ERDC/GRL TR-23-3

Geospatial Research Laboratory

US Army Corps of Engineers®
Engineer Research and Development Center

Tactical Geospatial Information Capabilities (TGIC)

UGV SLAM Payload for Low-Visibility Environments

Osama Ennasr, Mike Paquette, and Garry Glaspell

September 2023

UGV SLAM Payload for Low-Visibility Environments

Osama Ennasr, Mike Paquette, and Garry Glaspell

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

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," task "Tactical Data Generation and Processing"

Abstract

Herein, we explore using a low size, weight, power, and cost unmanned ground vehicle payload designed specifically for low-visibility environments. The proposed payload simultaneously localizes and maps in GPS-denied environments via waypoint navigation. This solution utilizes a diverse sensor payload that includes wheel encoders, inertial measurement unit, 3D lidar, 3D ultrasonic sensors, and thermal cameras. Furthermore, the resulting 3D point cloud was compared against a survey-grade lidar.

Contents

Figures

Preface

This study was conducted for the US Army Corps of Engineers under Program Element Number 0602146A, Project AT9, Task 01, "Tactical Geospatial Information Capabilities," task "Tactical Data Generation and Processing." 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. 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 authors would like to acknowledge the following individuals for their contributions to this project: Mr. Chuck Ellison, Mr. Steven Bunkley, Mr. Chuck Dickerson, and Mr. Richard Brown.

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

1 Introduction

1.1 Background

In 2017, Defense Advanced Research Projects Agency (DARPA) created the SubT Challenge (DARPA 2017) to develop innovative technologies that can augment underground operations. They focused on four primary areas: autonomy, perception, networking, and mobility. Under the perception task, possible hazards included dust, fog, mist, water, smoke, low light, and obscured and/or scattering environments. The focus of this report is to leverage a low size, weight, power, and cost (SWaP-C) payload to mitigate as many of the aforementioned hazards as possible.

1.2 Objectives

This report addresses the Army Multi-Domain Intelligence FY21–22 Science and Technology (S&T) 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 (RAS), 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

Our approach to develop an unmanned ground vehicle (UGV) payload that can operate in low-visibility environments utilizes both hardware and software modifications of our previously reported setup (Glaspell et al. 2020). Regarding hardware, we still use a 3D lidar. However, we swapped our RGB-D (red-green-blue and depth) cameras for thermal cameras. We also included 3D ultrasonic sensors for obstacle avoidance. For software, we still leverage RTAB-Map (Real-Time Appearance-Based Mapping) for SLAM (Simultaneous Localization and Mapping) but include `cmr_lidar-loop`, which uses laser scans for loop detection. We have also switched from using `move_base` to `move_base_flex` for waypoint navigation. Finally, we removed the t265 camera that we typically use for odometry, since it would not work in low visibility environments. Thus, for indoor navigation, we use `laser_scan_matcher` for odometry. For outdoor navigation, we used the UGV’s wheel encoders and an inertial measurement unit (IMU).

2 Sensor Setup and Description

2.1 Installation

This report leverages the Robot Operating System (ROS)—specifically Melodic Morenia. When we started the work, one of the packages, `cmr_lidarloop` did not have Noetic/Python 3 support. Support was later made available on 20 April 2022. As a result, this solution should run on ROS Noetic Ninjemys, but this has not been verified by the authors. ROS melodic has support until 2023, while Noetic has support until 2025. Instructions for installing ROS Melodic can be found at http://wiki.ros.org/melodic/Installation/Ubuntu.

It is assumed that the following packages, and their prerequisites, are installed in the catkin ws/src folder:

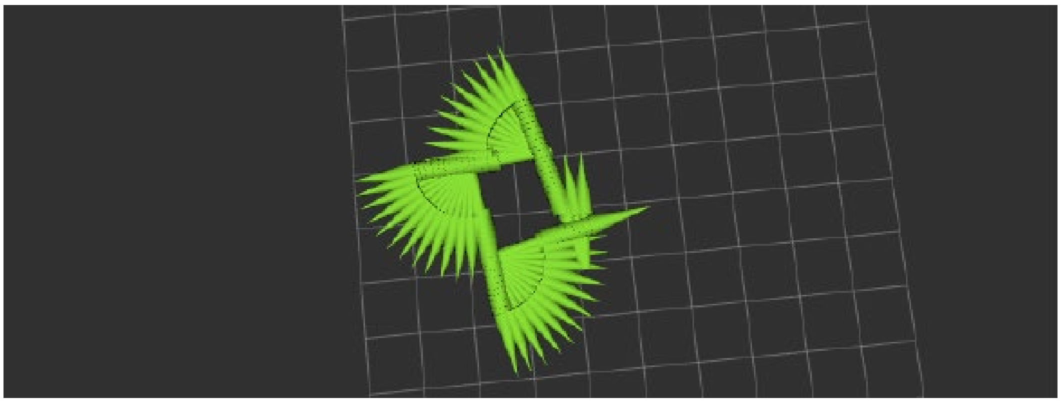
- `cmr_lidarloop` (MarvinStuede 2022)
- `ds4drv` (chrippa 2018)
- `flir_boson_usb` (astuff 2021)
- `m-explore` (hrnr 2021)
- `rr_openrover_stack` (RoverRobotics 2021)
- `rtabmap_ros` (introlab 2022)

It should be noted that `rr_openrover_stack` is deprecated in favor of Roverrobotics `ros1`. However, the launch files contained in this report were written for the older `rr_openrover_stack`. Also, while it is possible to install `rtabmap_ros` from the online repository, it is strongly advised to compile `rtabmap_ros` from source. Compiling from source allows for multi-camera support.

2.2 Rover

The first launch file that we typically execute is for the rover platform. The `bringup.launch` file is responsible for initiating the openrover driver, `twist_mux`, and the control input manager. There are several parameters included with the openrover driver. The default values are good out of the box. However, the parameter traction factor should be tailored to the individual robot and its operating environment. In our experience, this value changed significantly when using wheels versus tracks. Terrain affected the value as well. To test the validity of the assigned traction factor, we typically drove the bot in a "square" pattern and observed the topic `/rr_openrover_driver/odom_encoder` in RViz, an official 3D visualization tool of ROS. If the odometry topic in RViz, did not match the robot-driven pattern, we modified the traction factor and repeated the test. An example of the box test is shown in Figure 1.

Figure 1. Box test.



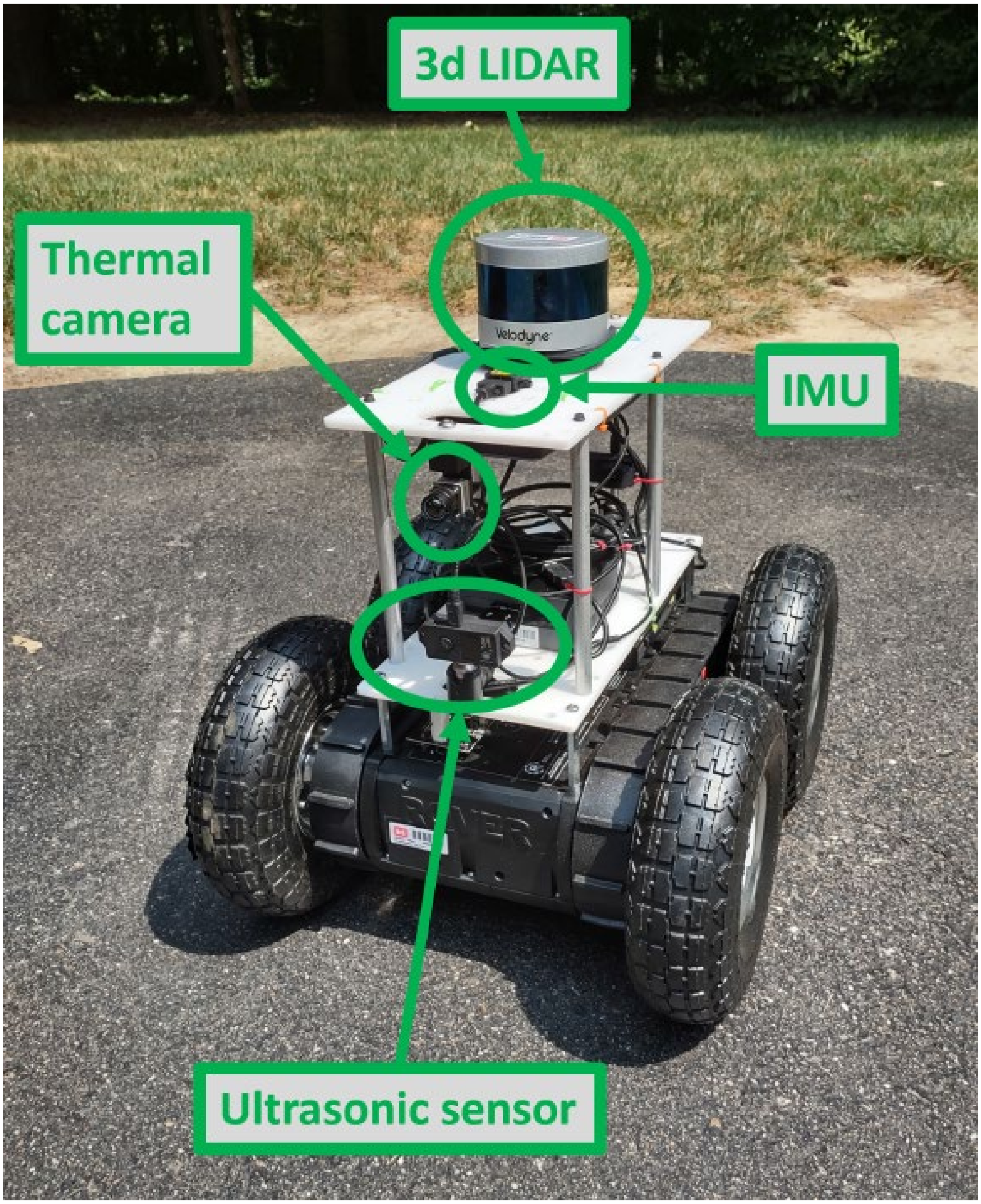
For our four-wheel drive robot, on concrete, the optimal value was 0.45. Conversely, when using tracks on concrete, the optimal value was 0.69. The drive type was another often-changed parameter. If using tracks, the drive type was set to "flipper." If using wheels, the drive type was set to "4wd." Also included in the bringup.launch was `twist_mux`, which allowed for multiple inputs when driving the robot. The default setup for `twist_mux` gave the highest priority to a joystick. This was particularly useful when the robot was stuck driving autonomously. The control input manager parameter was used to remap inputs for the joystick, depending on whether a Playstation 4 or Xbox joystick was used. For completeness, the udev device manager rule for the rover is listed below. The udev rule sets the symbolic links (symlinks), which point to files or directories, for both the rover and the joystick.

```
# creates fixed name for rover serial communication
KERNEL=="ttyUSB*", ATTRS{idVendor}=="0403", ATTRS{
   idProduct}=="6001", MODE:="0777", SYMLINK+="rover",
   RUN+="/bin/setserial /dev/%k low_latency"
KERNEL=="ttyUSB*", ATTRS{idVendor}=="0403", ATTRS{
   idProduct}=="6015", MODE:="0777", SYMLINK+="rover",
   RUN+="/bin/setserial /dev/%k low_latency"
# create fixed mapping for xbox control to avoid
   inconsistent naming
```

```
SUBSYSTEM=="input", KERNEL=="js*", ATTRS{name}=="Xbox
   Gamepad (userspace driver)", SYMLINK="input/jsX"
```

The complete modified `bringup.launch` file is provided in Appendix A. After the robot is brought online, we typically launch the files for the various sensors. For this build, we leveraged a Velodyne VLP 16 lidar, a Lord 3DMGX5-AHRS IMU, two Teledyne Forward-Looking Infrared (FLIR) Boson 320 thermal cameras, and two Toposens 3D ultrasonic sensors. Figure 2 shows the UGV with the sensors attached. Note that a rear thermal camera and rear ultrasonic sensor were also mounted but cannot be seen in the image.

Figure 2. Unmanned ground vehicle (UGV) with mounted sensors.



2.3 Lidar

To bring up the Velodyne lidar, we launched the VLP16 `points.launch` file. The default launch file had the lidar running at 600 rpm. This published the point cloud topic at approximately 10 hz. However, for our purposes, we increased the rpm to 1,200. As a result, the resulting point cloud topic was published at a rate of approximately 20 hz. The higher rotation rate was desirable since we are planning to use the lidar to help localize the robot. The complete launch file for Velodyne lidar is located in Appendix B. The lidar served two purposes—it was used to create 3D point clouds of the environment and also to localize the robot when navigating indoors. The launch file also contained the laserscan nodelet. This nodelet produced a 2D laser scan topic. Although it is possible to use this laser scan topic for localization, an alternative method is described in the subsequent launch files.

2.4 IMU

The Microstrain IMU has an axis printed on the label. However, the axis on the label is for North, East, Down (NED) coordinates. ROS expects East, North, Up (ENU) coordinates for the IMU. Note the microstrain ROS wrapper performed the NED to ENU transformation automatically by swapping x with y and negating z. Alternatively, setting the parameter frame-based ENU to true matched the orientation of the printed label to the ENU coordinate system. We also added the parameter `remove_imu_gravity` and set the value to true. It is important to note that the remove gravity parameter can also be set in the `robot_localization` launch file as well. Another important parameter was declination source. For outdoor environments, the value was set to two to align the heading with the magnetic north. In an indoor environment, we typically set the value to one, which equated to "none" or no heading correction. The complete launch file for the IMU is provided in Appendix C. The primary function of the IMU was to localize the robot.

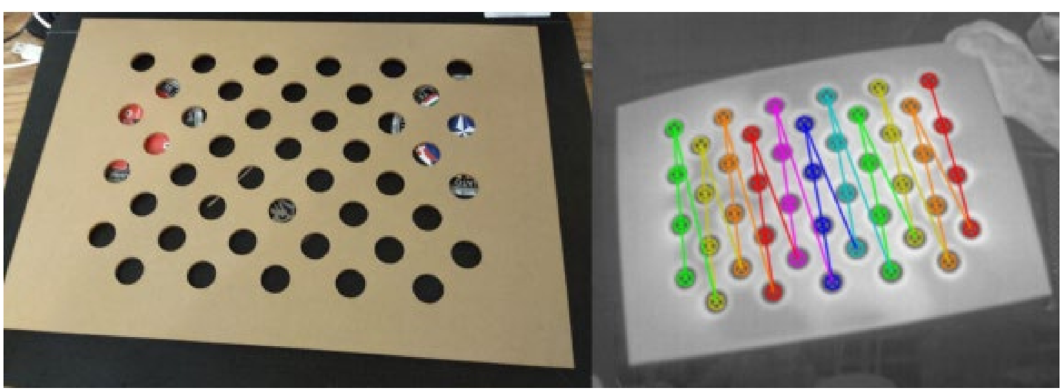
2.5 Thermal Cameras

The front FLIR Boson thermal camera was launched using the `flir_boson` rectified `front.launch` file. The complete launch file is provided in Appendix D. Since we are leveraging both front and rear thermal cameras, we wrote a launch file for the rear camera as well. The key differences between the front and rear launch files include parameters for namespace, `frame_id`, and dev to reflect either the front or rear camera. For the dev parameter, we added a rule to provide symlinks regardless of which USB port the cameras are plugged into or which camera is plugged in. As a result, `/dev/boson_f` and `/dev/boson_r` will always refer to the front and rear thermal cameras respectively. The rule to assign the symlinks is provided below. Inclusion of `ATTRindex}=="0"` ensures that the camera feed, published on `/dev/video0` is selected, rather than the metadata, published on `/dev/video1`.

```
SUBSYSTEMS=="usb", ATTR{index}=="0", ATTRS{idProduct}==
   "4007", ATTRS{idVendor}=="09cb", ATTRS{manufacturer
   }=="FLIR", ATTRS{product}=="Boson", ATTRS{serial}=="
   108945", MODE="0666", GROUP="video", SYMLINK+="
   bosonf"
SUBSYSTEMS=="usb", ATTR{index}=="0", ATTRS{idProduct}==
   "4007", ATTRS{idVendor}=="09cb", ATTRS{manufacturer
   }=="FLIR", ATTRS{product}=="Boson", ATTRS{serial}=="
   112201", MODE="0666", GROUP="video", SYMLINK+="
   bosonr"
```

To ensure that we can colorize the lidar point cloud with the thermal image, calibration of the thermal cameras was required. We also had to laser cut a calibration board since the thermal cameras could not use the calibration pattern used for RGB cameras. The laser-cut calibration board is shown in Figure 3. We used a hair dryer to generate a temperature difference between the board and the cut holes. Calibration was performed using the camera calibration found at https://github.com/ros-perception/image pipeline. The calibrated YAML file for the Boson 320 camera is listed in Appendix D.

Figure 3. Calibration of the thermal camera.

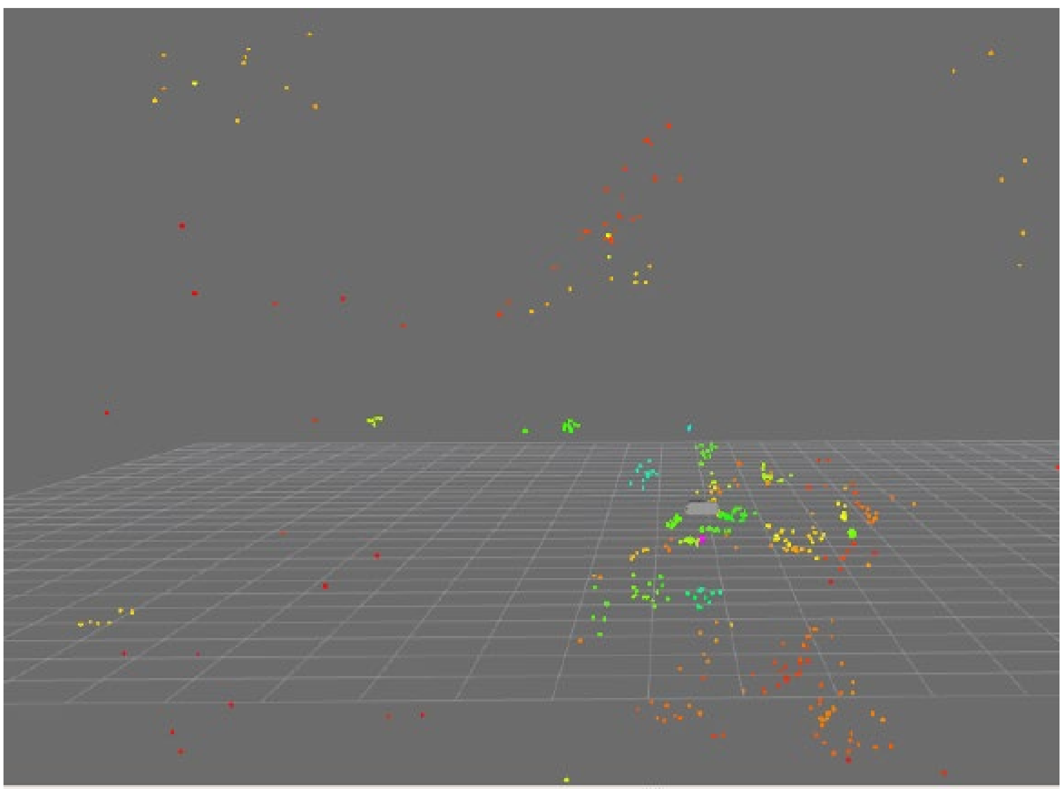


The full launch file for the thermal cameras is located in Appendix E. In these examples, the namespace and `frame_id` parameters were set for the front-facing camera. The `dev` param was set to match the SYMLINK provided in the udev rules. Also, the `camera_info_url` was set to the YAML file generated during the calibration step. A seperate launch file was created for the rear-facing camera. Once launched, the topic `/flir_boson_f/image_rect` can be viewed in RViz.

2.6 3D Ultrasonic Sensors

The purpose of this build is to operate without ambient light. Thus, we have replaced the RGB-D cameras that we typically use with Toposens 3D Ultrasonic Sensors 4. Specifically we are using the 2019 prototype model on the front and rear of the robot. The range of the sensors is 4 m, which is on par with the RGB-D cameras. The primary function of the ultrasonic sensors was to mark and clear obstacles from the costmap. A representative point cloud produced by the Toposens 3D ultrasonic sensor is shown in Figure 4.

Figure 4. Point cloud from an ultrasonic sensor.



Similarly to the thermal camera setup, udev rules were written for the ultrasonic sensors. Again, the udev rules generate SYMLINKS that can be used to identify the front or rear sensor regardless of which USB port they are plugged into. The udev rules are provided below.

```
SUBSYSTEMS=="usb", ATTRS{idProduct}=="ea60", ATTRS{
```

```
    idVendor}=="10c4", ATTRS{product}=="CP2102N USB to
    UART Bridge Controller", ATTRS{manufacturer}=="
    Silicon Labs", ATTRS{serial}=="58
    ddeb5e94bfe811993011373b0549ec", MODE="0666", GROUP=
    "dialout", SYMLINK+="ts3f"
SUBSYSTEMS=="usb", ATTRS{idProduct}=="ea60", ATTRS{
    idVendor}=="10c4", ATTRS{product}=="CP2102N USB to
    UART Bridge Controller", ATTRS{manufacturer}=="
    Silicon Labs", ATTRS{serial}=="80
    e53e6e6bbfe811a2f910373b0549ec", MODE="0666", GROUP=
    "dialout", SYMLINK+="ts3r"
```

The complete launch file for the front ultrasonic sensor is provided in Appendix F. The parameters for `frame_id` and `target_frame` are provided for the front-facing sensor. Also, the port is set to the symlink provided in the udev rule. For the rear ultrasonic sensor, a separate launch file is required. The `sensor_params.yaml` file is provided below. The values are tuned to reduce false negatives. The tradeoff is smaller obstacles may not be detected. This is necessary since phantom obstacles would adversely affect the UGV's path planner to avoid obstacles that are not really there. Also, the lidar is also used to identify obstacles and can detect the smaller obstacles missed by the ultrasonic sensor. While the lidar has better sensitivity, for detecting obstacles, the ultrasonic sensors prevail in foggy or dusty environments.

```
echo_rejection_threshold: 20
num_pulses: 7
peak_detection_window: 1
```

3 Software

3.1 Robot Localization

The robot localization package (Moore and Stouch 2014) is useful for fusing an arbitrary number of sensors using Kalman Filters. In short, odom data, IMU data, and pose and twist estimates can be combined to localize the robot in its environment. In this report, we assume the coordinate system below with robot localization publishing the map → odom → base link transformation. The base link is the frame associated with the robot. The base link frame is not considered to be a fixed frame and will move with the robot. All the sensors are attached to the `base_link` frame via static transforms. For our UGV, the static transforms for each of the sensors are provided below. The odom frame links to `base_link` and is considered a fixed frame even though it can drift over time. The map frame is another fixed frame that should not drift over time but may jump periodically if using a GPS.

```
<launch>
<!-- LIDAR -->
  <node pkg="tf" type="static_transform_publisher" name
    ="base_to_velo"
        args="0 0 0.4699 0 0 0 /base_link /velodyne 100"

  <!-- IMU -->
   <node pkg="tf" type="static_transform_publisher" name
="base_to_gx5"
         args="-0.08255 0 0.40005 0 0 0 /base_link /
gx5_link 1000" />

  <!-- Front and rear boson setup -->
   <node pkg="tf" type="static_transform_publisher" name
="base_to_boson_f"
         args="0.187325 0 0.3556 -1.57 0 -1.57 /base_link
/boson_camera_f 100" />

   <node pkg="tf" type="static_transform_publisher" name
="base_to_boson_r"
         args="-0.187325 0 0.3556 1.57 0 -1.57 /base_link
      /boson_camera_r 100" />
```

```
<!-- Front and rear ultrasonic sensors -->

    <node pkg="tf" type="static_transform_publisher" name
  ="base_to_ts3_f"
          args="0.187325 0 0.254 0 0 0 /base_link /
  toposens_f 100" />

  <node pkg="tf" type="static_transform_publisher" name
    ="base_to_ts3_r"
        args="-0.187325 0 0.254 -3.14159 0 0 /base_link /
    toposens_r 100" />
</launch>
```

The robot localization package provides both the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF). The general consensus is UKFs are slower but more accurate, and EKFs are faster. In this report we choose to use the EKF, prioritizing speed over accuracy. The fusion of the sensors publishes the filtered position, typically `/odometry/filtered`. The complete launch file is provided in Appendix G.

3.1.1 Outdoor Navigation

The robot localization package should be tuned to match your environment. For navigation outdoors, the assumption is there are regions of open space. As a result, we fuse wheel odometry and IMU together. The robot localization package allows tuning in 15 degrees of freedom: x, y, z, roll, pitch, yaw, vx, vy, vz, vroll, vpitch, vyaw, ax, ay, and az (“v” is for velocity and “a” is for acceleration). However, for wheel odometry, we set parameters “vx” and “vy” to true and everything else to false. Note if you do not have an IMU, then “vyaw” should be set to true as well. Since we are using an IMU, we set “vyaw” to false.

```
# Rover Odom
odom0: /rr_openrover_driver/odom_encoder
odom0_config: [false, false, false,
               false, false, false,
               true, true, false,
               false, false, false,
               false, false, false]
```

For the IMU, we set "yaw" and "vyaw" to true and everything else to false. This is because we assume our odometry is 2D. Note that we explicitly set 2D mode to true. The assumption is that our SLAM module will provide 3D pose correction. Our IMU settings are provided below. Conversely, if 2D mode is set to false, then roll, pitch, vroll, and vpitch should be set to true as well.

```
# IMU
imu0: /gx5/imu/data
imu0_config: [false, false, false,
              false, false, true,
              false, false, false,
              false, false, true,
              false, false, false]
```

The complete YAML file is provided in Appendix H. However, the file is written specifically for indoor navigation. To modify the YAML file for outdoor navigation, the contents below "Rover Odom" and "IMU" should be uncommented (remove the preceding #), and contents under "laser odom" should be commented out (add a preceding #).

3.1.2 Indoor Navigation

The assumption for indoor navigation is there are plenty of structures, such as walls and edges, in the immediate area to localize off of. The prevalence of these structures increases the chances that laser odometry performs well. Laser odometry is not recommended for outdoor navigation where there aren't enough structures to localize off of. However, in indoor environments, laser odometry is preferred over the wheel odom and IMU approach listed in the previous section. For laser odometry, we use the package `laser_scan_matcher` (CCNYRoboticsLab 2018). The launch file for `laser_scan_matcher` was already provided in Appendix G, thus we will only discuss the important parameters. First, contrary to the name, `laser_scan_matcher` can use point clouds. In our example, we remap the cloud parameter in the `laser_scan_matcher` node with the point cloud message `/voxel grid/output`. Note that `/voxel_grid/output` comes from the `/voxel_grid nodelet` that was listed in Appendix G. The `/voxel_grid` nodelet is responsible for taking the point cloud message `/velodyne_points` (raw lidar data) and only keeping the points between 15 cm and 100 cm in the *z* direction. This effectively removes the floor and ceiling from the /voxel_grid/output point cloud message. The laser_scan_matcher node can also use odom and IMU as a guess frame by setting 'use odom' and 'use imu' respectively to true. From testing, the best results were obtained using the IMU guess and not odom. The laser_scan_matcher node can output a number of different formats including pose, pose_stamped, pose_with_covariance, and pose_with_covariance_stamped. However, out of the various pose messages listed, robot localization only accepts the pose_with_covariance_stamped message type. As a result, we set the other pose parameters to false. The inclusion of the laser_scan_matcher node into robot localization is shown below. Specifically, we set *x*, *y*, and *yaw* to true and all other values to false. The complete YAML file for indoor navigation is provided in Appendix H and can be used without further modification.

```
#laser_odom
pose0: /pose_with_covariance_stamped
pose0_config: [true, true, false,
               false, false, true,
               false, false, false,
               false, false, false,
               false, false, false]
```

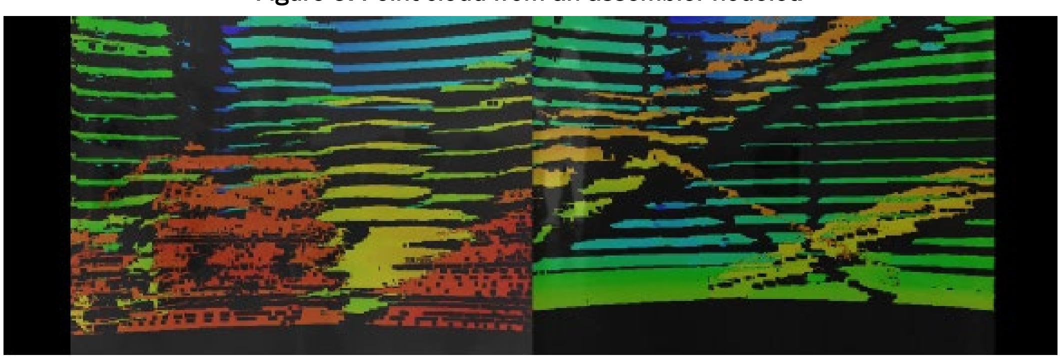
3.2 cmr_lidarloop

The cmr_lidarloop package is an extension of the SLAM module RTAB-Map. RTAB-Map will be discussed in length in the next section, but it is important to point out that RTAB-Map uses primarily visual data to correct the robot's pose. It is very accurate for single mapping sessions. However for multi session operation changes in lighting can diminish RTAB-Map's ability to loop close on revisited areas. Lidar is inherently illumination invariant, thus we can use use cmr_lidarloop to close loops regardless if the environmental lighting has changed. The full launch file for cmr_lidarloop is provided in Appendix I. In the launch file, the parameters for repo_path and cfg_file should be tailored to the user. The YAML file for identified by the pram_cfg file is provided in Appendix J. Most of the default values were used. The parameters scan_topic_name and odom_topic_name were modified to match our payload. Also, the parameter path clouds sets the location where cmr_lidarloop stores the point clouds used for scan registration.

3.3 RTAB-Map

In a previous publication, we compared different SLAM modules and described in detail why we prefer to use RTAB-Map (Glaspell et al. 2020). In this section, we focus on integrating `cmr_lidarloop` and our thermal cameras into RTAB-Map. The complete launch file is provided in Appendix K. The first nodelet listed in the `rtabmap_rover_cmr.launch` is the `point_cloud_assembler`. The `point_cloud_assembler` nodelet is responsible for concatenating point cloud messages, from the same topic, into a denser point cloud message. In our example, the number of point clouds aggregated is determined by the max clouds parameter, which we set to 10. Thus, each assembled cloud message from the `point_cloud_assembler` node is a combination of 10 individual `velodyne_points` scans. This is important since RTAB-Map typically writes to the database at a rate of 1 hz. This is controlled by the `Rtabmap/DetectionRate` parameter and the default value is 1. Consequently, if the lidar was running at 10 hz, we are only keeping 1 out of every 10 lidar scans. However, with the `point_cloud_assembler` nodelet running, we keep all 10 scans when writing to the database. Thus, the resulting 3D point clouds are 10 times denser when RTAB-Map writes to the database. If the lidar hz is set to 20, max clouds can be set to 20 as well. An example of a concatenated point cloud is shown in Figure 5. It is also noteworthy that the `point_cloud_assembler` nodelet subscribes to an odom topic. Therefore, the displacement of the robot is accounted for in the assembled point cloud when the robot is moving.

Figure 5. Point cloud from an assembler nodelet.

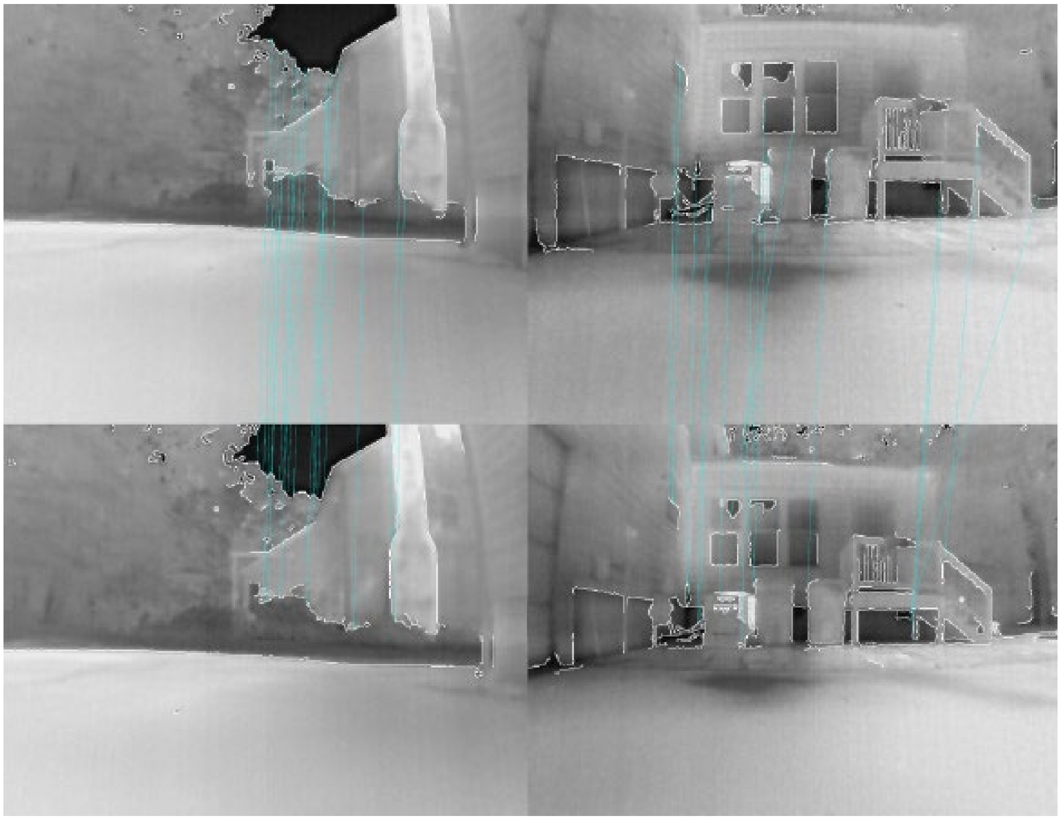


The second nodelet in the launch file is `pointcloud_to_depthimage_0`. This nodelet reprojects the point cloud from the `point_cloud_assembler` nodelet into the camera frame of the thermal camera to create a depth image. The `pointcloud_to_depthimage_0` nodelet is for the front thermal camera. This is repeated in the `pointcloud_to_depthimage_1` nodelet for the rear thermal camera.

The third nodelet listed in the launch file is `rgbd_sync_0`. Since the depth image produced by the `pointcloud_to_depthimage_0` nodelet publishes at a different rate than the thermal camera, the `rgbd_sync` nodelet ensures that the thermal image and the generated depth image are synchronized. The resulting `rgbd_image0` is passed to the RTAB-Map node. Note that the `pointcloud_to_depthimage_0` nodelet is for the front camera, and this process is repeated again for the rear camera.

The RTAB-Map node can ingest a variety of inputs including laser scans, point clouds RGB and depth images. In our example, we set `subscribe_rgbd` to true. We also set `rgbd_cameras` to 2 to account for both the front and rear thermal cameras. The topics `rgbd_image0` and `rgbd_image1` are passed as inputs. We also set `subscribe_scan_descriptor` to true since our point cloud message is coming from `cmr_lidarloop`. Specifically we remap `scan_descriptor` to `cmr_lidarloop/scan_descriptor`. The results can be seen in Figure 6. The two top images are from the front (left) and rear (rear) thermal cameras. The bottom images are from the same cameras after 1 second has elapsed. The blue lines connecting the top and bottom images represent the keyframes that are being tracked. Specifically, there are 17 keyframes tracked by the front thermal camera and 15 keyframes tracked by the rear thermal camera. The key frames are used to adjust the robot's pose. In this example, the `Kp/DetectorStrategy` parameter is set to 8 for GFTT/ORB. This uses both the Good Features to Track (GFTT) algorithm and Oriented FAST and rotated BRIEF (ORB) to track keyframes in the image. If more keyframes per scene are desired, the Kp/DetectorStrategy parameter can be set to 10 for ORB-Octree. This effectively doubles the number of generated keyframes. Both GFTT/ORB and ORB-Octree worked well with the thermal cameras. Note that we also set the parameter `Kp/RoiRatios` to 0.0 0.0 0.0 0.4. This prevents keyframes from being detected below the horizon.

Figure 6. Detected loop closures using the thermal camera.



Since we are using a 360-degree 3D lidar, we have set the `Reg/Strategy` to 1. Setting this value to 1 uses the Iterative Closest Point (ICP) algorithm for loop closure. This works well with our lidar since the field of view is large. Note that the keyframe transformation mentioned above is used as a guess for ICP.

For graph optimization there are a number of options, including Tree-based netwORk Optimizer (TORO), g2o, Georgia Tech Smoothing and Mapping (GTSAM), and Ceres. These are set by the `Optimizer/Strategy` parameter. Through testing, we have found that Ceres is the fastest, but GTSAM is the most accurate. The `Optimizer/Strategy` parameter was set to 2 for GTSAM (3 for Ceres).

Other parameters of note include `MaxObstacleHeight`, which is set to the height of the robot. This ensures that the generated occupancy grid generates obstacles that are relevant to the UGV and allows the UGV to drive into tunnels. Since we tested in an indoor environment, we set `MinGroundHeight` and `MaxGroundHeight` to act as a pass-through filter to remove the ground. Conversely, for outdoor navigation, where the ground is not flat, we would set `Grid/NormalsSegmentation` to true to account for uneven terrain. There are a number of other parameters listed in the RTAB-Map launch file. Appendix K retains the author's description of the various parameters. Also, the Wiki page for rtabmap-ros (https://wiki.ros.org/rtabmap_ros) provides good explanations and tutorials (Labbe 2022).

3.4 Move_base_flex

To get the robot from point A to point B we use move_base_flex (MBF) (Pütz et al. 2018). The primary motivation for switching to MBF is that it will allow us to leverage other map types, specifically meshes (Pütz et al. 2021). The MBF framework also incorporates State Machines or Behavior Trees for truly customized navigation. However, these concepts are outside the scope of the current work and will be addressed in a future report. Presently, MBF is fully compatible with ROS Noetic and backward compatible with Melodic. The full MBF launch file is provided in Appendix L. The structure of the launch file is similar to the original move_base (Glaspell et al. 2020). In regard to parameter, we pass our odom topic from robot localization and our map topic from RTAB-Map. We also use the move_base legacy relay and pass values to the base global planner and base local planner. The individual rosparam files that are passed to MBF will be addressed in the following subsections.

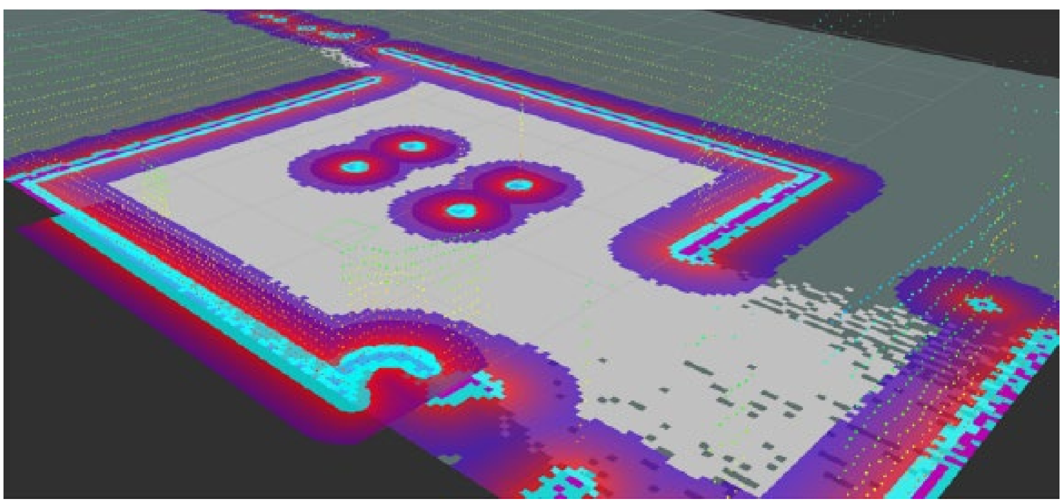
3.4.1 Common Costmap

The common costmap parameters get passed to both the global and local name spaces—that is why it appears twice in the MBF launch file. The common costmap sets the UGV's footprint and any additional padding. In our particular case, we kept this value small to ensure the robot could navigate doorways and confined tunnels. The common costmap is also responsible for the obstacles and inflation layers.

For our obstacle layer we use the spatio-temporal voxel layer (STVL). Ray tracing in 2D to remove dynamic obstacles is quite effective. However, the computational costs to do this in 3D are cost prohibitive. Rather than ray-trace, STVL temporally decays the voxels related to dynamic obstacles. This methodology uses significantly less processing power than the existing voxel layer plugin (Macenski, Tsai, and Feinberg 2020). In our setup, we have `lidar_mark`, `lidar_clear`, `rgbd1_mark`, `rgbd1_clear`, `rgbd2_mark`, and `rgbd2_clear` setup as the `observation_sources` for STVL. We pass the topic `/velodyne` points to both `lidar_mark` and `lidar_clear`. For `rgbd1_mark` and `rgbd1_clear`, we pass the topic `/toposens f/ts` cloud. This comes from our front 3D ultrasonic sensor. Similarly, we pass the topic `/toposens r/ts cloud`, from our rear ultrasonic sensor, to `rgbd2_mark rgbd2_clear`. There are a number of other parameters related to STVL, and the author's description is included in Appendix M. In short, the obstacle layer takes the data from the lidar and ultrasonic sensors and marks a costmap with the location of the potential barriers.

The inflation layer takes the data from the obstacle layer and creates an inflation buffer around it. We have the inflation radius set to 18 cm. This ensures that the robot can navigate through the doors and tunnels. However, in a larger conduit, this value could be increased to keep the UGV in the middle of the tunnel while it explores. Also, the cost scaling factor is a parabolic curve that transitions from lethal to nonlethal obstacles. A high value like 10 keeps the lethal cost close to the actual obstacle, whereas a lower value like 3 generates a more gradual curve. An example of STVL identifying obstacles with the inflation layer applied is shown in Figure 7.

Figure 7. Detection and inflation of obstacles using spatio-temporal voxel layer (STVL).

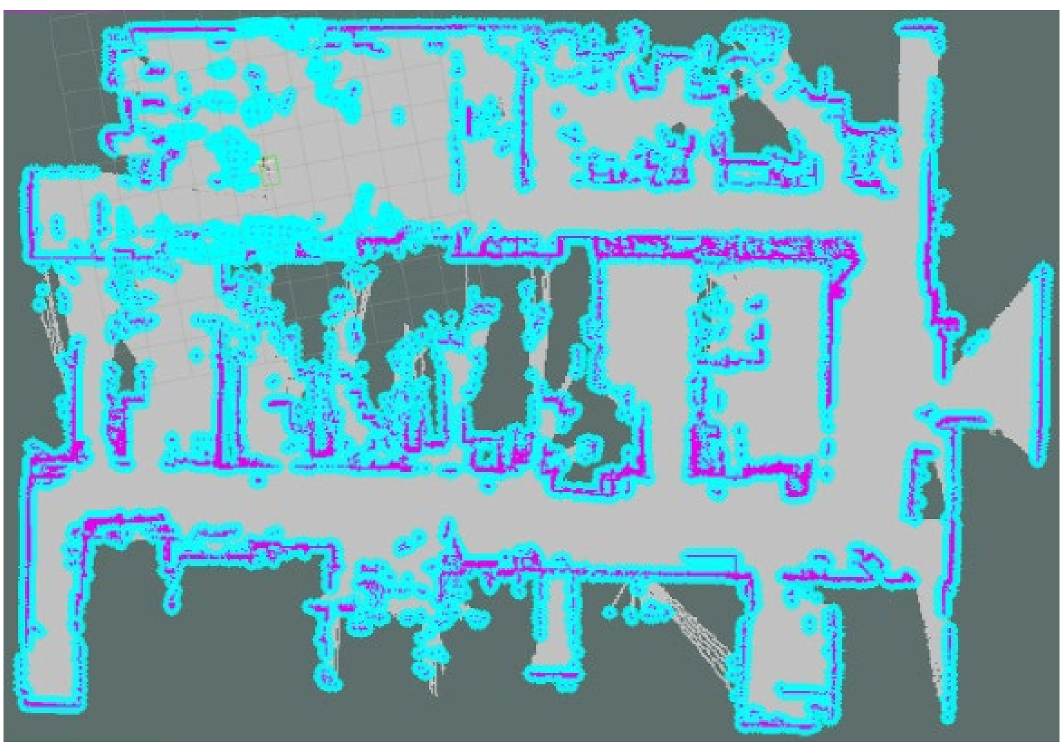


Finally, we pass the parameters for the global path planner in the common costmap. This is more for convenience. Alternatively, a separate YAML file could be generated for the global planner. There are typically three choices for the global path planner: carrot planner, navfn, and global planner. The carrot planner is the most simplistic of the three, and global planner seems to be the most robust. In fact, global planner has hooks for navfn built in. The parameters listed match the default parameters except for outline map, which we set to false. This is because we sometimes use a rolling global costmap when testing. The global path planner uses the global costmap, explained in the next section, to plan its route.

3.4.2 Global Costmap

The full global costmap YAML file is provided in Appendix N. We pass three plugins to MBF. The first plugin is the static layer that references `rtabmap_ros`. This pulls in the map from our SLAM module into the global cost map. Note that the global frame is set to the map layer. This is the same frame as our map. Next, we have the obstacle layer. This pulls in the spatio temporal voxel layer from the common costmap YAML file and marks obstacles on the global costmap. Finally we have the inflation layer, which was also outlined in the common costmap YAML file, and inflates both the static layer and obstacle layer. The global planner uses the global cost map generated by these parameters for path planning. An example of a global costmap with inflated obstacles is shown in Figure 8. The primary function of the global costmap is for path planning.

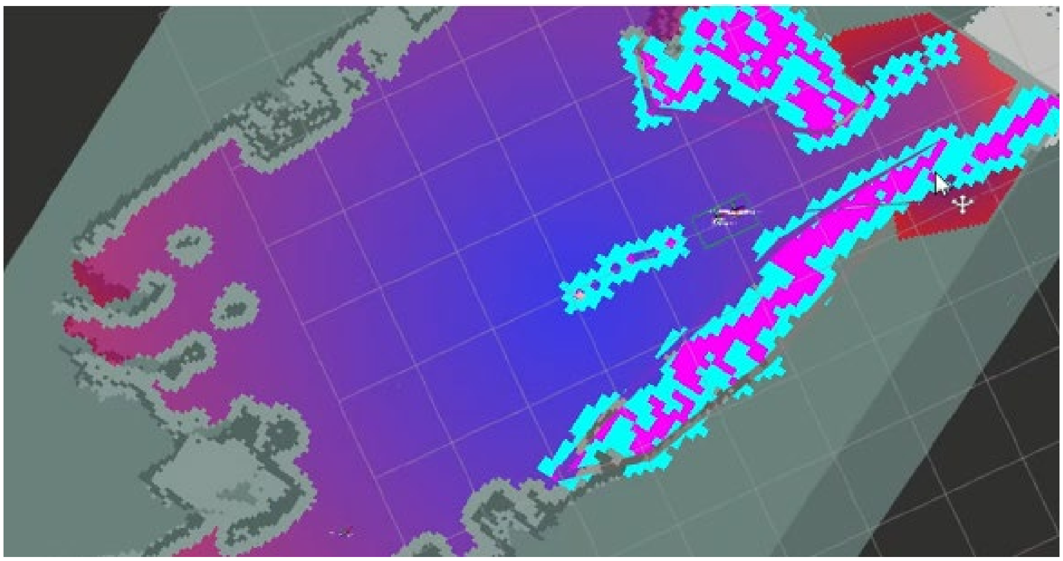
Figure 8. Inflated global costmap.



3.4.3 Local Costmap

The local costmap stacks on top of the global costmap. It is typically smaller than the global costmap and published at a faster rate than the global costmap set by the parameter update frequency. The global frame for the local costmap is set to our odom frame. Also, we set the parameter `rolling_window` to true since the local costmap moves with the robot. We only pass two plugins to the local costmap, specifically the obstacle layer and the inflation layer. An example of a local costmap with inflated obstacles is shown in Figure 9. The full local costmap YAML file is provided in Appendix O. The purpose of the local costmap is to rapidly identify obstacles that may hamper the UGV's movement.

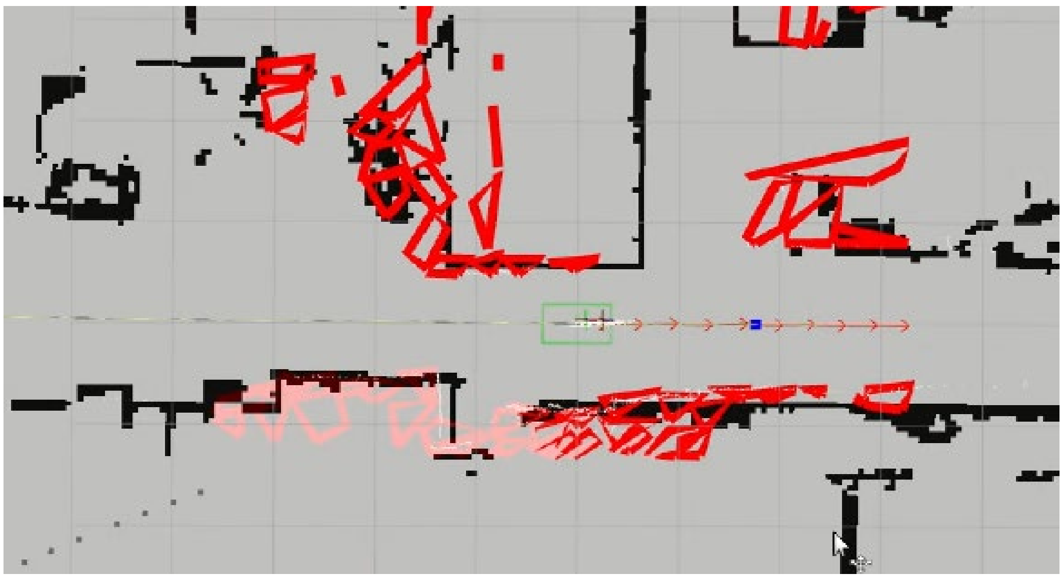
Figure 9. Inflated local costmap.



3.4.4 TEB Planner

Similar to the global planner, there are options for choosing a local path planner. Specifically they are base local, dwa, eband, and Timed Elastic Band (TEB) local planners. We prefer the TEB local planner since it can adapt for dynamic obstacles and uses costmap converter to generate primitive polygons around the costmap (Rosmann, Hoffmann, and Bertram 2017). This saves computational processing since we do not have to ray-trace to each and every point. The function of the local path planner is to send velocity commands to the UGV. As a result, we include parameters such as `max_vel_x`, `max_vel_x_backwards`, and `max_vel_theta` to control the UGV. Since we plan on using the robot in tunnels, we want it to back up when needed. Thus, we pass a low value to parameter `weight_kinematics_forward_drive`. For obstacles, the most important parameters are `min_obstacle_dist`, `inflation_dist`, and `include_dynamic_obstacles`. Note these distance values provided here are on par with inflation distance set in `common_costmap_yaml` file. The `include_dynamic_obstacles` parameter allows TEB to anticipate the motion of a moving obstacle and the path plan to avoid it. There are a number of costmap converter plugins that we can choose from. Our favorite is CostmapToPolygonsDBSMCCH. We have it set to create a polygon around every 20 points generated by the obstacle layer. An `examplecostmap_converter` is shown in Figure 10, and the full configuration is provided in Appendix P.

Figure 10. Costmap converter generating primitive polygons around obstacles.



3.4.5 move_base_flex Parameters

This YAML file passes the planner and controller to MBF. In our case, the planner is GlobalPlanner, and the controller is TebLocalPlannerROS. We also enable the recovery behaviors in this file. The first recovery behavior is conservative reset. This behavior clears the costmap 3 m past the robot. If that does not fix the issue, then the next recovery behavior is moveback recovery. Here the robot is instructed to back up 1 m to get its bearings. Typically, these two options are enough for the robot to localize itself on the map. If this is not sufficient, then the robot is instructed to perform `rotate_recovery`. The robot will spin in place. Finally, we have aggressive reset; this clears out the costmap completely. The full configuration is provided in Appendix Q.

The culmination of all these parameters is that the robot is capable of navigating by setting waypoints. As the robot encounters both static and dynamic obstacles in its environment, it can plan its path around them accordingly. Since the robot is leveraging SLAM, we do not need any a priori knowledge of its environment. In addition, the robot can fully operate and explore in a GPS-denied environment.

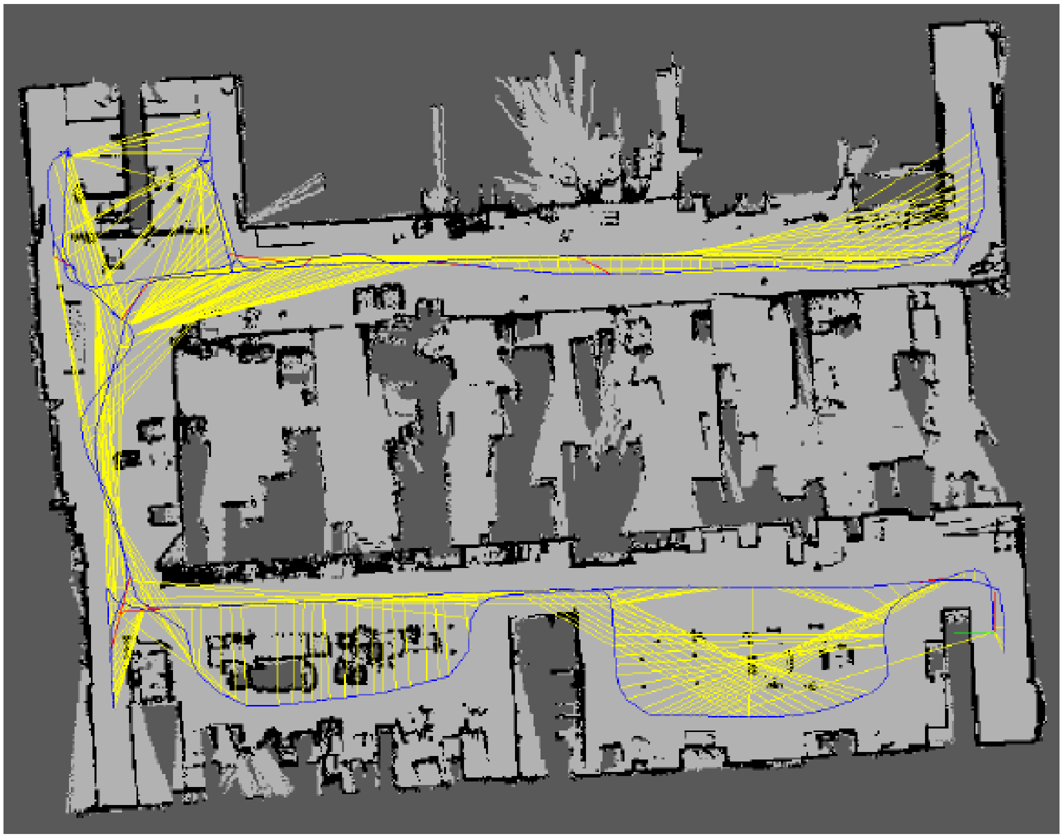
3.5 Mapping

A typical mapping session requires launching all the aforementioned files:

- `bringup.launch`
- `VLP16_points.launch`
- `microstrain.launch`
- `flir_boson_rectified_front.launch`
- `flir_boson_rectified_rear.launch`
- `ts3_f.launch`
- `ts3_r.launch`
- `ekf_flipperbot.launch`
- `transforms_flipperbot.launch`
- `cmr_lidarloop.launch`
- `rtabmap_rover_cmr.launch`
- `move_base_flex_rtabmap_stvl.launch`

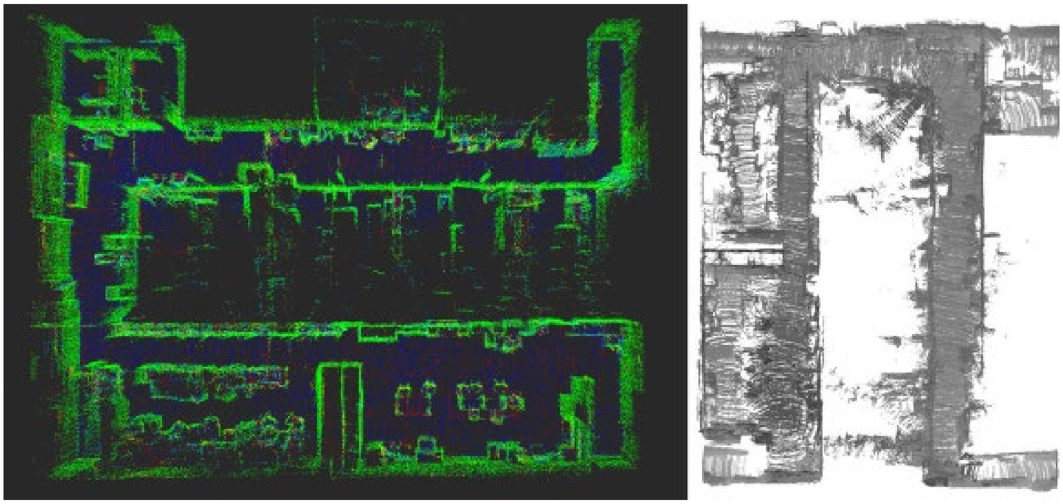
There are three products generated with this payload: a 2D occupancy grid, a 3D point cloud, and a 3D mesh. The 2D occupancy grid can be seen in Figure 11. Due to the small size of the 2D occupancy gird, this can be viewed in real time by the operator for situational awareness even if communication is degraded. The 2D occupancy grid can also be used to localize the robot if it returns to the area. Finally, the 2D occupancy grid can be shared with other robots that may be operating at the same time. The blue line indicates the path of the robot, and the yellow lines indicate loop closure.

Figure 11. A 2D occupancy grid of a building.



Typically within minutes of the robot returning to the operator, we can generate the 3D point cloud and 3D mesh. The 3D point cloud can be seen in Figure 12. The image on the left shows the point cloud in false color based on the lidar intensity. The image on the right shows the same point cloud with the pixels colored by the thermal camera.

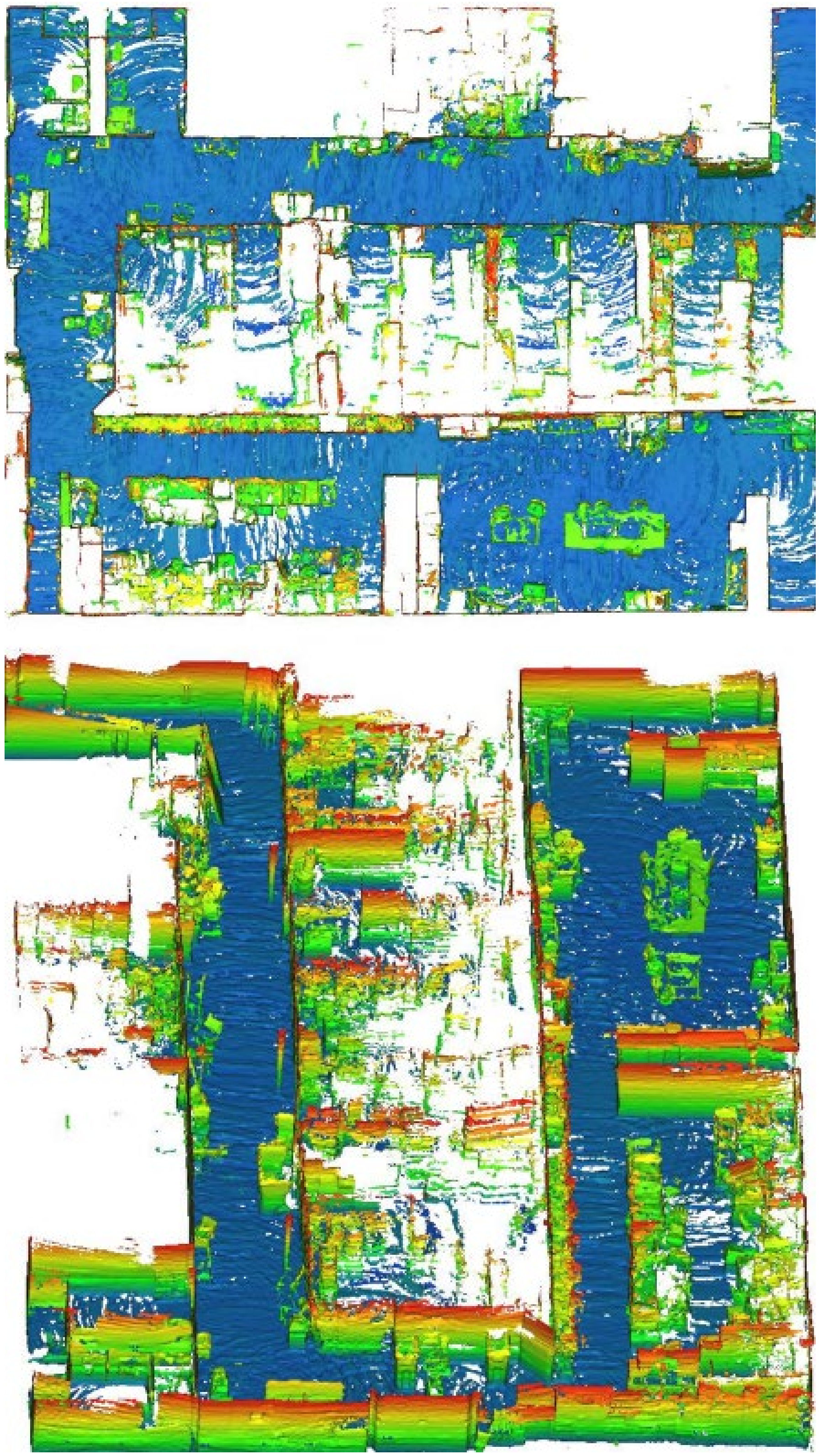
Figure 12. A 3D point cloud grid of a building. The *left* image is false color based on intensity. The *right* image is shaded using the thermal camera.



The 3D mesh can be seen in Figure 13. Note the mesh was generated by a 16-beam lidar. The density of the mesh is a result of the `point_cloud _assembler` nodelet. Compared to the file size of the point cloud, the file size of the mesh is typically smaller. The smaller files size is important since it reduces transfer times.

We chose this particular building for testing since it was also mapped with a stationary Leica survey-grade lidar. As a result, we compared the point cloud generated by the UGV, in complete darkness, with the point cloud generated with the survey-grade lidar. The results indicate that our payload had a root mean square (RMS) of 5 cm. This means each of point we collected was within 5 cm of what was measured by the survey-grade lidar. With the SWaP-C constraints of the UGV payload and the fact that it is collecting while moving, we were quite pleased with the result.

Figure 13. A 3D mesh of a building.



4 Conclusion

By leveraging thermal cameras, the robot was able to perform visual loop closure in complete darkness when tested in an indoor environment. Although not tested, the thermal cameras are expected to work well in fog and dusty environments as well (Teledyne FLIR 2020). The thermal cameras were immune to bright light sources, such as headlights, that would impair a typical RGB camera. Another advantage of using thermal cameras was that visual loop closure occurred regardless of the time of day. Specifically, an outdoor scene has different keyframes when viewed in the morning versus afternoon, due to the suns changing position. However, we observed similar keyframes when using thermal cameras regardless of the time of day or night. The inclusion of `cmr_lidarloop` also helped the robot localize its position regardless of whether the lights were on or off when tested in an indoor environment. The 3D ultrasonic sensors in combination with lidar were successful in marking obstacles as the robot navigated its environment. Finally, the low SWaP-C payload, for low-visibility environments, when compared to a survey-grade lidar resulted in an RMS of 5 cm.

References

astuff. 2021. "flir boson usb." https://github.com/astuff/flir boson usb.

CCNYRoboticsLab. 2018. "laser scan matcher." https://github.com/CCNYRoboticsLab/scan tools.

chrippa. 2018. "ds4drv." https://github.com/chrippa/ds4drv.

DARPA. 2017. "SubT Challenge." https://www.darpa.mil/program/darpa-subterranean-challenge.

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. P., S. R. Lessard, B. A. Christie, K. Jannak-Huang, N. C. Wilde, W. He, O. Ennasr, et al. 2020. "Optimized Low Size, Weight, Power and Cost (SWaP-C) Payload for Mapping Interiors and Subterranean on an Unmanned Ground Vehicle." ERDC/GRL TR-20-6. U.S. Army Engineer Research Development Center.

hrnr. 2021. "m explore." https://github.com/hrnr/m-explore.

introlab. 2022. "rtabmap_ros." https://github.com/introlab/rtabmap_ros.

Labbé, M. 2022. "rtabmap ros wiki." http://wiki.ros.org/rtabmap_ros.

Macenski, S., D. Tsai, and M. Feinberg. 2020. "Spatio-temporal Voxel Layer: A View on Robot Perception for the Dynamic World." *International Journal of Advanced Robotic Systems* 17 (2). https://doi.org/10.1177/1729881420910530.

Moore, T., and D. Stouch. 2014. "A Generalized Extended Kalman Filter Implementation for the Robot Operating System." In *Proceedings of the 13th International Conference on Intelligent Autonomous Systems* (IAS-13). New York: Springer (July).

Pütz, S., J. S. Simón, and J. Hertzberg. 2018. "Move Base Flex: A Highly Flexible Navigation Framework for Mobile Robots." In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (October).

Pütz, S., T. Wiemann, M. K. Piening, and J. Hertzberg. 2021. "Continuous Shortest Path Vector Field Navigation on 3D Triangular Meshes for Mobile Robots." In *2021 IEEE International Conference on Robotics and Automation* (*ICRA*).

Rosmann, C., F. Hoffmann, and T. Bertram. 2017. "Integrated Online Trajectory Planning and Optimization in Distinctive Topologies." *Robotics and Autonomous Systems* 17 (88).

RoverRobotics. 2021. "rr_openrover_stack." https://github.com/RoverRobotics/roverrobotics_stack.

Stüde, M. 2022. "cmr_lidarloop." https://github.com/MarvinStuede/cmr_lidarloop.

Teledyne FLIR. 2020. "Can Thermal Imaging See Through Fog and Rain?" R&D and Science. December 30. https://www.flir.com/discover/rd-science/can-thermal-imaging-see-through-fog-and-rain/.

Appendix A: bringup.launch

```
<launch>
   <arg name="openrover_node_name" default=" rr_openrover_driver"
  />
   <arg name="config_locks" default="$(find rr_open-
  rover_driver)/config/twist_mux_locks.yaml"/>
   <arg name="config_topics" default="$(find rr_open-
  rover_driver)/config/twist_mux_topics.yaml"/>


   <!-- OpenRover Driver -->
   <node pkg="rr_openrover_driver" type=" openrover_driver_node"
  name="$(arg openrover_node_name)" respawn="false" output="screen
  ">


<param name="port" value="/dev/rover" />
<param name="drive_type" value="4wd" />
        <param name="enable_timeout" type="bool" value= "true"/>
<param name="timeout" type="double" value="0.3"


  />
<param name="closed_loop_control_on" type="bool


  " value="false" />
        <!--<param name="closed_loop_control_on" type=" bool"
  value="true" />--><!-- Requires fast_data_rate
   >= 60 -->
        <param name="total_weight" type="double" value= "20.41"/>
<param name="traction_factor" value="0.45"/>
<param name="odom_covariance_0" value="0.01"/>
<param name="odom_covariance_35" value="0.03"/>

<param name="fast_data_rate" value="20.0"/>
<param name="medium_data_rate" value="2.0"/>
<param name="Kp" value="10"/>
<param name="Ki" value="30"/>
<param name="slow_data_rate" value="1.0"/>
   </node>

   <!-- Twist_mux -->
   <node pkg="twist_mux" type="twist_mux" name="
twist_mux" output="screen">
<remap from="cmd_vel_out" to="/cmd_vel/managed"
/>
```

```
      <rosparam file="$(arg config_locks)" command=" load"/>
      <rosparam file="$(arg config_topics)" command=" load"/>
  </node>

  <!-- Control input manager -->
  <node pkg="rr_control_input_manager" type=" control_input_man-
ager.py" name=" rr_control_input_manager_node" output="log" >
      <rosparam file="$(find rr_control_input_manager
)/config/input_topics.yaml" command="load"/>
  </node>
</launch>
```

Appendix B: VLP16_points.launch

```
<launch>
<!-- declare arguments with default values -->
<arg name="calibration" default="$(find velodyne_point-
 cloud)/params/VLP16db.yaml"/>
<arg name="device_ip" default="" />
<arg name="frame_id" default="velodyne" />
<arg name="manager" default="$(arg frame_id)
 _nodelet_manager" />
<arg name="max_range" default="130.0" />
<arg name="min_range" default="0.4" />
<arg name="pcap" default="" />
<arg name="port" default="2368" />
<arg name="read_fast" default="false" />
<arg name="read_once" default="false" />
<arg name="repeat_delay" default="0.0" />
<arg name="rpm" default="1200.0" />
<arg name="cut_angle" default="-0.01" />
<arg name="laserscan_ring" default="-1" />
<arg name="laserscan_resolution" default="0.007" />

<!-- start nodelet manager and driver nodelets -->
<include file="$(find velodyne_driver)/launch/ nodelet_man-
 ager.launch">
   <arg name="device_ip" value="$(arg device_ip)"/>
   <arg name="frame_id" value="$(arg frame_id)"/>
   <arg name="manager" value="$(arg manager)" />
   <arg name="model" value="VLP16"/>
   <arg name="pcap" value="$(arg pcap)"/>
   <arg name="port" value="$(arg port)"/>
   <arg name="read_fast" value="$(arg read_fast)"/>
   <arg name="read_once" value="$(arg read_once)"/>
   <arg name="repeat_delay" value="$(arg repeat_delay) "/>
   <arg name="rpm" value="$(arg rpm)"/>
   <arg name="cut_angle" value="$(arg cut_angle)"/>
</include>

<!-- start cloud nodelet -->
<include file="$(find velodyne_pointcloud)/launch/
cloud_nodelet.launch">
<arg name="calibration" value="$(arg calibration)"/
```

```
 >
   <arg name="manager" value="$(arg manager)" />
   <arg name="max_range" value="$(arg max_range)"/>
   <arg name="min_range" value="$(arg min_range)"/>

</include>

<!-- start laserscan nodelet -->
<include file="$(find velodyne_pointcloud)/launch/ la-
 serscan_nodelet.launch">
   <arg name="manager" value="$(arg manager)" />
   <arg name="ring" value="$(arg laserscan_ring)"/>
   <arg name="resolution" value="$(arg laserscan_resolution)"/>
</include>
</launch>
```

Appendix C: microstrain.launch

```
<?xml version="1.0"?>
<launch>
<!-- Standalone example launch file for 3DM-GX5-25 --
    >

<!-- Declare arguments with default values -->
<arg name="port" default="/dev/microstrain" />
<arg name="baudrate" default="115200" />
<arg name="imu_rate" default="100" />
<arg name="imu_frame_id" default="gx5_link" />
<arg name="debug" default="false" />
<arg name="diagnostics" default="true" />

<!-- For setting debug level to debug -->
<group if="$(arg debug)">
     <env name="ROSCONSOLE_CONFIG_FILE"
         value="$(find microstrain_mips)/config/ custom_roscon-
    sole.conf"/>
</group>

<!-- Microstrain sensor node -->
<node name="microstrain_mips_node" pkg="microstrain_mips"
type="microstrain_mips_node" output="screen" ns="gx5"
    >

     <param name="port" value="$(arg port)" type="str" /
    >
     <param name="baudrate" value="$(arg baudrate)" type

    ="int" />

     <param name="device_setup" value="true" type="bool"
     />
     <!-- General Settings -->
     <param name="readback_settings" value="true" type=" bool"
    />
     <param name="save_settings" value="true" type="bool " />
     <param name="auto_init" value="true" type="bool" />
          <!-- This parameter is to set wether the device
orientation uses a basic
```

```
    NED->ENU orientation swap or not. If true, the ENU reported
should
    match the label printed on the device. If false X & Y are
  swapped
    and Z is negated. -->
  <param name="frame_based_enu" value="false" type="
bool" />

  <!-- The GX5-25 is AHRS only, so need to turn off the other
messages -->
  <!-- AHRS Settings -->
  <param name="publish_imu" value="true" type="bool"
/>
  <param name="imu_rate" value="$(arg imu_rate)" type
="int" />
  <param name="imu_frame_id" value="$(arg imu_frame_id)"
type="str" />
  <!-- Declination source 1=None, 2=magnetic, 3= manual -
->
  <param name="declination_source" value="1" type=" int" />
  <param name="declination" value="0.23" type="double " />
  <!-- Filtered IMU rate is based on nav_rate since it is
tied in with the onboard Kalman Filter -->
  <!-- If you set the filtered_imu rate to be something fairly
high, make sure to lower the IMU rate

         above since it appears that the data rate can flood
  the USB. -->
  <param name="publish_filtered_imu" value="false" type="bool"
/>
  <!-- Remove gravity is only valid with the filtered IMU data. --
  >
  <param name="remove_imu_gravity" value="true" type= "bool"
/>
  <!-- Static IMU message covariance values -->
  <!-- Since internally these are std::vector we need to use
  the rosparam tags -->
  <rosparam param="imu_orientation_cov"> [0.01, 0, 0,
  0, 0.01, 0, 0, 0, 0.01]</rosparam>
  <rosparam param="imu_linear_cov"> [0.01, 0, 0, 0,
0.01, 0, 0, 0, 0.01]</rosparam>
  <rosparam param="imu_angular_cov"> [0.01, 0, 0, 0,
0.01, 0, 0, 0, 0.01]</rosparam>
<!-- GPS Settings -45 and -35 Only -->
<param name="gps_rate" value="4" type="int" />
<param name="gps_frame_id" value="navsat_link" type
 ="str" />
```

```
<!-- Filter Settings - GXx-45 Only -->
<param name="nav_rate" value="10" type="int" />
<param name="dynamics_mode" value="1" type="int" />
   <param name="odom_frame_id" value="wgs84_odom_link" type="str"
   />
   <param name="odom_child_frame_id" value="base_link" type="str"
   />
</node>

<!-- Diagnostics -->
<group if="$(arg diagnostics)">
   <!--<node pkg="rqt_topic" type="rqt_topic" name="
 rqt_topic"/>-->
   <!--<node pkg="rqt_plot" type="rqt_plot" name=" pid_setpoints"
      args="/yaw_pid_debug/Setpoint /vel_pid_debug/ Set-
 point"/>-->
<!-- Diagnostic Aggregator for robot monitor usage
 -->
   <node pkg="diagnostic_aggregator" type=" aggrega-
 tor_node" name="imu_diagnostic_aggregator">
      <rosparam command="load" file="$(find microstrain_mips)/con-
 fig/diagnostic_analyzers.yaml"
 />
</node>
</group>
</launch>
```

Appendix D: boson320.yaml


```
image_width: 640
image_height: 512 cam-
era_name: Boson320 cam-
era_matrix:
  rows: 3
  cols: 3
  data: [368.56049,  0.    , 308.06244,
              0.      ,   366.90826,  338.664    ,
              0.      ,         0.  ,    1.      ]
camera_model: plumb_bob
distortion_coefficients:
  rows: 1
  cols: 5
  data: [-0.226107, 0.031786, -0.007653, -0.002168,
     0.000000]
rectification_matrix:
       rows:     3
       cols:     3
       data:     [1.,  0.,   0.,
            0.,   1.,   0.,
            0.,   0.,   1.]
projection_matrix: rows:
  3
  cols: 4
  data: [278.98111,  0.    , 291.97537,  0.    ,
              0.      ,   299.84204,   360.00069,    0.      ,
              0.      ,         0.  ,         1.  ,    0.      ]
```

Appendix E: FLIR_boson_rectified_front.launch

```
<?xml version="1.0"?>
<launch>
<arg name="namespace" default="flir_boson_f"/>
<arg name="frame_id" default="boson_camera_f" />
<arg name="manager" default="$(arg frame_id)
    _nodelet_manager" />

   <!-- the linux file descriptor location for the camera -
    ->
<arg name="dev" default="/dev/bosonf"/>

<!-- valid values are 30.0 or 60.0 for Bosons -->
<arg name="frame_rate" default="60.0"/>

<!-- valid values are RAW16 or YUV -->
<arg name="video_mode" default="YUV"/>

<!-- valid values are TRUE or FALSE -->
<arg name="zoom_enable" default="FALSE"/>

<!-- valid values are Boson_320 or Boson_640 -->
<arg name="sensor_type" default="Boson_320"/>

<!-- location of the camera calibration file -->
   <arg name="camera_info_url" default="package://
    flir_boson_usb/example_calibrations/Boson320.yaml"/>

<group ns="$(arg namespace)">
      <!-- start nodelet manager -->
      <node pkg="nodelet" type="nodelet" name="$(arg man-
    ager)" args="manager" />

      <node pkg="nodelet" type="nodelet" name="$(arg man-
    ager)_driver" args="load flir_boson_usb/ BosonCamera
    $(arg manager)">
        <param name="frame_id" type="str" value="$(arg
    frame_id)"/>
<param name="dev" type="str" value="$(arg dev)"/>
    <param name="frame_rate" type="double" value="$( arg
frame_rate)"/>
```

```
    <param name="video_mode" type="str" value="$(arg
video_mode)"/>
    <param name="zoom_enable" type="bool" value="$( arg zoom_ena-
ble)"/>
    <param name="sensor_type" type="str" value="$(arg sen-
  sor_type)"/>
    <param name="camera_info_url" type="str" value="
$(arg camera_info_url)"/>
  </node>

  <node pkg="nodelet" type="nodelet" name="$(arg manager)_im-
age_proc" args="load image_proc/rectify
$(arg manager)">
    <remap from="image_mono" to="image_raw"/>
  </node>
</group>
</launch>
```

Appendix F: ts3_f.launch

```
<launch>
<group ns="toposens_f">
<!-- Parameters for debugging tools -->
<arg name="enable_debug" default="false" doc="Launch with debug-
ging tools such as RViz, rqt, etc.
Please specify separately." />
<arg name="launch_rviz" default="true" />
<arg name="require_rviz" default="true" doc="Shutdown if rviz is
closed, to avoid canceling via
terminal.
" />
<arg name="use_markers" default="false" />
<arg name="rviz_config" default="$(find toposens_point-
cloud)/rviz/toposens_pointcloud.rviz"
unless="$(arg use_markers)" />
<arg name="rviz_config" default="$(find toposens_mark-
ers)/rviz/toposens_markers.rviz" if="$( arg
use_markers)" />
<arg name="launch_rqt_reconfigure" default="false" />

<!-- Launch parameters -->
<arg name="frame_id" default="toposens_f" />
<arg name="target_frame" default="toposens_f" />
<arg name="lifetime_normals_vis" default="0.0" />
<arg name="port" default="/dev/ts3f" />
<arg name="scans_topic" default="ts_scans" />

<!-- Launch toposens_driver for TS3 -->
<include file="$(find toposens_driver)/launch/
ts3_driver.launch">
<arg name="frame_id" value="$(arg frame_id)" />
<arg name="port" value="$(arg port)" />
</include>

<!-- Launch pointcloud core functionality with parameters -->
<include unless="$(arg use_markers)" file="$(find toposens_point-
cloud)/launch/pointcloud.launch">
<arg name="target_frame" value="$(arg frame_id)" />
```

```
<arg name="scans_topic" value="$(arg scans_topic)"
/>
         <arg name="lifetime_normals_vis" value="$(arg
 lifetime_normals_vis)" />
</include>
  <!-- Launch Toposens Markers node -->
<include file="$(find toposens_markers)/launch/ markers.launch"
 if="$(arg use_markers)">
    <arg name="frame_id" value="$(arg frame_id)" />
  <arg name="target_frame" value="$(arg target_frame) " />
    <arg name="scans_topic" value="$(arg scans_topic)"
/>
  <arg name="sensor_mesh" value="$(find toposens_descrip-
 tion)/meshes/TS3.stl" />
  </include>


  <!-- Launch robot base frame publisher node
<include file="$(find toposens_description)/launch/ ts3_descrip-
 tion.launch" /> -->


  <!-- Launch debug tools -->
<include file="$(find toposens_bringup)/launch/debug. launch"
 if="$(arg enable_debug)">
    <arg name="launch_rviz" value="$(arg launch_rviz)"
/>
    <arg name="rviz_config" value="$(arg rviz_config)"
/>
  <arg name="require_rviz" value="$(arg require_rviz) " />
  <arg name="launch_rqt_reconfigure" value="$(arg launch_rqt_re-
 configure)" />
  </include>
</group>
</launch>
```

Appendix G: ekf_flipperbot.launch

```
<launch>
  <node pkg="robot_localization" type=" ekf_localization_node"
   name="ekf_se" clear_params=" true">
     <!--<rosparam command="load" file="$(find robot_localiza-
   tion)/params/ekf_flipperbot.yaml" />
   -->
     <rosparam command="load" file="/home/garry/github/
   MOWLES/IRL/yautja/robot_localization/param/ ekf_flipper-
   bot.yaml" />


    <!-- Placeholder for output topic remapping
<remap from="odometry/filtered" to=""/>
-->
  </node>

  <node pkg="nodelet" type="nodelet" name="pcl_manager" args="man-
     ager" output="screen" />
  <node pkg="nodelet" type="nodelet" name="voxel_grid" args="load
   pcl/VoxelGrid pcl_manager" output="screen ">


       <remap from="~input" to="/velodyne_points" />
       <rosparam> fil-
         ter_field_name: z fil-
         ter_limit_min: 0.15
         filter_limit_max: 1.0 fil-
         ter_limit_negative: False
         leaf_size: 0.01 input_frame:
         base_link output_frame:
         base_link
       </rosparam>
  </node>

  <node pkg="laser_scan_matcher" type=" laser_scan_matcher_node"
name="laser_scan_matcher_node" output="screen">
<param name="fixed_frame"   value="RLodom"/>
<param name="max_iterations" value="10"/>
<param name="use_imu"     value="true"/>
<param name="use_odom"    value="false"/>
<param name="use_cloud_input" value="true"/>
<param name="publish_tf"   value="false"/>
<param name="publish_pose"  value="false"/>
<param name="publish_pose_stamped" value="false"/>
<param name="kf_dist_linear" value="0.10"/>
<param name="kf_dist_angular" value="0.175"/>
```

```
<param name="cloud_range_min" value="0.4"/>
<param name="cloud_range_max" value="100.0"/>
<remap from="scan"      to="/scan"/>
    <remap from="cloud"      to="/voxel_grid/ output"/>
<remap from="imu"       to="/gx5/imu/data"/>
    <remap from="odom"      to="/ rr_openrover_driver/odom_en-
   coder"/>
    <!-- Publish Pose with Covariance Stamped for input to EKF -
    ->
<param name="do_compute_covariance" value="1"/>
    <param name="publish_pose_with_covariance" value=" false"/>
    <param name="publish_pose_with_covariance_stamped"
   value="true"/>
  </node>
 </launch>
```

Appendix H: ekf_flipperbot.yaml

```
frequency: 20
  silent_tf_failure: false
  sensor_timeout: 0.1
  two_d_mode: true trans-
  form_time_offset: 0.0
  transform_timeout: 0.0
  print_diagnostics: true de-
  bug: false
  debug_out_file: ~/robot_localization_debug.txt pub-
  lish_tf: true
  publish_acceleration: false permit_corrected_publication:
  false

  map_frame: map       # Defaults to "map" if unspecified
  odom_frame: RLodom      # Defaults to "odom" if unspecified
  base_link_frame: base_link # Defaults to "base_link" if
     unspecified
  world_frame: RLodom     # Defaults to the value of
     odom_frame if unspecified


  # All state-vector configs are:
  # x, y, z, roll, pitch, yaw, vx, vy, vz, vroll, vpitch, vyaw,
      ax, ay, az
# Rover Odom
#odom0: /rr_openrover_driver/odom_encoder #odom0_config:
[false, false, false,
#       false, false, false,
#       true, true, false,
#       false, false, false,
#       false, false, false] #
#odom0_queue_size: 10
#odom0_nodelay: false #odom0_differ-
ential: false #odom0_relative: false
#odom0_pose_rejection_threshold: 5
#odom0_twist_rejection_threshold: 1
# IMU
#imu0: /gx5/imu/data
#imu0_config: [false, false, false, #
false, false, true,
#       false, false, false,
#       false, false, true,
#       false, false, false] #
#imu0_nodelay: false #imu0_differen-
tial: false #imu0_relative: true
#imu0_queue_size: 10
#imu0_pose_rejection_threshold: 0.8
#imu0_twist_rejection_threshold: 0.8
```

```
#imu0_linear_acceleration_rejection_threshold: 0.8
#laser_odom
pose0: /pose_with_covariance_stamped
pose0_config: [true, true, false,
                  false, false, true,
                  false, false,
                  false, false,
                  false, false,
                  false, false,
                  false]
pose0_differential: false pose0_rela-
tive: false pose0_queue_size: 10
pose0_rejection_threshold: 2
pose0_nodelay: false
  # [ADVANCED] Some IMUs automatically remove acceleration
     due to gravity, and others don't. If yours doesn't,
     please set
  # this to true, and *make sure* your data conforms to REP-
     103, specifically, that the data is in ENU frame
     .
  imu0_remove_gravitational_acceleration: false

  # [ADVANCED] Note that if an acceleration measurement for
     the variable in question is available from one of the
  # inputs, the control term will be ignored.
  # Whether or not we use the control input during predicition.
     Defaults to false.
use_control: true
# Whether the input (assumed to be cmd_vel) is a geome-
   try_msgs/Twist or geometry_msgs/TwistStamped message.
   Defaults to
# false. stamped_con-
trol: false
# The last issued control command will be used in predic-
   tion for this period. Defaults to 0.2.
control_timeout: 0.2
# Which velocities are being controlled. Order is vx, vy,
   vz, vroll, vpitch, vyaw.
control_config: [true, false, false, false, false, true
   ]
# Places limits on how large the acceleration term will be.
     Should match your robot's kinematics.
acceleration_limits: [1.3, 0.0, 0.0, 0.0, 0.0, 3.4] # Accel-
eration and deceleration limits are not always
   the same for robots.
deceleration_limits: [1.3, 0.0, 0.0, 0.0, 0.0, 4.5] # If
your robot cannot instantaneously reach its
   acceleration limit, the permitted change can be controlled
   with these
# gains
```

```
acceleration_gains: [0.8, 0.0, 0.0, 0.0, 0.0, 0.9] # If
your robot cannot instantaneously reach its
    deceleration limit, the permitted change can be controlled
    with these

# gains

deceleration_gains: [1.0, 0.0, 0.0, 0.0, 0.0, 1.0]

process_noise_covariance:

[0.050, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0.050, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0.060, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0.030, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0.030, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0.060, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0.025, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0.025, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0.040, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0.010, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.010, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.020, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.010, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.010, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.015,
initial_estimate_covariance:
[1e-9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 1e-9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 1e-9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 1e-9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 1e-9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 1e-9, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 1e-9, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 1e-9, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 1e-9, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 1e-9, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1e-9, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1e-9, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1e-9, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1e-9, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1e-9,
```

Appendix I: cmr_lidarloop.launch

```
<?xml version="1.0"?>
<launch>
 <arg name="repo_path" default="/home/garry/github/
MOWLES/IRL/yautja" />
 <arg name="mapping" default="true" doc="If true, overwrite some
params for mapping"/>
 <arg name="cfg_file" default="$(arg repo_path)/cmr/ con-
fig/cmr_lidarloop_params.yaml"/>


     <!-- Shutdown Lidarloop -->
 <!--  <node pkg="cmr_os" type="shutdown_node" name="lidar-
loop_shutdown" required="true"/> -->


   <group ns="cmr_lidarloop">
   <env name="ROSCONSOLE_CONFIG_FILE"
     value="$(find cmr_lidarloop)/cfg/rosconsole. config"/>
     <!-- Loads main params from yaml -->
     <rosparam file="$(arg cfg_file)" command="load" />


     <group if="$(arg mapping)">
      <!-- Overwrite param to wait a duration until next de-
tection -->
      <param name="throttle_dur" type="double" value= "0.0"/>
     </group>


     <!-- Start main node -->
 <node name="lidar_loopdetection" pkg="cmr_lidarloop " type="li-
dar_loopdetection" output="screen" respawn
 ="true">
  </node>


     <!-- Start registration server -->
 <node name="lidar_registration_server" pkg=" cmr_lidarloop"
type="lidar_registration_server" output="screen" res-
pawn="true">
</node>
<!-- Start loop detector server -->
   <node name="lidar_loopdetector_server" pkg=" cmr_lidarloop"
 type="lidar_loopdetector_server.py" output="screen" res-
 pawn="true">
    </node>
</group>
</launch>
```

Appendix J: cmr_lidarloop_params.yaml

#Loop detection parameters
#Use standalone environment /cmr_lidarloop/detector to
train and test loop detectors

loop_probability_min: 0.524
#Minimum value for the loop probability, to accept a loop

R_min: 3.0
#Minimum radius in which loops are searched -> added to
this is the current uncertainty of the position (
computed with odometry)
r_max: 40.0
#Maximum range for feature computation n_verify:
1
#Verify loop -> detect at least another loop between cur-
rent id and [loop_id-n_verify,loop_id+n_verify]
#n_verify<1 -> disabled
n_max_nodes: 200
#Maximum number of nodes used for loop search -> if more
nodes are available, a random subset of size
n_max_nodes is used

alpha_thres: 0.5
#Multi session operation: alpha=nodes_WM_local_map/ nodes_WM_all
#alpha<alpha_thres -> Localization in map from different
session not yet done -> Search for loops throughout WM

n_ms_verify: 3
#Verification for multi session operation: n_ms_verify consecu-
tive loop candidates must lie within radius R_ms_verify

R_ms_verify: 3.0
#Verification for multi session operation: n_ms_verify consecu-
tive loop candidates must lie within radius R_ms_verify

n_ms_start: 3

#Multi session operation: for the first added
n_ms_start loop pairs search for loops throughout WM instead
within R_search

beta: 0.25

#Radius in which loops are searched R_search=R_min+beta

*R_odom

#Scan registration parameters
#Use standalone environment /cmr_lidarloop/src/ Registra-
tion_Test to test registration with desired scans

sky_direction: 3
#Axis pointing to the sky to identify z-axis in the current
coordinate system
#1->x pointing to the sky, -1->-x pointing to the sky, #2->y
pointing to the sky, -2->-y pointing to the sky, #3->z pointing
to the sky, -3->-z pointing to the sky, #0->z trimming disabled
z_limit: 0.3 #Z- coordinate at which the point
clouds are trimmed (to

```
    avoid  random  points  on  the  ground)
i_limit:        5
#Intensity filter: Delete points in point cloud with inten-
   sity<i_limit
r_limit:        30
#Range filter: Delete points in point cloud with range> r_limit
leafsize:       0.06
#Leaf size for voxel grid filter n_max_points:
7500
#Random downsampling after all filter steps, if number of
   points in scan is bigger than n_max_points
n_min_points:     7000
#Both filtered clouds should have more points than
   n_min_points. Otherwise loop pair is rejected, because
   registration is too challenging
min_inliers:      1000
#Minimum number of inliers to accept a transformation ( af-
   ter outlier rejection)
t_max:         3.0
#Maximum translational offset. Only if the translational
   offset calculated from the LiDAR registration is
   smaller,
#the loop is accepted and the link is sent to RTAB-Map
path_clouds:      '/home/garry/.ros/
lidarloop_clouds'
#Path where the point clouds of every registered loop pair
   are to be stored. If 'false' no clouds are saved
throttle_dur: 45.0
#When localized, wait at least this duration until next itera-
    tion. This avoids heavy CPU load.
#Will be overriden to 0.0 for mapping
#Topics
scan_topic_name:    '/velodyne_points' #Name of the
topic on which the laser scanner is
   publishing
odom_topic_name:    '/odometry/filtered' #Name of the
topic on which the odometry data is
published
```

Appendix K: rtabmap_rover_cmr.launch

```
<launch>
<arg name="gui_cfg" default="~/.ros/rtabmap_gui.ini"
 />
<arg name="launch_prefix" default=""/>
<arg name="output" default="screen"/>
<arg name="node_start_delay" default="15.0" />

<group ns="rtabmap">
  <node pkg="nodelet" type="nodelet" name=" point_cloud_assem-
 bler" args="standalone rtabmap_ros/ point_cloud_assembler" out-
 put="screen">
     <remap from="cloud" to="/velodyne_points"/>
     <remap from="odom" to="/odometry/filtered"/>
     <param name="max_clouds" type="int" value="10" />
     <param name="fixed_frame_id" type="string" value= "RLodom"
 />
  </node>

  <node pkg="nodelet" type="nodelet" name=" point-
 cloud_to_depthimage_0" args="standalone rtabmap_ros/point-
 cloud_to_depthimage" output="screen ">

       <remap from="camera_info" to="/flir_boson_f/ cam-
 era_info"/>
       <remap from="cloud" to="assembled_cloud"/>
       <remap from="image_raw" to="image_raw_0"/>
       <remap from="image" to="image_0"/>
       <param name="fixed_frame_id" type="string"
 value="RLodom"/>
       <param name="fill_holes_size" type="int" value=

"5"/>

"2"/>

<param name="decimation" type="int" value="0"/>
<param name="fill_iterations" type="int" value=

  </node>

  <node pkg="nodelet" type="nodelet" name=" rgbd_sync_0"
 args="standalone rtabmap_ros/rgbd_sync"
```

```
 output="screen">
      <remap from="rgb/image" to="/flir_boson_f/ im-
age_rect"/>
      <remap from="depth/image" to="image_raw_0"/>
      <remap from="rgb/camera_info" to="/flir_boson_f
/camera_info"/>
      <remap from="rgbd_image" to="rgbd_image0"/>
      <param name="approx_sync" value="true"/>
  </node>

  <node pkg="nodelet" type="nodelet" name=" point-
cloud_to_depthimage_1" args="standalone rtabmap_ros/point-
cloud_to_depthimage" output="screen ">


      <remap from="camera_info" to="/flir_boson_r/ camera_info"/>
      <remap from="cloud" to="assembled_cloud"/>
      <remap from="image_raw" to="image_raw_1"/>
      <remap from="image" to="image_1"/>
      <param name="fixed_frame_id" type="string" value="RLodom"/>
      <param name="fill_holes_size" type="int" value=

"5"/>

"2"/>

<param name="decimation" type="int" value="0"/>
<param name="fill_iterations" type="int" value=

  </node>

  <node pkg="nodelet" type="nodelet" name=" rgbd_sync_1"
args="standalone rtabmap_ros/rgbd_sync"
 output="screen">
      <remap from="rgb/image" to="/flir_boson_r/ im-
age_rect"/>
      <remap from="depth/image" to="image_raw_1"/>
      <remap from="rgb/camera_info" to="/flir_boson_r
/camera_info"/>
      <remap from="rgbd_image" to="rgbd_image1"/>
      <param name="approx_sync" value="true"/>
  </node>
```

```
  <node name="rtabmap" pkg="rtabmap_ros" type=" rtabmap" out-
put="screen" args="--delete_db_on_start"
  launch-prefix="bash -c 'sleep $(arg node_start_delay);
$0 $@' ">
<param name="frame_id" type="string" value="
base_link"/>
       <param name="odom_frame_id" type="string" value
="" />
       <param name="subscribe_depth" type="bool" value
="false"/>
      <param name="subscribe_scan_cloud" type="bool"
value="false"/>
      <param name="subscribe_scan" type="bool" value= "false"/>
      <param name="subscribe_scan_descriptor" type=" bool"
value="true"/>
      <!-- subscribe to the scan descriptor provided by
cmr_lidarloop (point cloud with corresponding lidar fea-
tures)-->
      <remap from="scan_descriptor" to="/ cmr_lidarloop/scan_de-
scriptor" />
      <param name="subscribe_odom_info" type="bool"
value="false"/>
      <param name="subscribe_rgb" type="bool" value="
false"/>
      <param name="subscribe_rgbd" type="bool" value=
"true"/>



       <param name="rgbd_cameras" type="int" value="2"
/>
       <remap from="rgbd_image0" to="rgbd_image0"/>
       <remap from="rgbd_image1" to="rgbd_image1"/>

       <remap from="odom" to="/odometry/filtered"/>
       <remap from="scan_cloud" to="/velodyne_points"/
>
       <remap from="scan" to="/scan"/>
       <param name="queue_size" type="int" value="50"/
>

-->

<!-- use actionlib to send goals to move_base

<param name="use_action_for_goal" type="bool"
```

```
value="true"/>
       <remap from="move_base" to="/move_base"/>

       <!-- RTAB-Map parameters -->
       <param name="RGBD/AngularUpdate" type="string"
value="0.1"/>
<param name="RGBD/LinearUpdate" type="string"
value="0.1"/>
       <param name="RGBD/NeighborLinkRefining" type="
string" value="true"/>
       <!-- Do odometry correction with consecutive la-
ser scans -->
       <param name="RGBD/ProximityBySpace" type="
string" value="true"/>
       <!-- Local loop closure detection (using esti-
mated position) with locations in WM -->
       <param name="RGBD/ProximityByTime" type="string "
value="false"/>
       <!-- Local loop closure detection with locations
in STM -->
       <param name="RGBD/ProximityPathMaxNeighbors" type="string"
value="30"/>
       <!-- Do also proximity detection by space by
merging close scans together. -->
       <param name="RGBD/OptimizeFromGraphEnd" type="
string" value="false"/>
       <!-- Optimize graph from initial node so /map
-> /odom transform will be generated -->
       <param name="RGBD/OptimizeMaxError" type="
string" value="3"/>
       <!-- Reject any loop closure causing large errors
(>3x links covariance) in the map -->
       <param name="RGBD/LocalRadius" type="string" value="10"/>
       <!-- limit length of proximity detections -->
       <param name="Reg/Strategy" type="string" value=

"1"/>

<!-- 0=Visual, 1=ICP, 2=Visual+ICP -->
<param name="Reg/Force3DoF" type="string" value

="true"/>
       <!-- 2D SLAM -->
       <param name="Grid/FromDepth" type="string" value="false"/>
       <!-- Create 2D occupancy grid from laser scan
```

```
--> 30"/>


<param name="Mem/STMSize" type="string" value="

<!-- increased to 30 to avoid adding too many

loop closures on just seen locations -->
       <param name="Vis/MinInliers" type="string"
value="10"/>
<!-- 3D visual words correspondence distance --
>
       <param name="Kp/DetectorStrategy" type="string"

  value="10"/>
      <param name="Vis/FeatureType" type="string"
value="10"/>
      <param name="Vis/EstimationType" type="string" value="0"/>

      <!-- <param name="SuperPoint/ModelPath"
type="string" value="/home/garry/github/superpoint. pt"/>
-->
      <param name="Optimizer/Strategy" type="string" value="2"/>
      <param name="Kp/RoiRatios" type="string" value= "0.0
0.0 0.0 0.4"/>
      <param name="Grid/MaxObstacleHeight" type=" string"
value="0.51"/>
      <param name="Grid/MinGroundHeight" type="string "
value="-0.05"/>
      <param name="Grid/MaxGroundHeight" type="string "
value="0.15"/>
       <param name="Grid/3D" type="string" value="true


"/>
       <param name="Grid/RayTracing" type="string"

value="true"/>
       <param name="Grid/RangeMax" type="string" value
="10"/>
       <param name="Grid/RangeMin" type="string" value
="0.19"/>
      <param name="Grid/NormalsSegmentation" type="
string" value="false"/>
      <param name="GridGlobal/AltitudeDelta" type="
string" value="0.0"/>
```

```
      <param name="Rtabmap/DetectionRate" type=" string"
value="1"/>
      <!-- ICP parameters -->
      <param name="Icp/VoxelSize"
type="string" value="0.1"/>
      <param name="Icp/PointToPlaneK"
type="string" value="20"/>
      <param name="Icp/PointToPlaneRadius" type="string"
value="0"/>
<param name="Icp/PointToPlane"
type="string" value="false"/>
<param name="Icp/Iterations" type="string" value="10"/>
<param name="Icp/Epsilon" type="string" value="0.001"/>
<param name="Icp/MaxTranslation" type="string" value="3"/>
<param name="Icp/MaxCorrespondenceDistance" type="string"
value="0.3"/>
<param name="Icp/Strategy" type="string" value="true"/>
<param name="Icp/OutlierRatio" type="string" value="0.7"/>
<param name="Icp/CorrespondenceRatio" type="string" value="0.2"/>
<param name="OdomF2M/BundleAdjustment" type="string"
value="3"/>
<param name="Icp/RangeMax" type="string" value="80"/>
<param name="Icp/PointToPlaneGroundNormalsUp" type="string"
value="0.3"/>
     </node>
   </group>
</launch>
```

Appendix L: move_base_flex_rtabmap_stvl.launch

```
<launch>
<!-- Arguments -->
<arg name="cmd_vel_topic" default="/cmd_vel" />
<arg name="odom_topic" default="/odometry/filtered" /
>
<arg name="node_start_delay" default="20.0" />

<!-- move_base -->
<node pkg="mbf_costmap_nav" type="mbf_costmap_nav"
name="move_base_flex" output="screen" launch-prefix= "bash
-c ’sleep $(arg node_start_delay); $0 $@’ ">
  <rosparam file="/home/garry/github/MOWLES/IRL/
yautja/move_base/config/common_costmap_rear_stvl. yaml"
command="load" ns="global_costmap" />
  <rosparam file="/home/garry/github/MOWLES/IRL/
yautja/move_base/config/common_costmap_rear_stvl. yaml"
command="load" ns="local_costmap" />
  <rosparam file="/home/garry/github/MOWLES/IRL/
yautja/move_base/config/global_costmap_stvl.yaml" command="load"
/>
  <rosparam file="/home/garry/github/MOWLES/IRL/
yautja/move_base/config/local_costmap_stvl.yaml" command="load"
/>
  <rosparam          file="/home/garry/github/MOWLES/IRL/
yautja/move_base/config/teb_local_planner.yaml"     com-
mand="load" />
  <rosparam                file="/home/garry/github/MOWLES/IRL/
yautja/move_base/config/move_base_flex.yaml" command
="load" />

   <remap from="cmd_vel" to="$(arg cmd_vel_topic)"/>
   <remap from="odom" to="$(arg odom_topic)"/>
   <remap from="map" to="/rtabmap/grid_map"/>
</node>

<node name="move_base_legacy_relay" pkg=" mbf_costmap_nav"
type="move_base_legacy_relay.py">
<param name="base_global_planner" value="
GlobalPlanner" />
   <param name="base_local_planner" value=" TebLocalPlannerROS" />
</node>
</launch>
```

Appendix M: common_costmap_stvl.yaml


```
max_obstacle_height: 9999
 footprint: [[-0.34, -0.19], [-0.34, 0.19], [0.34,
    0.19], [0.34, -0.19]]
 footprint_padding: 0.03
 transform_tolerance: 0.5

inflation_layer: cost_scaling_factor:
  3.0
  inflation_radius:  0.18
   obstacle_layer:
  enabled:         true
  voxel_decay:       10
  # seconds if linear, e^n if exponential decay_model:
  0
  # 0=linear, 1=exponential, -1=persistent voxel_size:
  0.05
  # meters (TODO: size of map should be based on voxel_size
  and beam angles of the lidar) track_unknown_space:   true
  # default space is known max_ob-
  stacle_height:   0.51
  # meters (setup to match robot height) unknown_threshold:
  15
  # voxel height mark_threshold:
  0
  # voxel height update_foot-
  print_enabled: true combina-
  tion_method:   1
  # 1=max, 0=override obsta-
  cle_range:     10.0
  # meters
  origin_z:        0.0
  # meters
  publish_voxel_map:    false # de-
  fault off transform_tolerance:
  0.2
  # seconds
  mapping_mode:
  # default off, saves map not for navigation map_save_duration:
  60
  # default 60s, how often to autosave observation_sources:
  lidar_mark lidar_clear rgbd1_mark rgbd1_clear rgbd2_mark
  rgbd2_clear lidar_mark:
  data_type: PointCloud2
  topic: /velodyne_points
  marking: true clearing:
  false
```

```
    min_obstacle_height: 0.15 #
    default 0, meters max_obsta-
    cle_height: 0.51 # default
    3, meters expected_up-
    date_rate: 0.5
       # default 0, if not updating at this rate at least, re-
       move from buffer

    observation_persistence: 0.0
       # default 0, use all measurements taken during now-
     value, 0=latest

    inf_is_valid: false
    # default false, for laser scans voxel_filter:
    true
       # default off, apply voxel filter to sensor, recommend
     on

    voxel_min_points: 0
       # default 0, minimum points per voxel for voxel filter

    clear_after_reading: true
       # default false, clear the buffer after the layer gets
     readings from it
 lidar_clear: ena-
    bled: true
       #default true, can be toggled on/off with as-
     sociated service call
    data_type: PointCloud2
    topic: /velodyne_points
    marking: false clearing:
    true

    max_z: 10.5
    # default 10, meters
    min_z: 0.2

    # default 0, meters

    vertical_fov_angle: 0.523
             # default 0.7, radians. For 3D lidars it's the
  symmetric FOV about the planar axis. vertical_fov_padding: 0.05
    # 3D Lidar only. Default 0, in meters horizontal_fov_angle: 6.29
       # 3D lidar scanners like the VLP16 have 360 deg horizontal FOV.
    decay_acceleration: 5 # de-
    fault 0, 1/s^2. model_type:
    1

    # default 0, model type for frustum. 0=depth camera
     , 1=3d lidar like VLP16 or similar
    rgbd1_mark:
    data_type: PointCloud2 topic:
    /toposens_f/ts_cloud marking: true
    clearing: false min_obstacle_height:
    -9999 # default 0, meters max_obsta-
    cle_height: 9999 # default 3, meters
    expected_update_rate: 0.0
       # default 0, if not updating at this rate at least, remove from
       buffer
```

```
    observation_persistence: 0.0
      # default 0, use all measurements taken during now- value,
     0=latest
    inf_is_valid: false
    # default false, for laser scans
    voxel_filter: false
      # default off, apply voxel filter to sensor, recommend on
    voxel_min_points: 0
      # default 0, minimum points per voxel for voxel filter
    clear_after_reading: true
      # default false, clear the buffer after the layer gets readings
     from it
  rgbd1_clear: ena-
    bled: true
      #default true, can be toggled on/off with associated
     service call
    data_type: PointCloud2 topic:
    /toposens_f/ts_cloud
    marking: false
    clearing: true
    max_z: 1.5
    # default 0, meters
    min_z: 0.2
    # default 10, meters verti-
    cal_fov_angle: 0.8745
      # default 0.7, radians. For 3D lidars it's the symmetric
     FOV about the planar axis.
    horizontal_fov_angle: 1.048
      # 3D lidar scanners like the VLP16 have 360 deg horizon-
     tal FOV.
    decay_acceleration: 5.0 #
    default 0, 1/s^2.
    model_type: 0
    # default 0, model type for frustum. 0=depth camera
     , 1=3d lidar like VLP16 or similar rgbd2_mark:
    data_type: PointCloud2 topic:
    /toposens_r/ts_cloud marking: true
    clearing: false min_obsta-
    cle_height: 0.15 # default
    0, meters max_obsta-
    cle_height: 0.51 # default
    3, meters expected_up-
    date_rate: 0.0
      # default 0, if not updating at this rate at least, re-
      move from buffer
    observation_persistence: 0.0
      # default 0, use all measurements taken during now-
     value, 0=latest
    inf_is_valid: false
```

```
  # default false, for laser scans voxel_filter:
  false
     # default off, apply voxel filter to sensor, recommend
   on
  voxel_min_points: 0
     # default 0, minimum points per voxel for voxel filter
  clear_after_reading: true
     # default false, clear the buffer after the layer gets
   readings from it
rgbd2_clear: ena-
  bled: true
   #default true, can be toggled on/off with
   associated service call
  data_type: PointCloud2 topic:
  /toposens_r/ts_cloud marking: false
  clearing: true
  max_z: 1.5
  # default 0, me-
  ters min_z: 0.2
  # default 10, meters verti-
  cal_fov_angle: 0.8745
    # default 0.7, radians. For 3D lidars it's the symmetric
   FOV about the planar axis.
  horizontal_fov_angle: 1.048
    # 3D lidar scanners like the VLP16 have 360 deg horizontal
   FOV.
  decay_acceleration: 5.0 #
  default 0, 1/s^2.
  model_type: 0
  # default 0, model type for frustum. 0=depth camera
   , 1=3d lidar like VLP16 or similar
   GlobalPlanner: allow_un-
       known: true cost_fac-
       tor: 3
       neutral_cost: 50
       lethal_cost: 253
       old_navfn_behavior: false
       use_dijkstra: true use_quad-
       ratic: true use_grid_path:
       false publish_potential: true
       outline_map: false
```

Appendix N: global_costmap_stvl.yaml

```
global_costmap:
global_frame: map ro-
bot_base_frame: base_link up-
date_frequency: 1.0
publish_frequency: 0.5
plugins:
  -       {name: static_layer, type: "rtabmap_ros::
 StaticLayer"}
  -       {name: obstacle_layer, type: " spatio_tem-
 poral_voxel_layer/SpatioTemporalVoxelLayer "}
  -       {name: inflation_layer, type: "costmap_2d::
InflationLayer"}
```

Appendix O: local_costmap_stvl.yaml


```
local_costmap:
global_frame: RLodom ro-
bot_base_frame: base_link update_fre-
quency: 2.0
publish_frequency: 0.5
rolling_window: true
width: 10
height: 10
resolution: 0.05
plugins:
 -      {name: obstacle_layer, type: " spatio_tem-
 poral_voxel_layer/SpatioTemporalVoxelLayer "}
 -      {name: inflation_layer, type: "costmap_2d::
InflationLayer"}
```

Appendix P: teb_local_planner.yaml

```
TebLocalPlannerROS:
  odom_topic: /odometry/filtered
  map_frame: map
  teb_autosize: True foot-
  print_model:
    type: "polygon"
    vertices: [[-0.34, -0.19], [-0.34, 0.19], [0.34,
0.19], [0.34, -0.19]]
# surge
max_vel_x: 0.6
max_vel_x_backwards: 0.5 allow_init_with_backwards_mo-
tion: True
# sway
max_vel_y: 0.0
# yaw max_vel_theta:
3.0
# min_turning_radius: 0.12
# acceleration
acc_lim_x: 1.0
acc_lim_y: 0.0
acc_lim_theta: 1.0
dt_ref: 0.5
dt_hysteresis: 0.1 global_plan_overwrite_orientation:
True max_global_plan_lookahead_dist: 5.0
feasibility_check_no_poses: 4
no_inner_iterations: 4
no_outer_iterations: 3
max_number_classes: 2


weight_kinematics_forward_drive: 5
# GoalTolerance xy_goal_toler-
ance: 0.2
yaw_goal_tolerance: 0.2
 free_goal_vel: False
global_plan_viapoint_sep: 2.5

# Obstacles include_costmap_obstacles:
True
costmap_obstacles_behind_robot_dist: 1.0
obstacle_poses_affected: 10
min_obstacle_dist: 0.18
inflation_dist: 0.23 include_dy-
namic_obstacles: True
obstacle_association_force_inclusion_factor: 1.5
```

```
  obstacle_association_cutoff_factor: 5

  ## Costmap converter plugin costmap_converter_spin_thread:
  True costmap_converter_rate: 2 costmap_converter_plugin:
  "costmap_converter::
CostmapToPolygonsDBSMCCH" #costmap_converter_plugin: "costmap_con-
  verter::
CostmapToLinesDBSRANSAC" #costmap_converter_plugin: "costmap_con-
  verter::
CostmapToLinesDBSMCCH" #costmap_converter_plugin: "costmap_con-
  verter::
CostmapToPolygonsDBSConcaveHull" #costmap_converter_plugin: ""
  # deactivate plugin

    ## Configure plugins (namespace move_base/ TebLocalPlanner-
  ROS/PLUGINNAME)
    ## The parameters must be added for each plugin separately
    costmap_converter/CostmapToLinesDBSRANSAC: cluster_max_distance:
         0.3
         cluster_min_pts: 2
         cluster_max_pts: 20
         ransac_inlier_distance: 0.2
         ransac_min_inliers: 10
         ransac_no_iterations: 2000
         ransac_remainig_outliers: 3 ransac_convert_outlier_pts:
         True ransac_filter_remaining_outlier_pts: False con-
         vex_hull_min_pt_separation: 0.1

    costmap_converter/CostmapToPolygonsDBSMCCH: cluster_max_dis-
         tance: 0.3
  cluster_min_pts: 2
  cluster_max_pts: 20
  convex_hull_min_pt_separation: 0.1
```

Appendix Q: move_base_flex.yaml


```
planners:
- name: GlobalPlanner
   type: global_planner/GlobalPlanner
controllers:
   - name: TebLocalPlannerROS
     type: teb_local_planner/TebLocalPlannerROS
shutdown_costmaps: false control-
ler_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 recov-

ery_behaviour_enabled: true

recovery_behaviors:
   -     {name: conservative_reset, type:
    clear_costmap_recovery/ClearCostmapRecovery}
   -     {name: moveback_recovery, type: moveback_recovery/
    MoveBackRecovery}
   -     {name: rotate_recovery, type: rotate_recovery/
    RotateRecovery}
   -     {name: aggressive_reset, type: clear_cost-
    map_recovery/ClearCostmapRecovery}
moveback_recovery: control-
     ler_frequency: 10.0
     linear_vel_back  : -0.5
     step_back_length  : 1.0
     step_back_timeout : 15.0
conservative_reset: reset_dis-
     tance: 3.0

aggressive_reset:
reset_distance: 0.0
rotate_recovery:
max_rotational_vel: 4.0
min_in_place_rotational_vel: 3.0
```

Abbreviations

AI	Artificial intelligence
DARPA	Defense Advanced Research Projects Agency
EKF	Extended Kalman Filter
ENU	East, North, Up
FLIR	Forward-looking infrared
GFTT	Good Features to Track
GTSAM	Georgia Tech Smoothing and Mapping
ICP	Iterative Closest Point
IMU	Inertial measurement unit
Lidar	Light detection and ranging
MBP	move_base_flex
NED	North, East, Down
ORB	Oriented FAST and rotated BRIEF
RGB	Red-green-blue
RGB-D	Red-green-blue and depth
RMS	Root mean square
ROS	Robot operating system
RTABMap	Real-Time Appearance-Based Mapping
SLAM	Simultaneous Localization and Mapping
STVL	Spatio-temporal voxel layer
SWaP-C	Size, weight, power, and cost

TORO	Tree-based netwORk Optimizer
UKF	Unscented Kalman Filter

REPORT DOCUMENTATION PAGE

1. REPORT DATE
September 2023

2. REPORT TYPE
Final Technical Report

3. DATES COVERED

START DATE	END DATE
FY2020	FY2022

4. TITLE AND SUBTITLE
UGV SLAM Payload for Low-Visibility Environments

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT
0602146A

5d. PROJECT NUMBER
AT9

5e. TASK NUMBER
01

5f. WORK UNIT NUMBER

6. AUTHOR(S)
Osama Ennasr, Mike Paquette, 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

8. PERFORMING ORGANIZATION REPORT NUMBER
ERDC/GRL TR-23-3

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Headquarters, US Army Corps of Engineers
Washington, DC 20314-1000

10. SPONSOR/MONITOR'S ACRONYM(S)
USACE

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

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

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Herein, we explore using a low size, weight, power, and cost unmanned ground vehicle payload designed specifically for low-visibility environments. The proposed payload simultaneously localizes and maps in GPS-denied environments via waypoint navigation. This solution utilizes a diverse sensor payload that includes wheel encoders, inertial measurement unit, 3D lidar, 3D ultrasonic sensors, and thermal cameras. Furthermore, the resulting 3D point cloud was compared against a survey-grade lidar.

15. SUBJECT TERMS
Computer vision; Infrared imaging; Military robots; Military surveillance; Optical radar; Remote sensing; Ultrasonic imaging; Visual discrimination

16. SECURITY CLASSIFICATION OF:

a. REPORT	b. ABSTRACT	C. THIS PAGE
Unclassified	Unclassified	Unclassified

17. LIMITATION OF ABSTRACT
SAR

18. NUMBER OF PAGES
76

19a. NAME OF RESPONSIBLE PERSON

19b. TELEPHONE NUMBER (include area code)