SLAM
PERSON MAPPING

Meet Our Team

Saad Rafiq
Mason Melead
Rose Ochoa
Colton Richard
Jimmy Abouhamzeh

SLAM

Real-time person mapping

This project uses Simultaneous Localization and Mapping (SLAM) to track and map people in real-time within a changing environment. By combining a stream of footage and machine learning, the system will identify and follow people as they move, updating the map continuously. This project can be applied to areas like indoor navigation, autonomous robots, and smart surveillance, with a focus on reliable human detection.

Project Overview

Person and Depth Detection

Generate SLAM Occupancy Grid

Flask API

Calculate and Plot Detections

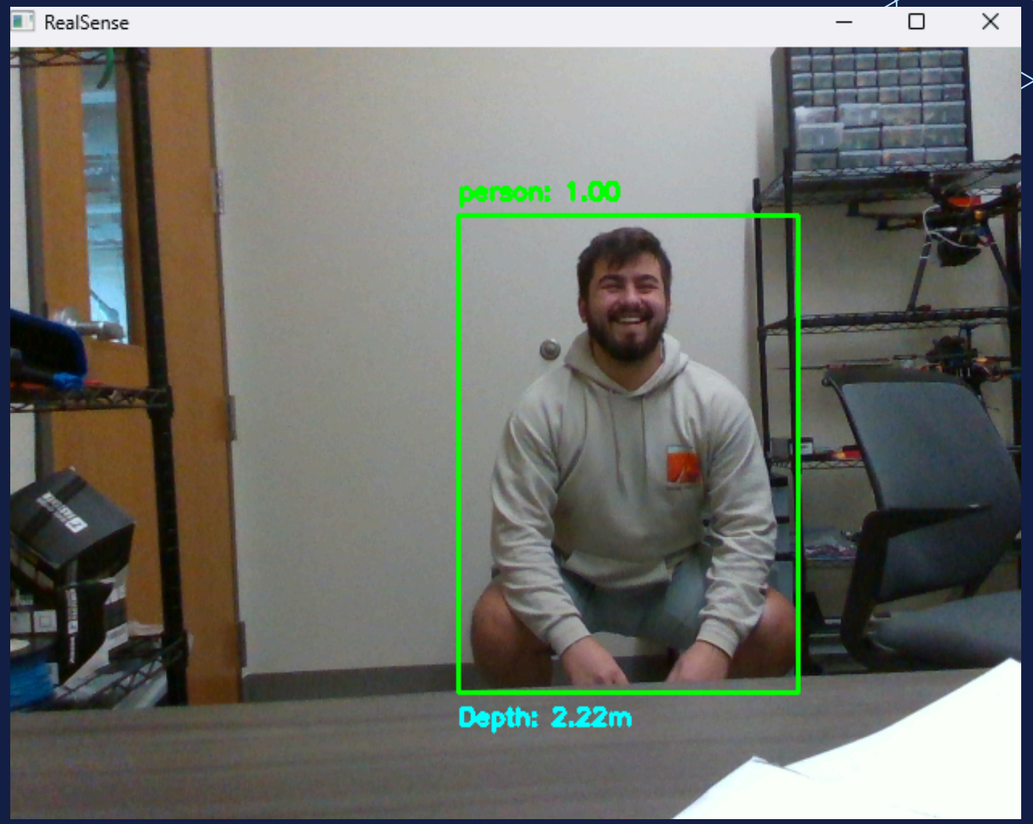
User Interface

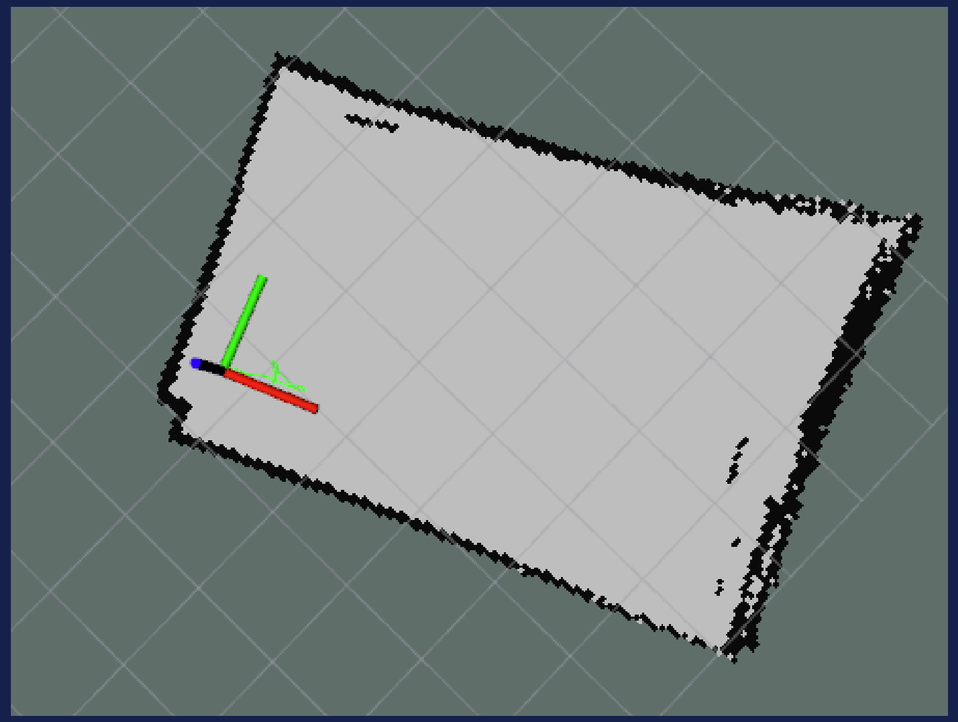
Person and Depth Detection

- Model Input
  - Utilizes MobileNet-SSD for object detection
  - Requires a prototxt and caffeemodel for configuration
- Depth Estimation
  - Leverages the Intel RealSense depth camera
- Output
  - Returns a list of dictionaries
  - [{depth: (meters), confidence: (0-1)}...]

Person and Depth Detection (cont.)

- Pipeline
  - Initialize Realsense camera pipeline for RGB and depth streams
  - Read and preprocess RGB frames for detection using OpenCV’s DNN module
  - Perform forward pass on the MobileNet-SSD model
  - Filter detections for people only (class ID: 15)
  - Measure the depth at the center of the bounding boxes for detected people





Hector-SLAM

- Software Stack
  - Robot Operating System (ROS) for easy setup
  - 2D LiDAR publisher outputs distance (m) and angle (rad)
- Outputs
  - Publishes a 1024x1024 Occupancy Grid
  - Occupancy Grid - an array which holds three values
    - 0 - free explored space
    - 100 - occupied space (obstacle)
    - -1 - unexplored space
    - Each index of the array represents .05 meters
  - Publishes the position of the LiDAR in real-world coordinates
    - Data includes (x, y, roll, pitch, yaw)

API - Roadblock

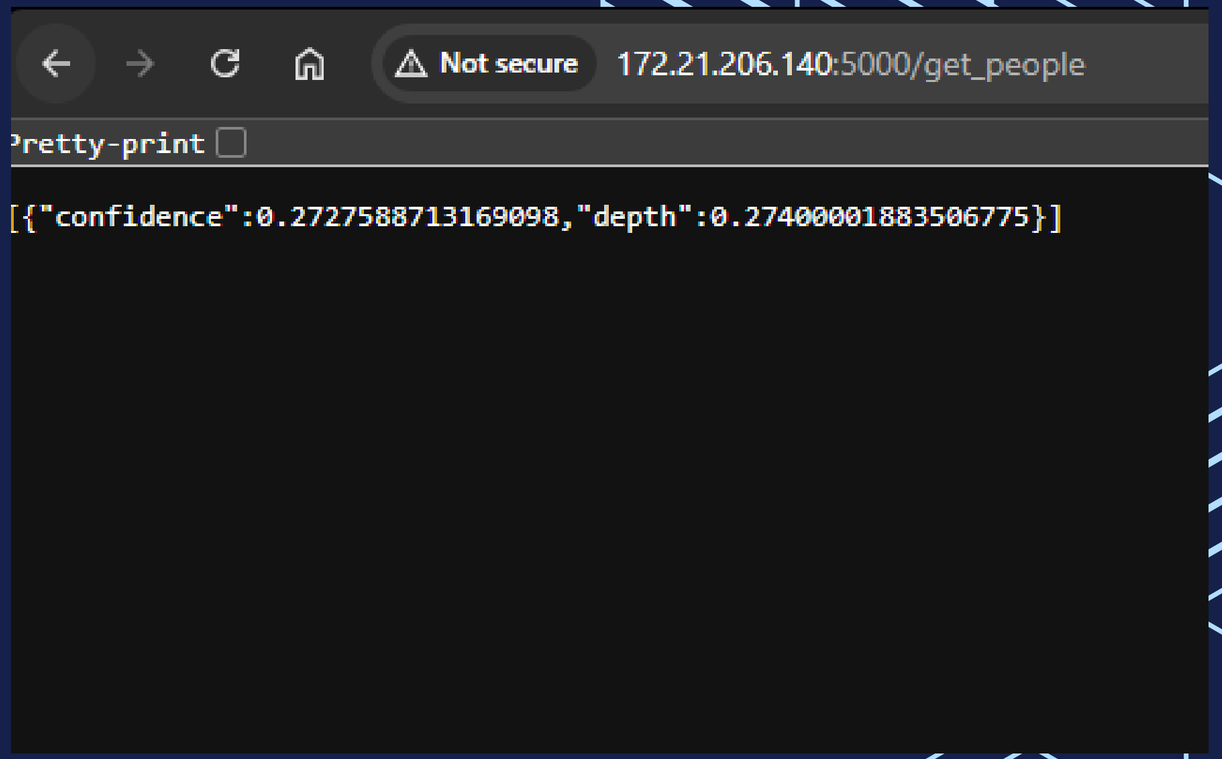
- Intel Realsense
  - Pyrealsense only works on Python versions 3.6 - 3.11
  - Requires binaries to be built for Ubuntu 18.04

- ROS
  - Melodic and Noetic distributions support Hector-SLAM
  - Noetic runs on Ubuntu 20.04
  - Melodic runs on 18.04
  - Melodic uses Python 2

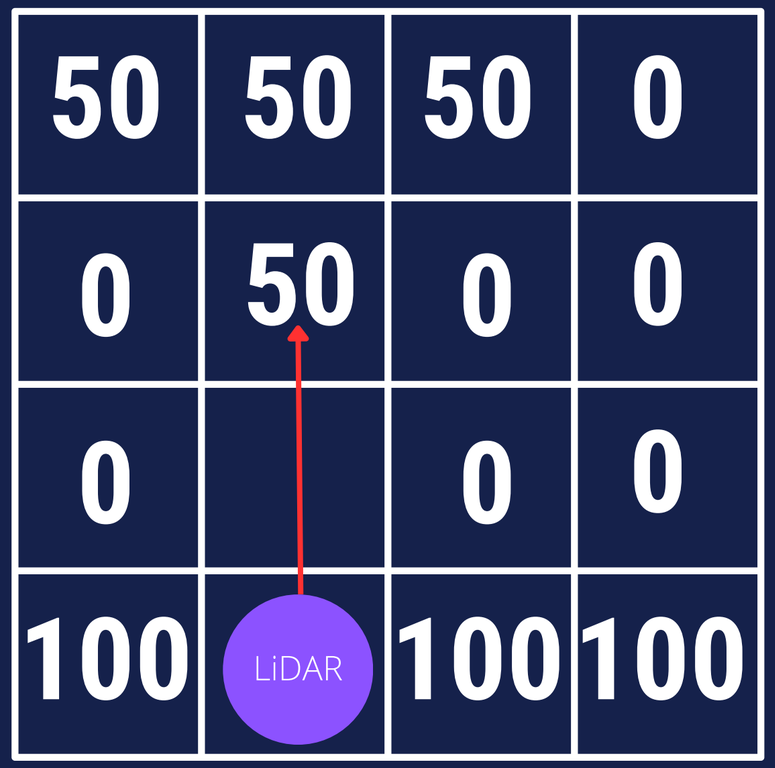
API - Roadblock (cont.)

- Solution
  - Create a Flask API to run and publish output from the detection
    - Runs on any computer with pyrealsense
  - Run SLAM on a Linux computer and grab detection data when needed

- Pros
  - Distributed computation reducing bottlenecks
  - Faster data access
  - Scalability for individual components
- Cons
  - Need a separate compute for the camera code
  - Potential latency
  - Relies on network



2m x 2m

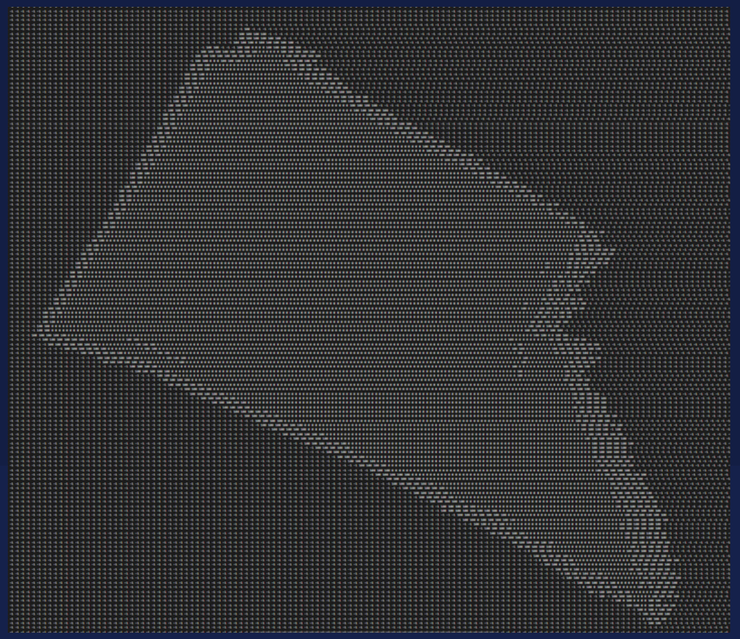


Putting it Together

- Data Available
  - Occupancy Grid
  - LiDAR’s position in the grid
  - Distance of detected persons
- Plotting
  - Fetch data from API
  - Get the most recent grid from SLAM
  - Convert depth (m) to grid coordinates
  - Calculate the detected person's position relative to the LiDAR
  - Introduce a fourth value to the grid representing a person, 50

User Interface

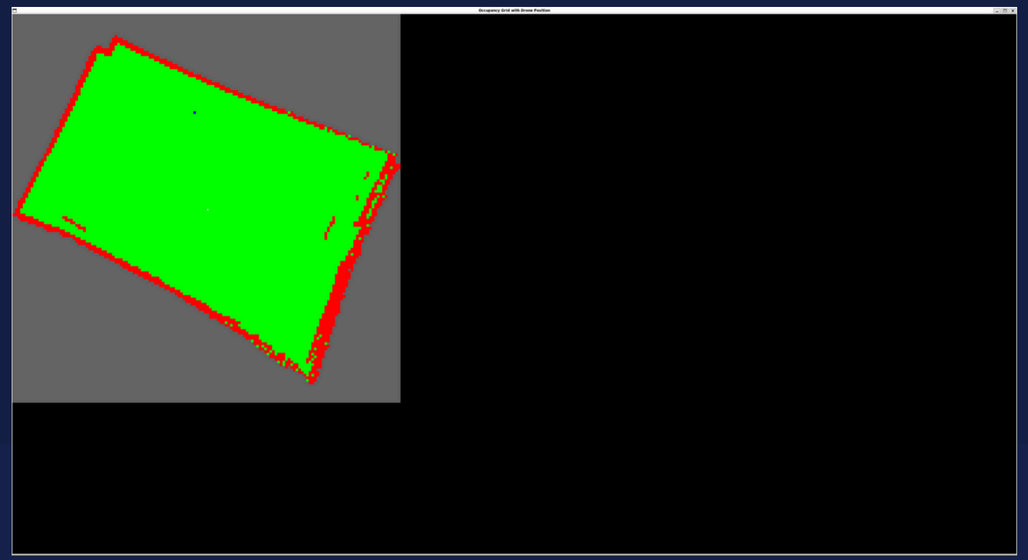
- Grid Visualization
  - Represents unexplored cells, free space, obstacles, and detected persons
    - Gray: Unexplored (-1)
    - Green: Free space (0)
    - Red: Obstacles (100)
    - Blue: Detected persons (50)
- Drone’s position
  - Displays the drone as a white circle in the grid
  - User can use live position data or move the drone around in the GUI using keyboard



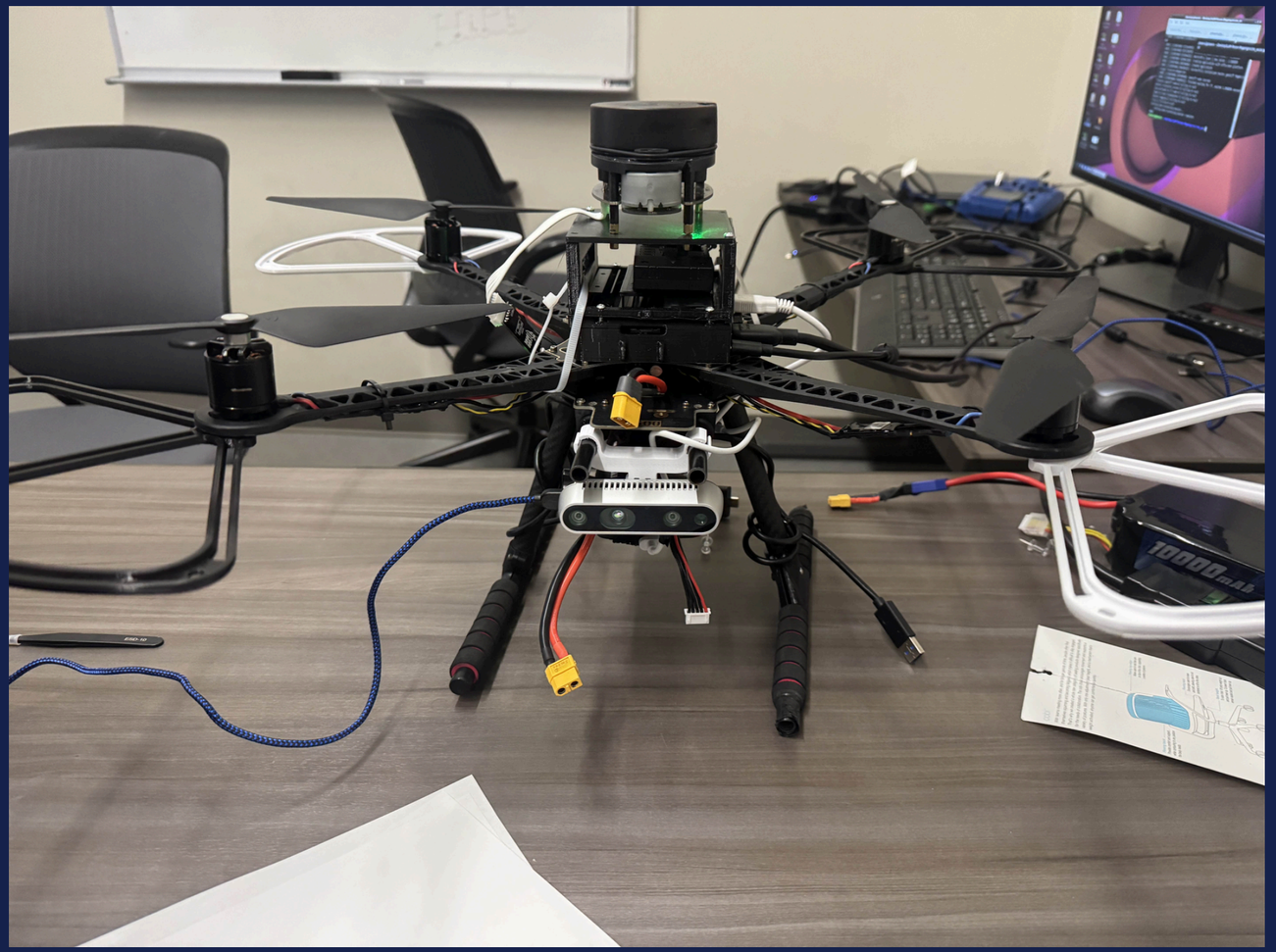
User Interface (cont.)

- Problem
  - Updating and plotting the 1024x1024 grid is computationally expensive and takes time
- Solution
  - Shrink the grid to only the relevant portion for visualization

- Grid Shrinkage
  - Use OpenCV to create a mask by isolating relevant cells (0, 100, 50)
  - Use cv2.findNonZero to identify non-zero points in mask
  - Use cv2.boundingRect to compute the smallest rectangle around these points
  - Shrinking grid from 1024x1024 to around 438x438



Results



THANK YOU