of exploring a building during MSSPIX using waypoint navigation.

Figure 13. Waypoint exploration.

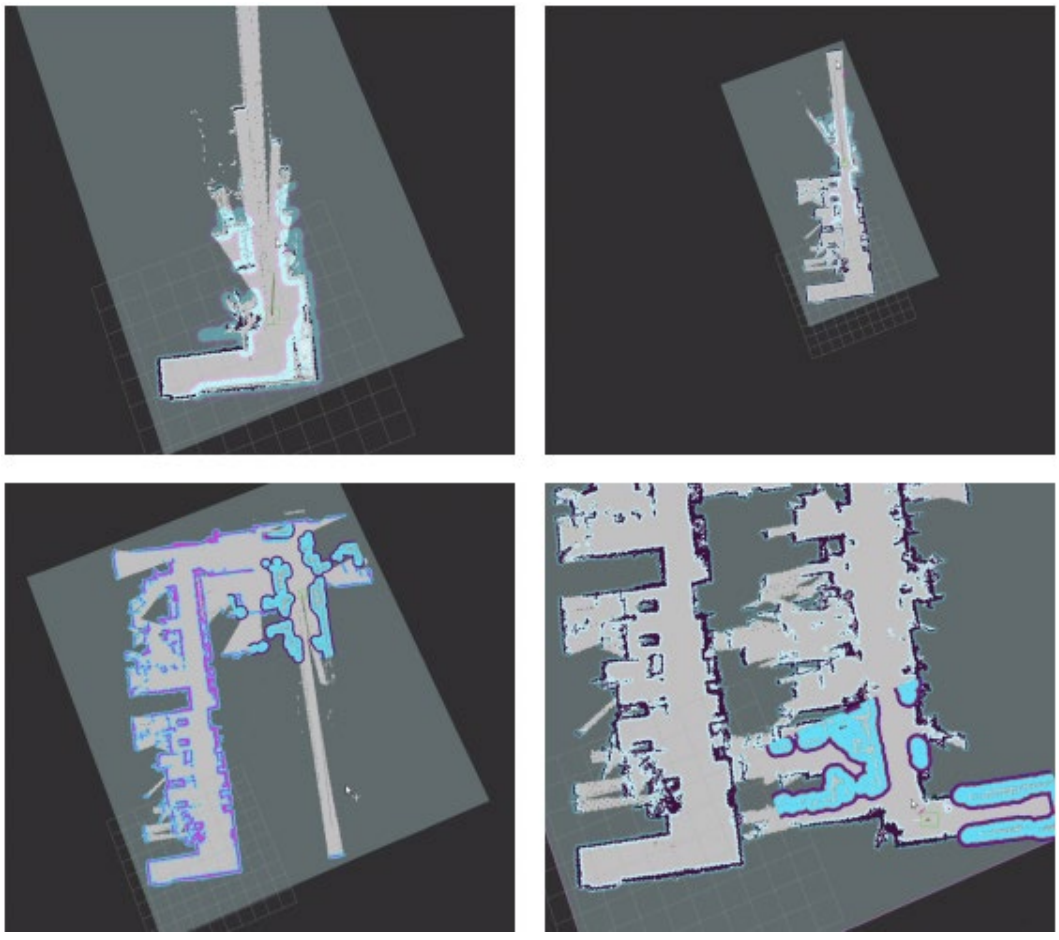
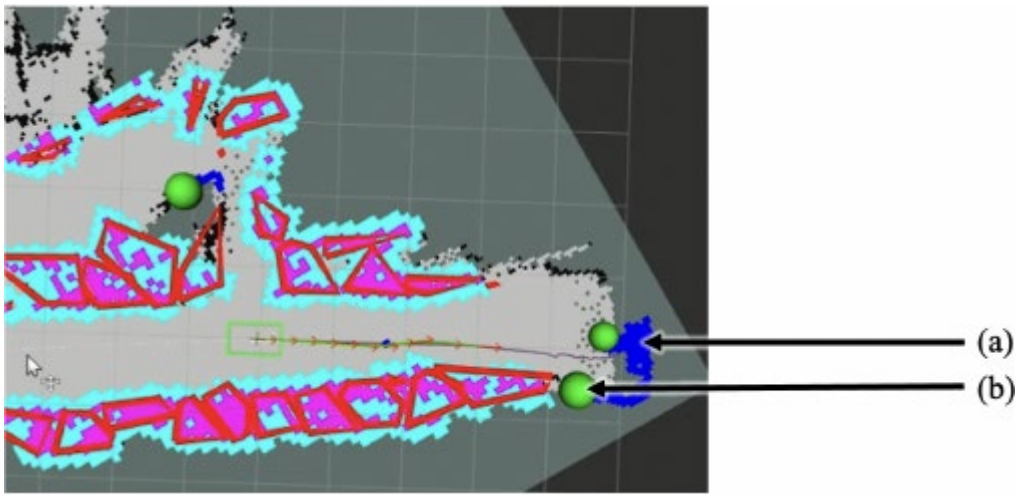


Figure 14. Waypoint exploration interior.

4.2 Autonomous Exploration

For frontier-based exploration, the boundary between *known* (free or occupied) and *unknown* points (referred to as a frontier) becomes a possible navigation goal. Associated with each frontier is a cost for clearing the frontier that includes factors such as distance, orientation, and size. In this report, we leverage the `explore_lite` package to send goals to the path planner, where the exploration node selects and publishes the frontier with lowest cost as a navigation goal. The path-planning node subscribes to the navigation goals and attempts to steer the robot toward goals while keeping the cost at a minimum. This process repeats for all existing frontiers and with the addition of new frontiers. Figure 15 shows an example of frontier-based exploration.

Figure 15. Autonomous exploration.



However, since `explore_lite` does not have the same virtual fencing capabilities that exist in other frontier-based exploration applications such as the `frontier_exploration` package, we leveraged the modified version reported in Christie et al. 2021 to ensure that our `explore_lite` node does not endlessly send new navigation goals to our path planner in an unbounded environment. Furthermore, a fourth layer is added to the costmaps that includes *virtual obstacles*, allowing us to create virtual barriers to limit the robot from exploring certain spaces. These virtual obstacles can be published or cleared by using the `rostopic_pub` command in the terminal. An example of a bash script that generates a virtual line obstacle is given in Listing 1.

Listing 1. An example bash script to generate a virtual line obstacle.

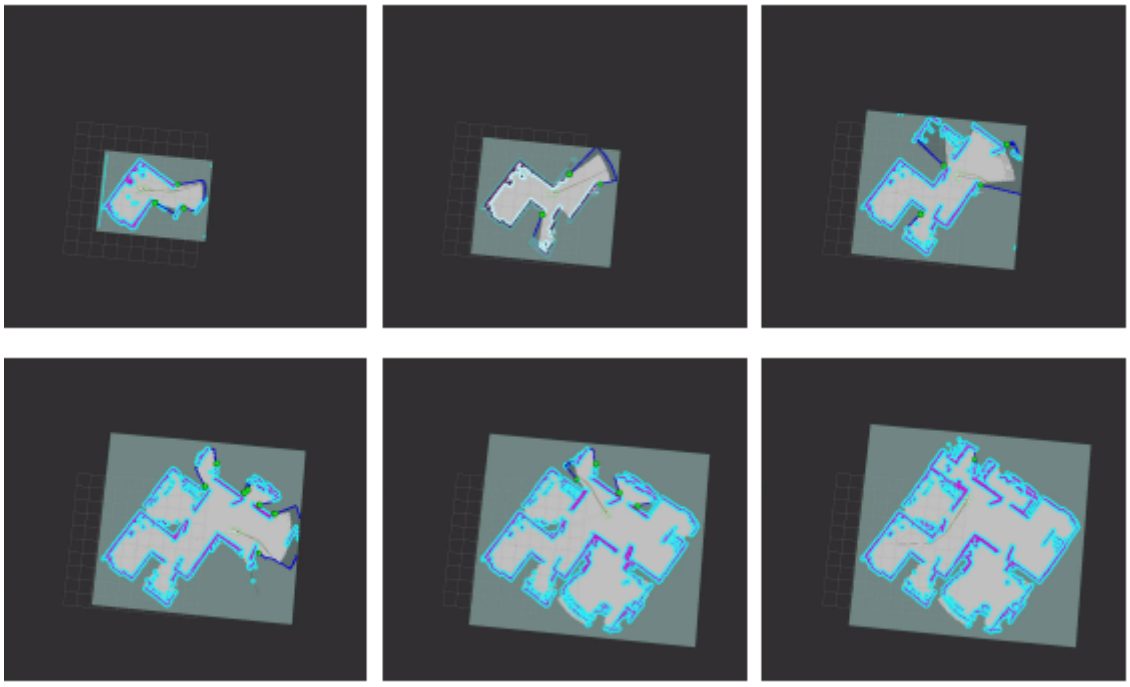
```
1 #!/bin/bash
2
3 # bash script for auto generating a virtual line
obstacle
4 #
5 # example usage:
6 # $ bash virtualLine.sh -w 0.5 -d 0.25
7
8 width=''
9 distance=''
10
11 while getopts 'w:d:' flag; do
```

```
12 case "${flag}" in
13 w) width=${OPTARG};;
14 d) distance=${OPTARG};;
15 esac
16 done
17
18 # horizontal line
19 xHrz=-$distance;
20 yHrz=$width;
21 hrzForm="[{x: $xHrz, y: -$yHrz}, {x: $xHrz, y: $yHrz}]"
22
23 echo "hrzForm = $hrzForm"
24
25 rostopic pub /virtual_costamp_layer/obsctacles
     custom_msgs/Obstacles "list: [form: $hrzForm]"
```

The modified exploration package from Christie et al. (2021) also ensures that the robot returns to its starting position if no frontiers are remaining and the exploration is complete. For situations where we require the connection between the robot and the laptop to be maintained, we incorporated the `lost_comms_recovery` package into our autonomous navigation and exploration code. This package works by periodically pinging the laptop over Wi-Fi and sets a goal to return to the starting position if the pinging continuously fails for a set number of times. The time period between pings depends on the computational load and the available resources. In this work, we set the timeout ping count to 20, which on average translated to approximately 1 minute of no connection. Appendix D details the launch files used for exploration and lost connection recovery.

The autonomous exploration strategy was tested in the AGC bunker and during the MSSPIX 2022 demo. Once again, no significant difference in the overall mapping performance was observed when using autonomous exploration compared to using the other two modes of operation. Figure 16 shows snapshots of the autonomous exploration strategy that operates in the building environment in the MSSPIX demo, which took an average of 10 to 12 minutes to complete.

Figure 16. Interior autonomous exploration.



Autonomous exploration was also tested in the small tunnel at the MSSPIX facility, and Figure 17 shows snapshots of those results. The exploration strategy was terminated before exploring the entire tunnel system as the tunnel terrain did not meet our 2D environment assumption for the robot due to ground slopes and rough terrain. However, the robot was still able to explore the near-flat area at the tunnel entrance, and the collected 3D point cloud is shown in Figure 18, which shows the main entrance along with smaller side tunnels and access ports at the top.

Figure 17. Tunnel autonomous exploration.

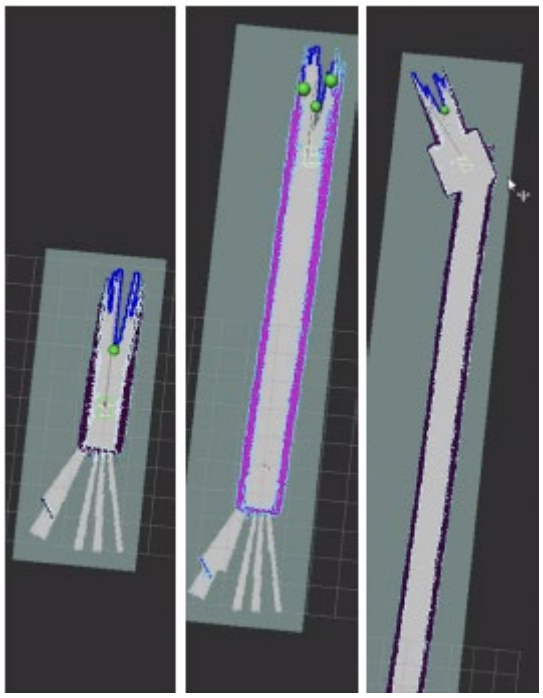
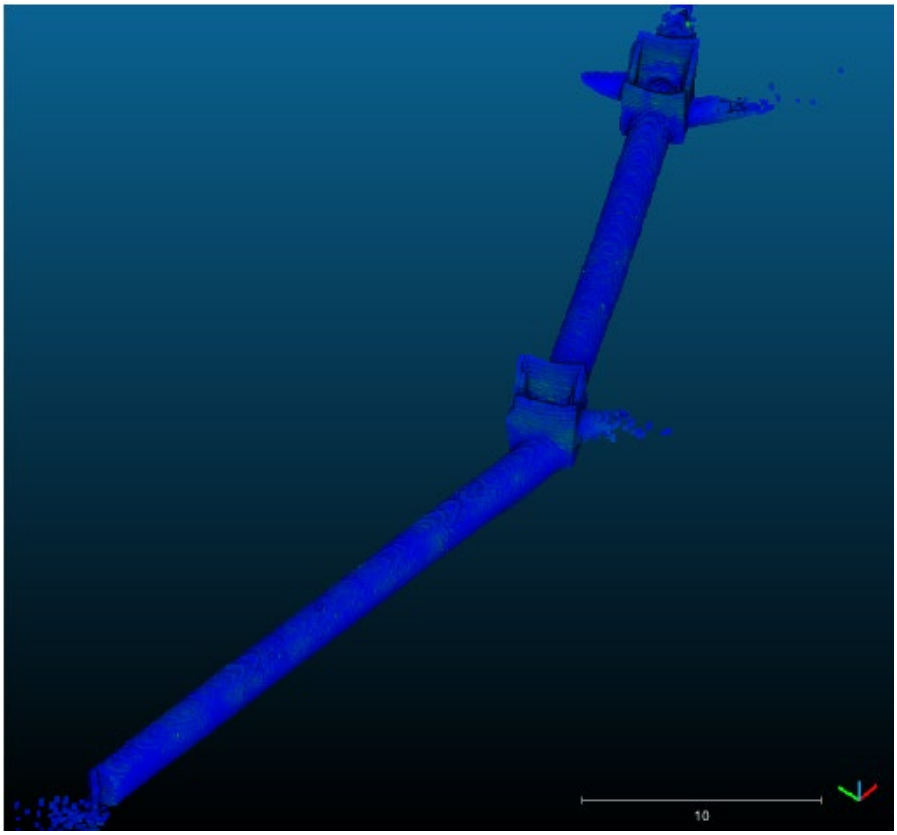


Figure 18. Tunnel exploration point cloud.



5 Conclusion and Future Directions

In this report we present a low SWaP-C odometry and mapping solution for GPS-denied environments. This solution includes autonomous navigation and mapping capabilities in 2D environments. Our mapping robot was tested in multiple indoor facilities, and the generated point clouds accurately matched those obtained using a handheld survey lidar system. A 2D path planning and autonomous exploration approach is also presented, which uses ROS packages to effectively explore unknown environments.

Future efforts will consider expanding on this work in multiple directions to optimize exploration of various environments. Specifically, future work will incorporate ground segmentation for improved navigation, as well as incorporating elevation changes into path planning strategies. Finally, more advanced exploration strategies will be investigated as a substitute for frontier-based exploration.

Bibliography

Castellanos, J. A., J.Neira, and J. D. Tardós. 2001. "Multisensor Fusion for Simultaneous Localization and Map Building." *IEEE Transactions on Robotics and Automation* 17 (6): 908–14.

Censi, A. 2008. "An ICP Variant Using a Point-to-Line Metric." In *2008 IEEE International Conference on Robotics and Automation*: 19–25.

Christie, B. A, O. Ennasr, and G. P. Glaspell. 2021. "Autonomous Navigation and Mapping in a Simulated Environment." Technical Report. ERDC/GRL TR-21-5. Alexandria, VA: US Army Corps of Engineers.

Department of the Army. 2021. "Army Multi-Domain Intelligence FY21-22 S&T Focus Areas." https://apps.dtic.mil/sti/pdfs/AD1114489.pdf.

Glaspell, G., S. Lessard, B. Christie, K. Jannak-Huang, N. Wilde, W. He, O. Ennasr, et al. 2020. Optimized Low Size, Weight, Power and Cost (SWaPC) Payload for Mapping Interiors and Subterranean on an Unmanned Ground Vehicle. Technical Report. ERDC/GRL TR-20-6. Alexandria, VA: US Army Corps of Engineers.

Habich, T.-L., M. Stuede, M. Labbé, and S. Spindeldreier. 2021. "Have I Been Here before? Learning to Close the Loop with LiDAR Data in Graph-Based SLAM." In *2021 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*: 504–10.

Jacobson, A., F. Zeng, D. Smith, N. Boswell, T. Peynot, and M. Milford. 2018. "Semi-Supervised Slam: Leveraging Low-Cost Sensors on Underground Autonomous Vehicles for Position Tracking." In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*: 3970–77.

Keidar, M., and G. A. Kaminka. 2014. "Efficient Frontier Detection for Robot Exploration." *International Journal of Robotics Research* 33 (2): 215–36.

Labbé, M., and F. Michaud. 2019. "RTAB-Map as an Open-Source Lidar and Visual Simultaneous Localization and Mapping Library for Largescale and Long-Term Online Operation." *Journal of Field Robotics* 36 (2): 416–46.

Rösmann, C., F. Hoffmann, and T. Bertram. 2017. "Integrated Online Trajectory Planning and Optimization in Distinctive Topologies." *Robotics and Autonomous Systems* 88: 142–53.

Wang, D., H. Wang, and L. Liu. 2016. "Unknown Environment Exploration of Multi-Robot System with the FORDPSO." *Swarm and Evolutionary Computation* 26: 157–74.

Wang, Y., A. Liang, and H. Guan. 2011. "Frontier-Based Multirobot Map Exploration Using Particle Swarm Optimization." In *2011 IEEE symposium on Swarm Intelligence*: 1–6.

Zakiev, A., R. Lavrenov, E. Magid, M. Svinin, and F. Matsuno. 2019. "Partially Unknown Environment Exploration Algorithm for a Mobile Robot." *Journal of Advanced Research in Dynamical and Control Systems* 11 (08).

Appendix A: Ouster Connectivity

In this appendix, we discuss how we set up autoconnectivity with the ouster senor. First, we created a new wired connection using our eno1 device by running the command shown in Listing A-1 in the terminal.

Listing A-1. Command for creating a new wired connection.

```
1 sudo ip addr add 10.5.5.1/24 dev eno1
```

After disabling the default wired connection through `nm-connection-editor`, we then added a new `systemd` service by adding the file shown in Listing A-2 to the `/etc/systemd/system/` directory.

Listing A-2. The ouster-dnsmasq.service file created for auto configuring the network for the Ouster sensor.

```
1 [Unit]
2 After=NetworkManager.service
3 [Service]
4 Type=forking
5 ExecStart=/usr/sbin/dnsmasq -C /dev/null -F
    10.5.5.50,10.5.5.100,infinite -i eno1 --bind-dynamic
6 [Install]
7 WantedBy=multi-user.target
```

Finally, the service was enabled by running the commands shown in Listing A-3.

Listing A-3. Commands for enabling newly connected service.

```
1 sudo chmod 644 /etc/systemd/system/ouster-dnsmasq.
    service
2 systemctl daemon-reload
3 systemctl enable ouster-dnsmasq.service
```

Appendix B: SLAM Launch Files

In this appendix, we list the launch and configuration files used for our SLAM approach. First, for localization, we utilize the following (Listing B-1 and Listing B-2).

Listing B-1. Launch file for robot localization and laser scan matcher.

```
1 <launch>
2 <arg name="repo_path" default="/home/nuc/MOWLES/IRL/
  preTest/rover_demo" />
3
4 <node pkg="laser_scan_matcher" type="
  laser_scan_matcher_node" name="
  laser_scan_matcher_node" output="screen">
5 <param name="fixed_frame" value="cam_t265_1_odom_frame"/>
6 <param name="max_iterations" value="30"/>
7 <param name="use_imu" value="true"/>
8 <param name="use_odom" value="false"/>
9 <param name="use_cloud_input" value="true"/>
10 <param name="publish_tf" value="false"/>
11 <param name="publish_pose" value="true"/>
12 <param name="publish_pose_stamped" value="false"/>
13 <param name="kf_dist_linear" value="0.10"/>
14 <param name="kf_dist_angular" value="0.175"/>
15 <param name="cloud_range_min" value="0.4"/>
16 <param name="cloud_range_max" value="100.0"/>
17 <remap from="cloud" to="/lidar/voxel_grid/output"/>
18 <remap from="imu" to="/cam_t265_1/imu"/>
19 <remap from="odom" to="/rr_openrover_driver/odom_encoder"/>
20 <param name="do_compute_covariance" value="1"/>
21 <param name="publish_pose_with_covariance" value="true"/>
22 <param name="publish_pose_with_covariance_stamped"
   value="true"/>
23 </node>
24
25 <node pkg="robot_localization" type="
  ekf_localization_node" name="ekf_se" clear_params="true">
26 <rosparam command="load" file="$(arg repo_path)/
  robot_localization/ekf_flipperbot.yaml" />
27 </node>
28 </launch>
```

Listing B-2. YAML parameters for robot localization.

```
1
2 frequency: 30
3
4 sensor_timeout: 0.1
5
6 two_d_mode: true
7
8 transform_time_offset: 0.0
9
10 transform_timeout: 0.0
11
12 print_diagnostics: false
13
14 debug: false
15
16 debug_out_file: ~/robot_localization_debug.txt
17
18 publish_tf: false
19
20 publish_acceleration: false
21
22 map_frame: map
23 odom_frame: cam_t265_1_odom_frame
24 base_link_frame: base_link
25 world_frame: cam_t265_1_odom_frame
26
27 pose0: /pose_with_covariance_stamped
28 pose0_config: [true, true, false,
29 false, false, true,
30 true, true, false,
31 false, false, true,
32 false, false, false]
33 pose0_differential: false
34 pose0_relative: false
35 pose0_queue_size: 10
36 pose0_rejection_threshold: 2 # Note the difference in parame-
   ter name
37 pose0_nodelay: false
38
39 use_control: true
40 stamped_control: false
41 control_timeout: 0.2
42 control_config: [true, false, false, false, false, true]
43 acceleration_limits: [1.3, 0.0, 0.0, 0.0, 0.0, 3.4]
```

```
44 deceleration_limits: [1.3, 0.0, 0.0, 0.0, 0.0, 4.5]
45 acceleration_gains: [0.8, 0.0, 0.0, 0.0, 0.0, 0.9]
46 deceleration_gains: [1.0, 0.0, 0.0, 0.0, 0.0, 1.0]
47
48
49 process_noise_covariance:
50 [0.05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
51 0, 0.05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
52 0, 0, 0.06, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
53 0, 0, 0, 0.03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
54 0, 0, 0, 0, 0.03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
55 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
56 0, 0, 0, 0, 0, 0, 0.025, 0, 0, 0, 0, 0, 0, 0, 0,
57 0, 0, 0, 0, 0, 0, 0, 0.025, 0, 0, 0, 0, 0, 0, 0,
58 0, 0, 0, 0, 0, 0, 0, 0, 0.04, 0, 0, 0, 0, 0, 0,
59 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.01, 0, 0, 0, 0, 0,
60 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.01, 0, 0, 0, 0,
61 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0 0 0,
62 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.01 0, 0,
63 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.01, 0,
64 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.015]
65
66 initial_estimate_covariance:
67 [1e-9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
68 0, 1e-9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
69 0, 0, 1e-9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
70 0, 0, 0, 1e-9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
71 0, 0, 0, 0, 1e-9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
72 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
73 0, 0, 0, 0, 0, 0, 1e-9, 0, 0, 0, 0, 0, 0, 0, 0,
74 0, 0, 0, 0, 0, 0, 0, 1e-9, 0, 0, 0, 0, 0, 0, 0,
75 0, 0, 0, 0, 0, 0, 0, 0, 1e-9, 0, 0, 0, 0, 0, 0,
76 0, 0, 0, 0, 0, 0, 0, 0, 0, 1e-9, 0, 0, 0, 0, 0,
77 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1e-9, 0, 0, 0, 0,
78 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
79 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1e-9, 0, 0,
80 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1e-9, 0,
81 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1e-9]
```

We also generate new RGB-D messages from the lidar and the cameras in order to maximize the range of the depth images for our SLAM approach as shown in Listing B-3.

Listing B-3. Launch file generating depth images and synchronizing with the camera feed.

```
1 <launch>
2 <arg name="node_start_delay" default="0.0" />
3 <arg name="queue_size" default="150"/>
4
5 <group ns="rtabmap">
6 <node pkg="nodelet" type="nodelet" name="
 pc2di_manager" args="manager" output="screen" launch
 -prefix="bash -c 'sleep $(arg node_start_delay); $0
 $@' "/>
7 <node pkg="nodelet" type="nodelet" name="
 rgbd_sync_manager" args="manager" output="screen"
 launch-prefix="bash -c 'sleep $(arg node_start_delay
 ); $0 $@' "/>
8
9 <!--FISHEYE 1-->
10 <node pkg="nodelet" type="nodelet" name="
 pointcloud_to_depthimage_1" args="load rtabmap_ros/
 pointcloud_to_depthimage pc2di_manager" output="
 screen" launch-prefix="bash -c ' sleep $(arg
 node_start_delay); $0 $@' ">
11 <!-- subscribed topics -->
12 <remap from="camera_info" to="/cam_t265_1/
  fisheye1/camera_info"/>
13 <remap from="cloud" to="/os_cloud_node/points"/>
14 <!-- published topics -->
15 <remap from="image_raw" to="image_raw_1"/>
  <!-- output in mm -->
16 <remap from="image" to="image_1"/> <!-- output in m -->
17 <!-- parameters -->
18 <param name="fixed_frame_id" type="string" value="os_sensor"/>
19 <param name="fill_holes_size" type="int" value="30"/>
20 <param name="fill_iterations" type="int" value="1"/>
21 <param name="decimation" type="int" value="1"/>
22 <param name="queue_size" type="int" value="$(arg
  queue_size)"/>
23
24 </node>
25 <node pkg="nodelet" type="nodelet" name="
 rgbd_sync_1" args="load rtabmap_ros/rgbd_sync
 rgbd_sync_manager" output="screen" launch-prefix="
 bash -c ' sleep $(arg node_start_delay); $0 $@' ">
26 <!-- subscribed topics -->
27 <remap from="rgb/image" to="/cam_t265_1/fisheye1/image_raw"/>
28 <remap from="depth/image" to="image_raw_1"/>
```

```
29 <remap from="rgb/camera_info" to="/cam_t265_1/
fisheye1/camera_info"/>
30 <!-- published topics -->
31 <remap from="rgbd_image" to="rgbd_image_1"/>
32 <remap from="rgbd_image/compressed" to="
 rgbd_image_1/compressed"/>
33
34 <!-- Should be true for not synchronized camera Topics
35 (e.g., false for kinectv2, zed, realsense, true
  for xtion, kinect360)-->
36 <param name="approx_sync" value="true"/>
  <!-- set to true because we are using the pc2di node-->
37 <param name="queue_size" type="int" value="$(arg
  queue_size)"/>
38 </node>
39
40 <!--FISHEYE 2 -->
41 <node pkg="nodelet" type="nodelet" name="
  pointcloud_to_depthimage_2" args="load rtabmap_ros/
  pointcloud_to_depthimage pc2di_manager" output="
  screen" launch-prefix="bash -c ' sleep $(arg
  node_start_delay); $0 $@' ">
42 <!-- subscribed topics -->
43 <remap from="camera_info" to="/cam_t265_1/
  fisheye2/camera_info"/>
44 <remap from="cloud" to="/os_cloud_node/points"/>
45
46 <!-- published topics -->
47 <remap from="image_raw" to="image_raw_2"/>
  <!-- output in mm -->
48 <remap from="image" to="image_2"/> <!-- output in m -->
49 <!-- parameters -->
50 <param name="fixed_frame_id" type="string" value="os_sensor"/>
51 <param name="fill_holes_size" type="int" value="30"/>
52 <param name="fill_iterations" type="int" value="1"/>
53 <param name="decimation" type="int" value="1"/>
54 <param name="queue_size" type="int" value="$(
  arg queue_size)"/>
55
56 </node>
57 <node pkg="nodelet" type="nodelet" name="
  rgbd_sync_2" args="load rtabmap_ros/rgbd_sync
  rgbd_sync_manager" output="screen" launch-prefix="
  bash -c ' sleep $(arg node_start_delay); $0 $@' ">
58 <!-- subscribed topics -->
```

```
59 <remap from="rgb/image" to="/cam_t265_1/
  fisheye2/image_raw"/>
60 <remap from="depth/image" to="image_raw_2"/>
61 <remap from="rgb/camera_info" to="/cam_t265_1/
  fisheye2/camera_info"/>
62 <!-- published topics -->
63 <remap from="rgbd_image" to="rgbd_image_2"/>
64 <remap from="rgbd_image/compressed" to="
  rgbd_image_2/compressed"/>
65
66 <!-- Should be true for not synchronized camera
  topics
67 (e.g., false for kinectv2, zed, realsense, true
  for xtion, kinect360)-->
68 <param name="approx_sync" value="true"/>
  <!-- set to true because we are using the pc2di node-->
69 <param name="queue_size" type="int" value="$(arg
  queue_size)"/>
70 </node>
71 </group>
72
73
74
75
76 </launch>
```

Finally, we utilized RTABMap for graph-based SLAM with the following configuration in Listing B-4.

Listing B-4. Launch file Simultaneous Localization and Mapping Using Real-Time Appearance-Based Mapping.

```
1 <launch>
2 <arg name="node_start_delay" default="0.0" />
3 <arg name="queue_size" default="300"/> <!-- errors
  appeared when queue_size is small -->
4
5 <group ns="rtabmap">
6 <node name="rtabmap" pkg="rtabmap_ros" type="rtabmap"
  output="screen" args="--delete_db_on_start" launchprefix="
  bash -c 'sleep $(arg node_start_delay); $0 $@' ">
7 <!-- subscribed topics -->
8 <remap from="odom" to="/cam_t265_1/odom/
```

```
 sample"/> <!-- use initial odometry guess from T265
 tracking camera -->
9 <remap from="rgbd_image0" to="rgbd_image_1"/>
 <!-- use if multiple RGBD cameras are used (using rgbd_sync) -->
10 <remap from="rgbd_image1" to="rgbd_image_2"/>
  <!-- use if multiple RGBD cameras are used
  (using rgbd_sync) -->
11 <remap from="scan_descriptor" to="/cmr_lidarloop/
  scan_descriptor" />
12
13 <!-- configuration parameters -->
14 <param name="subscribe_scan" type="bool"
  value="false"/> <!-- NOT subscribing to the 2D laser scan -->
15 <param name="subscribe_scan_cloud" type="bool"
  value="false"/> <!-- NOT subscribing to a 3D point cloud-->
16 <param name="subscribe_scan_descriptor" type="bool"
  value="true"/> <!-- subscribe to the scan
  descriptor provided by cmr_lidarloop (point cloud
  with corresponding lidar features)-->
17 <param name="scan_cloud_max_points" type="int"
  value="0" /> <!-- maximum number of points( 0
  means the max is defined by the size of ranges
  vector for topic): 2048 x 16 x 10?)-->
18 <param name="subscribe_stereo" type="bool"
  value="false" /> <!-- NOT subscribing to a stereo image-->
19 <param name="subscribe_rgbd" type="bool"
  value="true"/> <!-- subscribing to an RGBD
  image (using rgbd_sync)-->
20 <param name="rgbd_cameras" type="int"
  value="2"/> <!-- number of RGBD cameras in use -->
21 <param name="subscribe_rgb" type="bool"
  value="false"/> <!-- NOT subscribing to a
  separate RGB image (NOT using rgbd_sync) -->
22 <param name="subscribe_depth" type="bool"
```

```
  value="false"/> <!-- NOT subscribing to a
  seperate depth image (NOT using rgbd_sync) -->
23 <param name="frame_id" type="string"
  value="base_link"/> <!-- the name of frame
  attached to the mobile base -->
24 <param name="map_frame_id" type="string"
  value="map" /> <!-- the frame attached to the map-->
25 <param name="odom_frame_id" type="string"
  value="" /> <!-- The frame attached to
  odometry. If empty, rtabmap will subscribe to odom
  topic to get odometry. If set, odometry is got from tf -->
26 <param name="queue_size" type="int"
  value="$(arg queue_size)"/> <!-- Size of message
  queue for each synchronized topic. (increased to
  accomodate different message rates) -->
27 <param name="subscribe_odom_info" type="bool"
  value="false"/> <!-- set to true to fix
  problems with rtabmap subscribing to odom_info -->
28
29 <!-- use actionlib to send goals to move_base -->
30 <param name="use_action_for_goal" type="bool" value="true"/>
31 <remap from="move_base" to="/move_base"/>
32
33 <!-- Algorithm parameters"-->
34 <param name="RGBD/AngularUpdate" type="string" value="0.4"/>
35 <param name="RGBD/LinearUpdate" type="string" value="0.4"/>
36 <param name="RGBD/NeighborLinkRefining" type="
  string" value="true"/> <!-- Do odometry correction
  with consecutive laser scans -->
37 <param name="RGBD/ProximityBySpace" type="
  string" value="true"/> <!-- Local loop closure
  detection (using estimated position) with locations in WM -->
38 <param name="RGBD/ProximityByTime" type="
  string" value="false"/> <!-- Local loop closure
```

```
  detection with locations in STM -->
39 <param name="RGBD/ProximityPathMaxNeighbors" type="
  string" value="30"/> <!-- Do also proximity
  detection by space by merging close scans together.-->
40 <param name="RGBD/OptimizeFromGraphEnd" type="
  string" value="false"/> <!-- Optimize graph from
  initial node so /map -> /odom transform will be generated -->
41 <param name="RGBD/OptimizeMaxError" type="
  string" value="3"/> <!-- Reject any loop closure
  causing large errors (>3x link's covariance) in the map -->
42 <param name="RGBD/LocalRadius" type="
  string" value="10"/> <!-- limit length of
  proximity detections -->
43 <param name="Reg/Strategy" type="
  string" value="1"/> <!-- 0=Visual, 1=ICP, 2=Visual+ICP -->
44 <param name="Reg/Force3DoF" type="
  string" value="false"/> <!-- 3D SLAM -->
45 <param name="Grid/FromDepth" type="
  string" value="false"/> <!-- Create 2D occupancy
  grid from laser scan -->
46 <param name="Mem/STMSize" type="
  string" value="30"/> <!-- increased to 30 to
  avoid adding too many loop closures on just seen locations -->
47 <param name="Vis/MinInliers" type="
  string" value="20"/> <!-- 3D visual words
  correspondence distance -->
48 <param name="Kp/DetectorStrategy" type="string" value="8"/>
49 <param name="Vis/FeatureType" type="string" value="8"/>
50 <param name="Optimizer/Strategy" type="string" value="2"/>
51 <param name="Vis/BundleAdjustment" type="string" value="3"/>
52 <param name="Kp/RoiRatios" type="
  string" value="0.0 0.0 0.0 0.4"/>
53 <param name="Grid/MaxObstacleHeight" type="
  string" value="0.8"/>
```

```
54 <param name="Grid/MinGroundHeight" type="
  string" value="-0.10"/>
55 <param name="Grid/MaxGroundHeight" type="
  string" value="0.10"/>
56 <param name="Grid/3D" type="string" value="true"/>
57 <param name="Grid/RayTracing" type="string" value="true"/>
58 <param name="Grid/RangeMax" type="string" value="5"/>
59 <param name="Grid/NormalsSegmentation" type="
  string" value="false"/>
60
61 <!-- ICP parameters -->
62 <param name="Icp/VoxelSize" type="string" value="0.1"/>
63 <param name="Icp/PointToPlaneK" type="string" value="20"/>
64 <param name="Icp/PointToPlaneRadius" type="string" value="0"/>
65 <param name="Icp/PointToPlane" type="string" value="true"/>
66 <param name="Icp/Iterations" type="string" value="20"/>
67 <param name="Icp/Epsilon" type="string" value="0.001"/>
68 <param name="Icp/MaxTranslation" type="string" value="7"/>
69 <param name="Icp/MaxCorrespondenceDistance" type="
  string" value="1"/>
70 <param name="Icp/Strategy" type="
  string" value="1"/> <!-- ICP implementation: 0=Point
  Cloud Library, 1=libpointmatcher, 2=CCCoreLib (
  CloudCompare).-->
71 <param name="Icp/OutlierRatio" type="string" value="0.7"/>
72 <param name="Icp/CorrespondenceRatio" type="
  string" value="0.2"/>
73 <param name="OdomF2M/BundleAdjustment" type="
  string" value="3"/>
74 <param name="Icp/RangeMax" type="string" value="40"/>
75 </node>
76 </group>
77
78 </launch>
```

Appendix C: Path Planning and Navigation

In this appendix, we list the launch and configuration files used for our path planning and navigation stack approach (Listings C-2 through C-9), as well as our collision velocity filter to ensure safe teleoperation of the robot (Listing C-1).

Listing C-1. Launch file for collision velocity filter for teleoperation.

```
1 <launch>
2 <arg name="repo_path" default="/home/nuc/MOWLES/IRL
 /preTest/rover_demo" />
3 <arg name="config_path" default="$(arg repo_path)/
 safe_teleop/config" />
4
5 <group ns="safe_teleop">
6 <node pkg="nodelet" type="nodelet" name="
 pcl_manager" args="manager" output="screen" />
7 <node pkg="nodelet" type="nodelet" name="
 passthrough_x" args="load pcl/PassThrough
 pcl_manager" output="screen">
8 <remap from="~input" to="/os_cloud_node/points" />
9 <rosparam file="$(arg config_path)/lidar_filter_x
 .yaml" command="load"/>
10 </node>
11 <node pkg="nodelet" type="nodelet" name="
  passthrough_z" args="load pcl/PassThrough
  pcl_manager" output="screen">
12 <remap from="~input" to="/safe_teleop/passthrough_x/output"/>
13 <rosparam file="$(arg config_path)/lidar_filter_z
  .yaml" command="load"/>
14 </node>
15
  16 <node pkg="erdc_safe_teleop" type="collision_velocity_fil-
  ter" name="collision_velocity_filter" output="screen">
17 <!-- remap incoming variables -->
18 <remap from="command_in" to="/cmd_vel/joystick"/>
19 <remap from="joy" to="/joystick" />
20 <!-- remap outgoing variables -->
21 <remap from="command" to="/cmd_vel/joystick_filtered"/>
22 <!-- <remap from="relevant_obstacles" to="/base/
  collision_velocity_filter/relevant_obstacles"/> -->
23 <!-- setup anti-colsion costmap -->
24 <rosparam file="$(arg config_path)/safe_teleop_costmap.yaml"
  command="load" ns="safe_teleop_costmap"/>
```

```
25 <!-- load parameter file -->
26 <rosparam file="$(arg config_path)/
  collision_velocity_filter_params.yaml" command="load"/>
27 </node>
28 </group>
29
30 </launch>
```

Listing C-2. YAML parameters for the collision velocity filter (collision_velocity_filter.yaml).

```
1 #frequency at which the get_footprint service should be called
2 footprint_update_frequency: 0.5
3
4 #Parameters specifying slow down behaviour
5 pot_ctrl_vmax: 0.85 #default: 0.6
6 pot_ctrl_vtheta_max: 8.0 #default: 0.8
7 pot_ctrl_kv: 1.0 #damping default: 1.0
8 pot_ctrl_kp: 2.0 #stiffness default: 2.0
9 pot_ctrl_virt_mass: 0.5 #default: 0.8
10 max_acceleration: [0.5, 0.5, 4.7]
11
12 #Parameters specifying collision velocity filter
13 # global_frame: map
14 # robot_base_frame: base_link
15 costmap_obstacle_treshold: 250 # costmap cost threshold for
  obstacles
16 influence_radius: 2.0 #[m] distance from robot_center
17 obstacle_damping_dist: 0.10 # used as slow-down dist
18 stop_threshold: 0.03 #[m]
19 use_circumscribed_threshold: 9.0 #[rad/s] scan for obstacles
  in circumscribed radius of robot even when tube filter is ena-
  bled during fast rotations
20 enable_override_button: 11 # use the menu button on xbox
  controller for override
```

Listing C-3. YAML parameters for the collision velocity filter costmap (safe_teleop_costmap.yaml).

```
1 # global information
2 global_frame: base_link
3 robot_base_frame: base_link
4 update_frequency: 2.0
5 publish_frequency: 1.0
6
7 # local map settings
8 rolling_window: true
9 width: 5.0
10 height: 5.0
```

```
11 resolution: 0.05
12
13 # footprint and range
14 footprint: [[-0.34, -0.19], [-0.34, 0.19], [0.34, 0.19],
   [0.34,-0.19]]
15 footprint_padding: 0.01
16 transform_tolerance: 0.5
17
18 #layers
19 plugins:
20
21 # The obstacle layer tracks the obstacles as read by the sen-
   sor data. The ObstacleCostmapPlugin marks and
22 # raytraces obstacles in two dimensions, while the
VoxelCostmapPlugin does so in three dimensions.
23 # - {name: obstacle_layer_0, type: "costmap_2d::Obsta-
   cleLayer"}
24 - {name: safe_teleop_layer, type: "costmap_2d::ObstacleLayer"}
25 - {name: safe_inflater_layer, type: "costmap_2d::Infla-
   tionLayer"}
26
27 safe_teleop_layer:
28
29 enabled: true
30
31 # Definition of the sensors/observation sources
32 observation_sources: scan_unified_filtered
33 # observation_sources: point_cloud_rgbd point_cloud_rgbd_rear
34
35 # # Parameters of the source
36 scan_unified_filtered:
37 topic: /safe_teleop/passthrough_z/output # The topic on which
sensor data comes in for this source, default: source_name
38 data_type: PointCloud2 # The data type associated
   with the topic, default: "PointCloud"
39 marking: true # Whether or not this
   observation is used to mark obstacles, default: true
40 clearing: true # Whether or not this
   observation is used to clear out freespace, default: false
41 inf_is_valid: false # Allows for Inf values in
   "LaserScan" observation messages. The Inf values are
   converted to the laser maximum range. default: false
42
43 # The default maximum distance from the robot at which an
   obstacle will be inserted into the costmap in meters
44 obstacle_range: 2.5 # default: 2.5
```

```
45
46 # The default range in meters at which to raytrace out obsta-
   cles from the map using sensor data
47 raytrace_range: 3.0 # default: 3.0
48
49 # If false, each pixel has one of 2 states: lethal obstacle or
   free. If true, each pixel has one of 3 states: lethal obstacle,
   free, or unknown
50 track_unknown_space: false # default: false
51
52 # If true, the robot footprint will clear (mark as free) the
   space in which it travels.
53 footprint_clearing_enabled: false # default: true
54
55 # Whether keep unknown as unknown or clear it
56 track_unknown: false
57
58 safe_inflater_layer:
59 inflation_radius : 0.01
60 cost_scaling_factor: 3.0
```

Listing C-4. Launch file for navigation and path planning stack.

```
1 <launch>
2 <arg name="repo_path" default="/home/nuc/MOWLES/IRL/
  preTest/rover_demo" />
3 <arg name="cmd_vel_topic" default="/cmd_vel" />
4 <arg name="odom_topic" default="/cam_t265_1/odom/sample" />
5 <arg name="node_start_delay" default="0.0" />
6
7 <group ns="rgbd">
8 <node pkg="nodelet" type="nodelet" name="
  pcl_manager" args="manager" output="screen" />
9 <node pkg="nodelet" type="nodelet" name="voxel_grid
  " args="load pcl/VoxelGrid pcl_manager" output="screen">
10 <remap from="~input" to="/cam_d435_1/depth/color/points" />
11 <rosparam>
12 filter_field_name: z
13 filter_limit_min: 0.15
14 filter_limit_max: 0.85
15 filter_limit_negative: False
16 leaf_size: 0.01
17 input_frame: base_link
18 output_frame: base_link
19 </rosparam>
20 </node>
21 </group>
```

```
22
23 <group ns="rgbd_rear">
24 <node pkg="nodelet" type="nodelet" name="
  pcl_manager" args="manager" output="screen" />
25 <node pkg="nodelet" type="nodelet" name="voxel_grid
  " args="load pcl/VoxelGrid pcl_manager" output="screen">
26 <remap from="~input" to="/cam_d435_2/depth/color/points" />
27 <rosparam>
28 filter_field_name: z
29 filter_limit_min: 0.15
30 filter_limit_max: 0.85
31 filter_limit_negative: False
32 leaf_size: 0.01
33 input_frame: base_link
34 output_frame: base_link
35 </rosparam>
36 </node>
37 </group>
38
39 <group ns="lidar">
40 <node pkg="nodelet" type="nodelet" name="
  pcl_manager" args="manager" output="screen" />
41 <node pkg="nodelet" type="nodelet" name="voxel_grid
" args="load pcl/VoxelGrid pcl_manager" output="screen">
42 <remap from="~input" to="/os_cloud_node/points"/>
43 <rosparam>
44 filter_field_name: z
45 filter_limit_min: 0.15
46 filter_limit_max: 0.85
47 filter_limit_negative: False
48 leaf_size: 0.1
49 input_frame: base_link
50 output_frame: base_link
51 </rosparam>
52 </node>
53 </group>
54
55 <node pkg="move_base" type="move_base" respawn="false
  " name="move_base" output="screen" launch-prefix="
  bash -c ' sleep $(arg node_start_delay); $0 $@'  ">
56 <rosparam file="$(arg repo_path)/move_base/config/
  common_costmap_rear.yaml" command="load" ns="global_costmap"/>
57 <rosparam file="$(arg repo_path)/move_base/config/
  common_costmap_rear.yaml" command="load" ns="local_costmap" />
58 <rosparam file="$(arg repo_path)/move_base/config/
```

```
  global_costmap.yaml" command="load"/>
59 <rosparam file="$(arg repo_path)/move_base/config/
local_costmap.yaml" command="load" />
60 <rosparam file="$(arg repo_path)/move_base/config/
teb_local_planner.yaml" command="load" />
61 <rosparam file="$(arg repo_path)/move_base/config/
move_base.yaml" command="load" />
62
63 <remap from="cmd_vel" to="$(arg cmd_vel_topic)"/>
64 <remap from="odom" to="$(arg odom_topic)"/>
65 <remap from="map" to="/rtabmap/grid_map"/>
66 </node>
67 </launch>
```

Listing C-5. YAML parameters shared by both local and global costmaps (common_costmap_rear.yaml).

```
1 max_obstacle_height: 9999
2 footprint: [[-0.34, -0.19], [-0.34, 0.19], [0.34, 0.19],
  [0.34,-0.19]]
3 footprint_padding: 0.03
4 transform_tolerance: 0.5
5
6 virtual_layer:
7 enabled: true
8 zone_topics: [/virtual_costamp_layer/zone]
9 obstacle_topics: [/virtual_costamp_layer/obsctacles]
10
11 inflation_layer:
12 cost_scaling_factor: 10.0
13 inflation_radius: 0.06
14
15 obstacle_layer:
16 track_unknown_space: true
17 obstacle_range: 4.9
18 raytrace_range: 5.0
19 observation_sources: point_cloud_lidar point_cloud_rgbd
  point_cloud_rgbd_rear
20 point_cloud_lidar: {data_type: PointCloud2, expected_up-
  date_rate: 0.5, topic: /lidar/voxel_grid/output, marking: true,
  clearing: true, max_obstacle_height: 9999, min_obstacle_height
  : -9999}
21 point_cloud_rgbd: {data_type: PointCloud2, expected_up-
  date_rate: 0.5, topic: /rgbd/voxel_grid/output, marking: true,
  clearing: true, max_obstacle_height: 9999, min_obstacle_height:
  -9999}
22 point_cloud_rgbd_rear: {data_type: PointCloud2,
```

```
  expected_update_rate: 0.5, topic: /rgbd_rear/voxel_grid/output,
  marking: true, clearing: true, max_obstacle_height: 9999,
  min_obstacle_height: -9999}
23
24 GlobalPlanner:
25 allow_unknown: true
26 cost_factor: 3
27 neutral_cost: 50
28 lethal_cost: 253
29 old_navfn_behavior: false
30 use_dijkstra: true
31 use_quadratic: true
32 use_grid_path: false
33 publish_potential: true
34 outline_map: false
```

Listing C-6. YAML parameters for the global costmap (global_costmap.yaml).

```
1 global_costmap:
2 global_frame: map
3 robot_base_frame: base_link
4 update_frequency: 10.0
5 publish_frequency: 10.0
6
7 plugins:
8 - {name: static_layer, type: "rtabmap_ros::StaticLayer"}
9 - {name: obstacle_layer, type: "costmap_2d::ObstacleLayer"}
10 - {name: inflation_layer, type: "costmap_2d::InflationLayer"}
11 - { name: virtual_layer, type: "virtual_costmap_layer::
  VirtualLayer" }
```

Listing C-7. YAML parameters for the local costmap (local_costmap.yaml).

```
1 local_costmap:
2 global_frame: cam_t265_1_odom_frame
3 robot_base_frame: base_link
4 update_frequency: 10.0
5 publish_frequency: 10.0
6 rolling_window: true
7 width: 7
8 height: 7
9 resolution: 0.05
10
11 plugins:
12 - {name: obstacle_layer, type: "costmap_2d::ObstacleLayer"}
13 - {name: inflation_layer, type: "costmap_2d::InflationLayer"}
14 - { name: virtual_layer, type: "virtual_costmap_layer::
  VirtualLayer" }
```

Listing C-8. YAML parameters for move base (move_base.yaml).

```
shutdown_costmaps: false
controller_frequency: 10.0
controller_patience: 5.0
planner_frequency: 1.0
planner_patience: 5.0
oscillation_timeout: 10.0
oscillation_distance: 0.2
max_planning_retries: 10

base_local_planner: "teb_local_planner/TebLocalPlannerROS"

base_global_planner: "global_planner/GlobalPlanner"

recovery_behaviors: [
{name: conservative_clear_1, type: clear_costmap_recovery/
  ClearCostmapRecovery},
{name: aggressive_clear_1, type: clear_costmap_recovery/
  ClearCostmapRecovery},
]
conservative_clear_1:
reset_distance: 1.0
aggressive_clear_1:
reset_distance: 0.0
```

Listing C-9. YAML parameters for Timed Elastic Band local planner (teb_local_planner.yaml).

```
TebLocalPlannerROS:
odom_topic: /cam_t265_1/odom/sample
map_frame: map
teb_autosize: True
footprint_model:
type: "line"
line_start: [-0.15, 0.0] # for type "line"
line_end: [0.15, 0.0] # for type "line"
vertices: [[-0.34, -0.19], [-0.34, 0.19], [0.34, 0.19],
 [0.34, -0.19]] # for type "polygon"

# surge
max_vel_x: 0.7
max_vel_x_backwards: 0.65
allow_init_with_backwards_motion: True

# sway
max_vel_y: 0.0

```

```
19 # yaw
20 max_vel_theta: 2.0
21
22 # acceleration
23 acc_lim_x: 2.0
24 acc_lim_y: 0.0
25 acc_lim_theta: 2.0
26
27 dt_ref: 0.5
28 dt_hysteresis: 0.1
29 global_plan_overwrite_orientation: True
30 max_global_plan_lookahead_dist: 5.0
31 feasibility_check_no_poses: 4
32 no_inner_iterations: 4
33 no_outer_iterations: 3
34 max_number_classes: 2
35
36 weight_kinematics_forward_drive: 5
37
38 # GoalTolerance
39 xy_goal_tolerance: 0.2
40 yaw_goal_tolerance: 0.2
41 free_goal_vel: False
42 global_plan_viapoint_sep: 2.5
43
44 # Obstacles
45 include_costmap_obstacles: True
46 costmap_obstacles_behind_robot_dist: 1.0
47 obstacle_poses_affected: 10
48 min_obstacle_dist: 0.21 # for line needs to include robot
  footprint
49 inflation_dist: 0.065
50 include_dynamic_obstacles: True
51 obstacle_association_force_inclusion_factor: 1.5
52 obstacle_association_cutoff_factor: 5
53
54 ## Costmap converter plugin
55 costmap_converter_spin_thread: True
56 costmap_converter_rate: 2
57 costmap_converter_plugin: "costmap_converter::
  CostmapToPolygonsDBSMCCH"
58
59 ## Configure plugins (namespace move_base/TebLocalPlannerROS/
  PLUGINNAME)
60 ## The parameters must be added for each plugin separately
```

```
61 costmap_converter/CostmapToLinesDBSRANSAC:
62 cluster_max_distance: 0.3
63 cluster_min_pts: 2
64 cluster_max_pts: 20
65 ransac_inlier_distance: 0.2
66 ransac_min_inliers: 10
67 ransac_no_iterations: 2000
68 ransac_remainig_outliers: 3
69 ransac_convert_outlier_pts: True
70 ransac_filter_remaining_outlier_pts: False
71 convex_hull_min_pt_separation: 0.1
72
73 costmap_converter/CostmapToPolygonsDBSMCCH:
74 cluster_max_distance: 0.3
75 cluster_min_pts: 2
76 cluster_max_pts: 20
77 convex_hull_min_pt_separation: 0.1
```

Appendix D: Exploration Launch Files

In this appendix, we list the launch files used for our exploration and lost connectivity approach.

Listing D-1. Launch file for exploration.

```
1 <launch>
2 <arg name="node_start_delay" default="0.0" />
3 <node pkg="explore_lite" type="explore" respawn="false"
 name="explore" output="screen" launch-prefix="bash
 -c ' sleep $(arg node_start_delay); $0 $@'  ">
4 <param name="robot_base_frame" value="base_link"/>
5 <param name="costmap_topic" value="move_base/
 global_costmap/costmap"/>
6 <param name="costmap_updates_topic" value="move_base/
 global_costmap/costmap_updates"/>
7 <param name="visualize" value="true"/>
8 <param name="planner_frequency" value="1.0"/>
9 <param name="progress_timeout" value="20.0"/>
10 <param name="potential_scale" value="3.0"/>
11 <param name="orientation_scale" value="0.0"/>
12 <param name="gain_scale" value="1.0"/>
13 <param name="transform_tolerance" value="0.5"/>
14 <param name="min_frontier_size" value="0.2"/>
15 </node>
16 </launch>
```

Listing D-2. Launch file for lost connectivity.

```
1 <launch>
2 <env name="ROSCONSOLE_FORMAT" value="[${severity
 }][${thread}][${node}/${function}:${line}]: ${
 message}"/>
3
4 <arg name="ips_to_monitor" default="hpomen"/> <!--
 using hostname (here defined in /etc/hosts) works
 too -->
5
6 <node pkg="lost_comms_recovery" type="
 lost_comms_recovery" name="lost_comms_recovery"
 clear_params="true" output="screen" respawn="true" >
7 <param name="ping_fail_count" value="20"/>
8 <param name="goal_frame_id" value="map"/>
9 <param name="ips_to_monitor" value="$(arg ips_to_monitor)"/>
10 <!-- INPUT TOPICS -->
11 <remap from="recovery_pose" to="recovery_pose" />
```

```
12 <!-- OUTPUT TOPICS -->
13 <remap from="cmd_vel" to="cmd_vel" /> <!—zero motors -->
14 <remap from="joy" to="joy" /> <!-- zero joystick -->
15 </node>
16 </launch>
```

Abbreviations

AGC	Army Geospatial Center
AI	Artificial intelligence
HW	Hardware
ICP	Iterative closest point
IMU	Inertial measurement unit
ML	Machine learning
MSSPIX	Maneuver Support, Sustainment and Protection Integration eXperiments
RGB-D	Red-green-blue and depth
RMSE	Root mean square error
ROS	Robot Operating System
RTAB-Map	Real-Time Appearance-Based Mapping
SLAM	Simultaneous localization and mapping
SubT	Subterranean
SWaP-C	Size, weight, power, and cost
UGV	Unmanned ground vehicle

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
September 2023	Final Technical Report	FY2020	FY2022

4. TITLE AND SUBTITLE

Low Size, Weight, Power and Cost (SWaP-C) Payload for Autonomous Navigation and Mapping on an Unmanned Ground Vehicle

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT
		0602146A
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**
AT9	01	

6. AUTHOR(S)

Osama Ennasr, Brandon Dodd, Michael Paquette, Charles Ellison, and Garry Glaspell

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

US Army Engineer Research and Development Center (ERDC)
Geospatial Research Laboratory (GRL)
7701 Telegraph Road
Alexandria, VA 22315-3864

US Army Engineer Research and Development Center (ERDC)
Information Technology Laboratory (ITL)
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

8. PERFORMING ORGANIZATION REPORT NUMBER

ERDC TR-23-17

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Headquarters, US Army Corps of Engineers Washington, DC 20314-1000	USACE	

12. DISTRIBUTION/AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Autonomous navigation and unknown environment exploration with an unmanned ground vehicle (UGV) is extremely challenging. This report investigates a mapping and exploration solution utilizing low size, weight, power, and cost payloads. The platform presented here leverages simultaneous localization and mapping to efficiently explore unknown areas by finding navigable routes. The solution utilizes a diverse sensor payload that includes wheel encoders, 3D lidar, and red-green-blue and depth cameras. The main goal of this effort is to leverage path planning and navigation for mapping and exploration with a UGV to produce an accurate 3D map. The solution provided also leverages the Robot Operating System.

15. SUBJECT TERMS

Cameras; Detectors; Geospatial data; Military robots; Military surveillance; Optical radar; Optics; Remote sensing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
Unclassified	Unclassified	Unclassified	SAR	60

19a. NAME OF RESPONSIBLE PERSON	19b. TELEPHONE NUMBER (include area code)