Assistive Robot in Low-Visibility Conditions to Support First Responders

Final Project Report - ELEC 537

Alex Smith - as636	Mehul Goel - mg278
Steven - sc311	Sugiarto Wibowo - sw183

1. Introduction

This project focuses on the development of an assistive robot designed to support first responders in low-visibility environments. A primary challenge in deploying robots for emergency response is the harsh and unpredictable nature of disaster zones. In such scenarios, visibility is frequently compromised; for example, during fire response operations, robots relying solely on optical sensors often fail to detect objects obscured by dense smoke or fail to register transparent surfaces like glass, leading to inevitable collisions.

To address these limitations, this study presents a robot prototype that integrates sensor fusion to ensure reliable navigation. While the system utilizes an Intel RealSense depth camera to generate point clouds for mapping, this optical technology faces significant performance degradation in smoke-filled areas or against transparent obstacles. To overcome this, the system is augmented with ultrasonic sensors, allowing the robot to identify objects and avoid collisions even when optical visibility is near zero. Ultimately, this approach aims to increase rescue speed, reduce evacuation times, and significantly minimize safety risks for first responders.

2. System Design

2.1. Hardware Setup

The system consists are built on top of two different embedded platforms, including:

- Raspberry Pi

  The Raspberry Pi serves as the primary onboard computer, which is powered using a power bank with 5V. It runs ROS2 and handles sensor data acquisition from the Intel RealSense camera. It performs preprocessing, publishes sensor topics, and manages communication with the motor controller and ultrasonic module.

  The Intel RealSense camera is mounted on the front of the robot, providing dense depth maps and point cloud data used for environment mapping. The authors used the Intel RealSense D435i, which provides visual-inertial odometry to estimate the robot's motion.

- Arduino Uno

  An Arduino Uno that is powered by 9V power sources from 6 x 1.5V batteries handles all low-level hardware control. The Arduino communicates with the Raspberry Pi over serial communication to control the two DC Motors. Furthermore, an ultrasonic sensor is mounted below the camera to measure object distance and used to support the collision avoidance algorithm.

2.2. Software Architecture

The software stack is divided between two platforms: the Raspberry Pi, which handles high-level perception and sensor fusion, and the Arduino Uno, which handles low-level motor control and ultrasonic sensing. The Raspberry Pi runs ROS2 Jazzy and Intel RealSense SDK

publishes RGB, infrared, and depth images to nodes, which feed into the RTAB-Map SLAM node (/rtabmap) along with visual odometry and 3D Map. The /motor_serial_node handles serial communication with the Arduino to send WASD-based motor commands and receive ultrasonic distance readings. An IMU filter (/imu_filter_madgwick) provides stable orientation estimates. Mapping and loop-closure results are visualized through /rtabmap_viz, with transforms managed by TF listener nodes.

While the Arduino UNO handles the ultrasonic reading and feeds it to the collision avoidance algorithm, and also controls the motor when receiving commands from the Raspberry Pi. Figure 1 shows the complete connection between two different platforms.

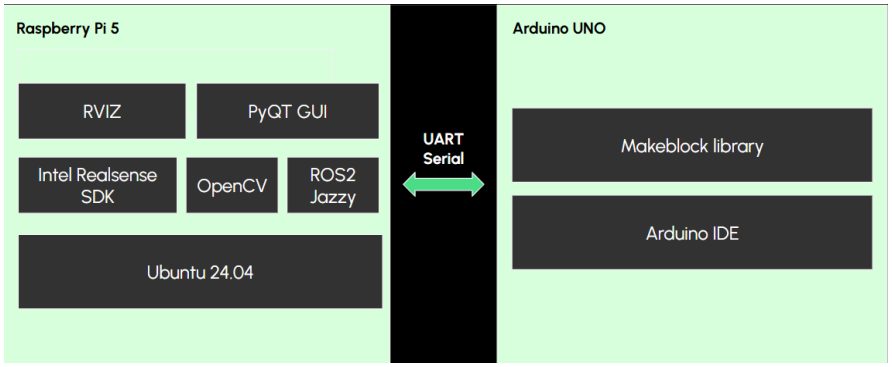


Figure 1. System Architecture

2.3. Algorithm and Implementation Details

2.3.1. Mapping with RTAB-MAP

The mapping framework utilizes RTAB-Map, a graph-based SLAM approach with a novel memory management architecture. To ensure real-time performance in large-scale environments, RTAB-Map manages the map graph across Short-Term, Working, and Long-Term Memory (STM, WM, LTM). This allows the system to archive older nodes to LTM while keeping active nodes in WM for loop closure detection, ensuring that processing complexity remains constant over time.

While the Intel RealSense depth camera generates the primary occupancy grid for global path planning, the system employs a layered costmap approach for navigation. Data from the ultrasonic array is injected into the local costmap layer. This sensor fusion strategy allows the robot to maintain a consistent global map via RTAB-Map while simultaneously reacting to immediate, low-visibility hazards detected by the ultrasonic sensors that the optical system may miss.

2.3.2. Collision avoidance

The collision avoidance works on Arduino UNO to support the robust mapping. If the measured distance is less than or equal to 15 cm, then the motor will be disabled from moving forward and send the serial command ‘Obj1’ to the Raspberry Pi to disable the GUI Control. Otherwise, if the distance is more than 15 cm, then the Arduino will process the commands, including ‘W’ for moving forward, ‘S’ for moving backward, ‘A’ for turning left, and ‘D’ for turning right. It will repeat until the system goes off.

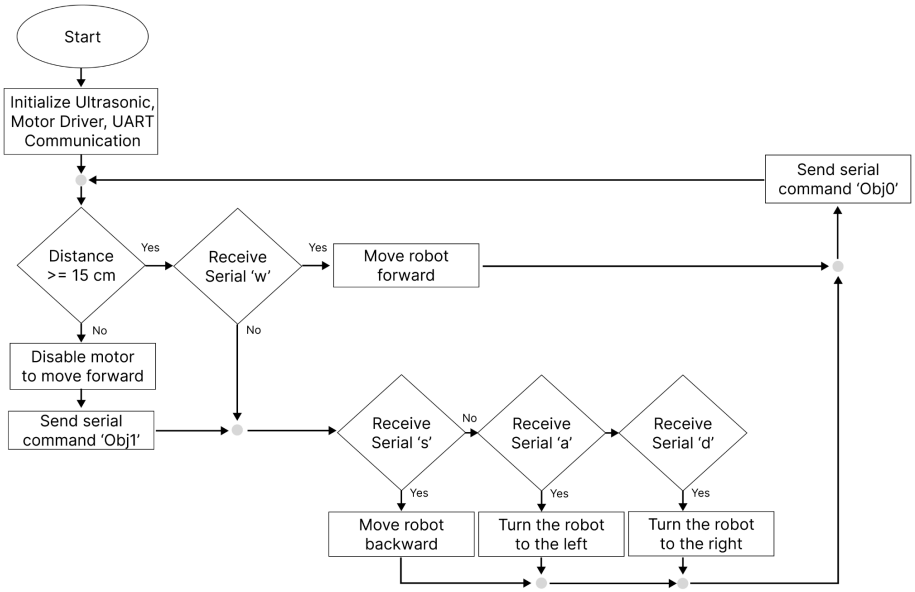
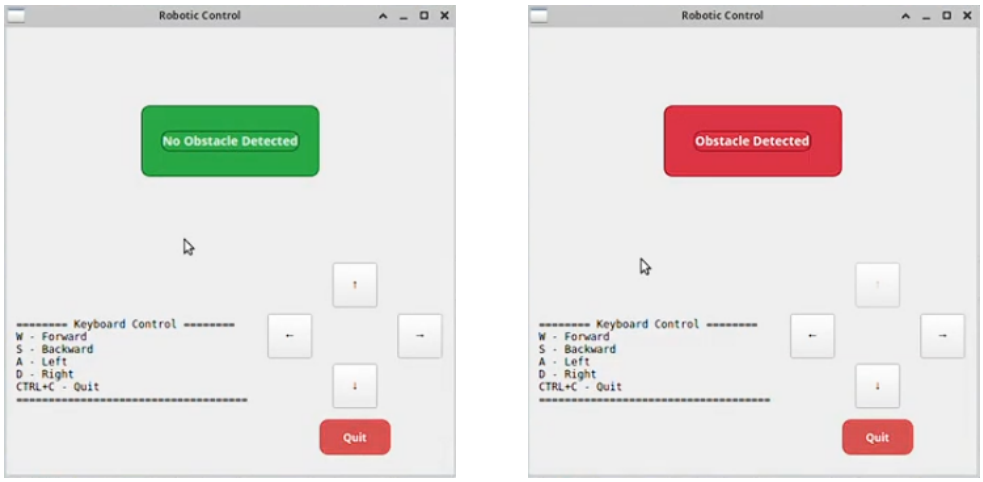


Figure 2. Collision avoidance algorithm on Arduino

2.3.3. Graphical User Interface (GUI) Control

The robot movement is controlled by using a GUI that publishes node messages and serial data to the Arduino. The published message included 'W, A, S, D'. Moreover, when an obstacle is detected less than or equal to 15 cm, the indicator of the object will turn red, and the forward button will be disabled. Otherwise, the indicator will be green.



(i) (ii)

Figure 3. GUI for controlling the robot (i) No obstacle detected (ii) Obstacle detected

3. Results and Evaluation

3.1. Experimental Setup

Experiments were conducted in an indoor environment using a small test enclosure. Part of the enclosure was covered on top to create a dark section for testing low-light conditions. Various objects, including cardboard boxes, 3D-printed objects, and transparent glass cups, were

placed randomly inside the enclosure to evaluate how well each sensing modality detected different surface types. The robot mapped the environment using each of the sensing modalities of the Intel RealSense camera: stereo, infrared, and combined Stereo-Infrared. All were combined with the ultrasonic sensor to provide collision avoidance.

To avoid human bias, we controlled the robot without direct line-of-sight, relying entirely on the sensor outputs visualized on the computer through RTAB-Map. This allowed us to evaluate the system's ability to support remote navigation when visibility is limited.

We also tested performance under low-visibility conditions by introducing dense artificial fog using dry ice. The fog was placed directly in front of the robot to examine whether the RealSense camera would fail and whether the ultrasonic sensor could still detect obstacles. These tests allowed us to compare sensor reliability across normal and degraded visibility conditions.

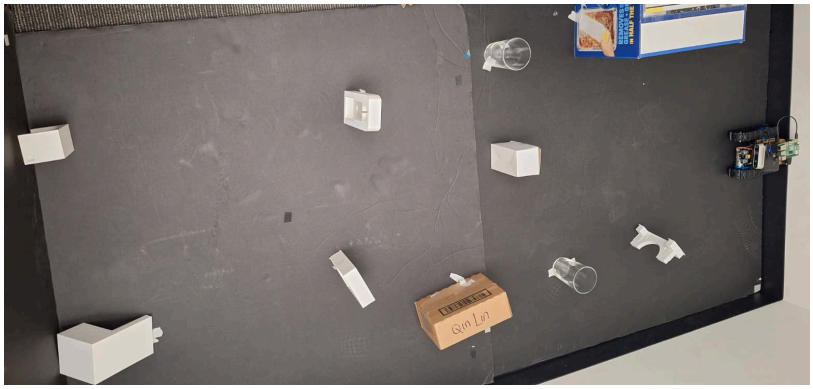


Figure 4. Simulated arena for the building

3.2. Results

The system successfully utilized RTAB-Map using multi-model sensing (stereo, infrared, and RGB-D) to generate real-time 2D and 3D maps of the environment. As shown in Figure 5, the combined visualization displays the generated 3D point cloud along with the robot's estimated trajectory. While both Stereo (top right) and RGB-D (top left) methods produced functional maps, the RGB-D sensor provided denser environmental details in clear conditions.

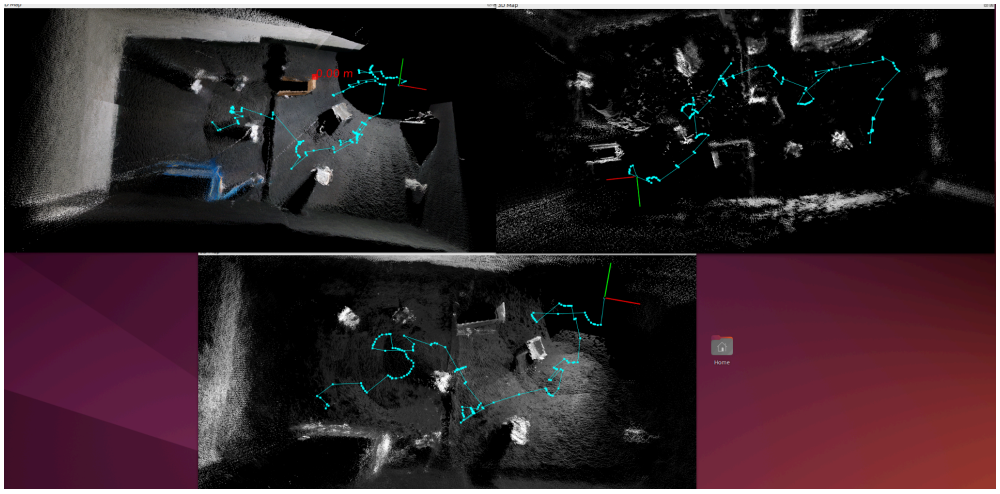


Figure 5. 3D-mapped point cloud

(i) RGB-D (ii) Stereo (iii) Infrared

Figure 6. 2D mapped environment

In simulated low-visibility conditions using dry ice, visual depth estimation proved unreliable. Figure 7 illustrates a RealSense depth map where smoke obscures vision, resulting in odometry error. Similarly, the depth camera failed to detect transparent glass, effectively "seeing through" the obstacle. However, the integration of ultrasonic sensors compensated for these visual failures, using the acoustic sensor consistently detected glass and smoke-hidden objects, triggering the collision avoidance logic when the distance dropped below 15 cm.

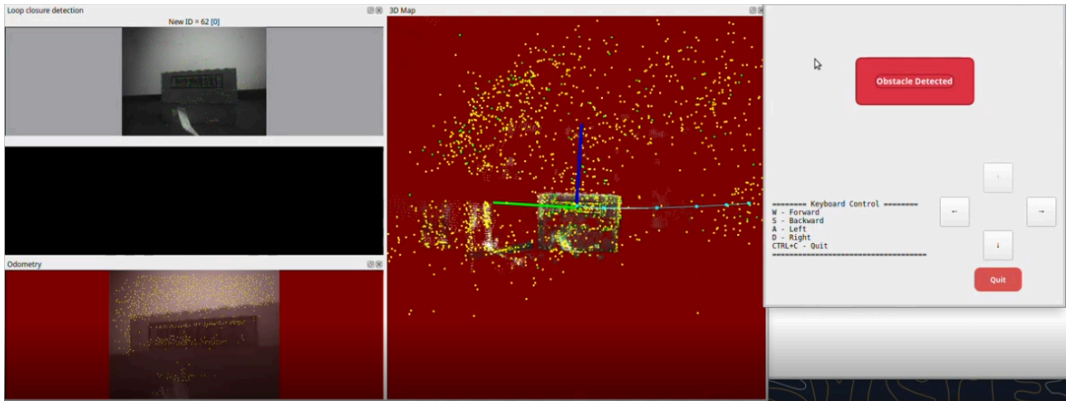


Figure 7. Odometry showing error in the fog, an ultrasonic detected object

3.3. Challenges

During the mapping experiments using a stereo depth camera, several challenges were encountered that affected the consistency and quality of the results:

- Loop-closure detection and integration issues on textureless surfaces, such as white walls or uniform boards
- Jerk and Drift: The robot experienced significant shaking during movement, and its tires frequently drifted, introducing unmodeled motion that the odometry system could not compensate for.
- Confusion in recognizing objects with printed patterns. The depth camera struggled when facing posters, small text, or images within a box. These surfaces often contain repeating patterns or small-scale features that the stereo algorithm cannot reliably match, resulting in incorrect depth estimation and loop-closure integration.

(i)

(ii)

Figure 8. Challenges on several objects mapping (i) objects continuing small texts or patterns (ii) plain surface

4. Conclusion

This project successfully developed a multi-modal sensing system to enhance situational awareness for first responders in low-visibility environments. By fusing RGB-D visual data with acoustic sensing, the system achieved reliable real-time mapping and obstacle detection even in the presence of smoke and transparent objects scenarios using ultrasonic sensors, while traditional visual SLAM typically fails. This approach demonstrated that integrating acoustic data significantly improves navigation safety and robustness compared to single-sensor solutions.

Despite a successful demonstration, there are some limitations, like the current platform experiencing mechanical jerk and odometry errors while detecting the edges, which complicate the mapping process. Furthermore, the depth camera struggled with depth estimation on plain, textureless surfaces, leading to gaps in the generated maps. Future work will address current limitations by integrating IMU data for localization and applying graph optimization (e.g., g2o, GTSAM) to correct odometry drift. Moreover, we can make it robust, both hardware and software, to work for more harsh conditions.

5. References

[1] M. Labbé and F. Michaud, “Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation,” arXiv preprint arXiv:2407.15304, Jul. 2024.

[2] Introlab, “rtabmap_launch at ros2,” rtabmap_ros Repository, GitHub. [Online]. Available: https://github.com/introlab/rtabmap_ros/tree/ros2/rtabmap_launch. [Accessed: Nov. 30, 2025].

💻Link to Github Repository
📺Link to Results Video