

ME 399 Final Report

2D MAPPING WITH OV2640 CAMERA

Ineza Orlyse

December 12, 2025

Northwestern University

1 Introduction

Autonomous mobile robots play an increasingly important role in areas ranging from warehouse logistics to household assistance, and at the core of these systems are reliable methods for sensing and understanding their environment. This project focuses on the design and implementation of a small car-like robot that uses an ESP32 micro-controller and two DC motors for locomotion, combined with image-based perception and Python-based processing to interpret its surroundings. Rather than relying solely on pre-defined paths or manual control, the robot uses image manipulation techniques to process visual input and incrementally build a representation of the space it moves through.

The primary goal of this project was to develop a low-cost, compact platform capable of generating a 2D map of its environment as it traverses it. The ESP32 provides onboard computation and wireless communication, while the motor system enables differential drive motion for maneuverability. Captured images are streamed to a Python environment, where image manipulation methods, such as filtering, thresholding, and feature extraction are used to detect relevant structures and update a 2D map in real time. This report describes the hardware design, software architecture, mapping pipeline, and overall system performance, and reflects on the limitations and potential improvements for future iterations of the robot.

2 Methods

2.1 Software

The core of the project was a Python program that acted as both the perception and control layer for the robot. The script periodically requested image frames from an HTTP server hosted on the ESP32, which streamed JPEG images from the onboard camera. Each frame was decoded into an OpenCV image and passed through an image-processing pipeline designed to emphasize obstacles and structural features in the environment. First, the image was resized and converted to grayscale, then a local variance-based texture map was computed to distinguish smooth regions from high-texture areas. After thresholding and inverting this map to isolate smoother surfaces, the program applied Gaussian blurring and Canny edge detection, and finally masked the edge image with the smooth-region map to obtain a filtered edge image representing prominent, obstacle-like boundaries in the scene.



Figure 1: Original and processed hallway images.

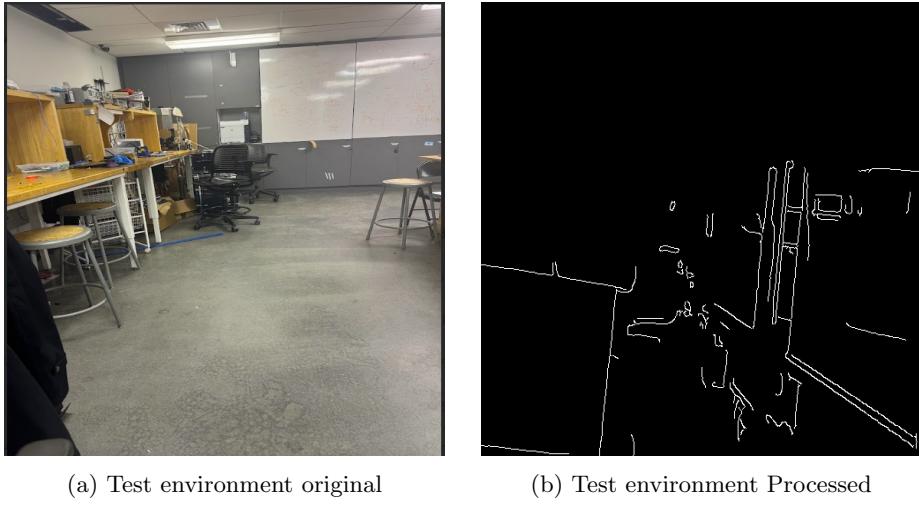


Figure 2: Original and processed test environment images.

2.2 Hardware

To convert visual information into motion decisions, the processed edge image was divided into three regions corresponding to the left, center, and right portions of the robot's field of view. The program summed the edge intensities in each region as a proxy for obstacle density and then applied a simple decision rule: if the front region contained relatively few edges, the robot was commanded to move forward; otherwise, it attempted to turn toward the side

with fewer detected edges. These discrete motion commands (forward, left turn, or right turn) were sent to the ESP32 via a PWM command interface, which adjusted the speeds of the two drive motors to execute either straight motion or in-place turns. The control loop ran at a fixed period, logging each motion decision to a CSV file. This sequence of motion primitives served as a compact representation of the robot's trajectory and formed the basis for reconstructing a 2D map of the environment in post-processing.

The hardware platform for this project was built around a Freenove ESP32-S3 WROOM development board with a pre-mounted OV2640 camera module. The ESP32-S3 served as both the main processor and the wireless interface, hosting the HTTP server used for streaming camera images and receiving motor commands. The OV2640 camera provided real-time video of the robot's forward-facing view, which was processed off-board in Python. The ESP32 board was powered from a 9 V battery, with the onboard voltage regulation supplying the required 3.3 V for the micro-controller and camera.

Locomotion was provided by two DC motors mounted at the rear of the chassis in a differential-drive configuration. Motor power was supplied directly from the 9 V battery through an H-bridge motor driver, which allowed the ESP32 to control both the direction and speed of each motor using PWM signals. The H-bridge's logic inputs were connected to the ESP32's GPIO pins, while the grounds of the ESP32, H-bridge, and battery were tied together to provide a common reference. A free-rolling caster wheel at the front of the robot supported the chassis and allowed the robot to pivot smoothly during turns.

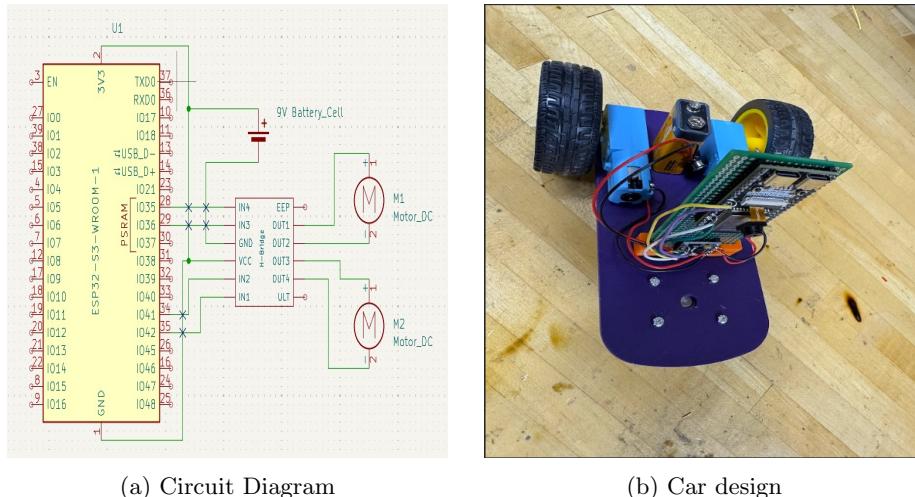


Figure 3: Component setup.

Mechanically, the robot was built on a simple 3D-printed base plate that provided mounting points for the ESP32 board, H-bridge, and battery. The two driven rear wheels were 65 mm rubber wheels from SparkFun, chosen for their traction and compatibility with the small DC motors. The front caster and rear wheels were positioned to keep the center of mass low and approximately centered between the wheels, improving stability during motion. This compact, lightweight chassis provided a stable platform for testing the vision-based navigation and mapping algorithms.

3 Results

To evaluate the system's performance, it was tested in five different environments while recording four metrics for each run: number of collisions, qualitative coverage level (high/medium/low), and total duration. In the following subsections, I present the results for each environment, along with a brief description of its layout and conditions.

3.1 Qualitative Environment Data

Simple corridor: A straight hallway with two parallel walls and no internal obstacles, providing a constrained, predictable path for the robot.



(a) Environment 1

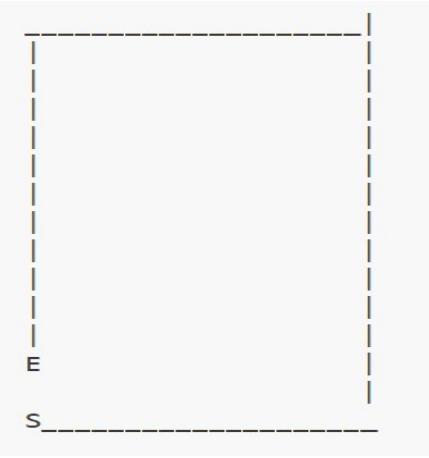


(b) Environment 1 computed map

Empty room: A roughly rectangular open space with no obstacles inside, allowing the robot to move freely and test mapping in an unobstructed environment.

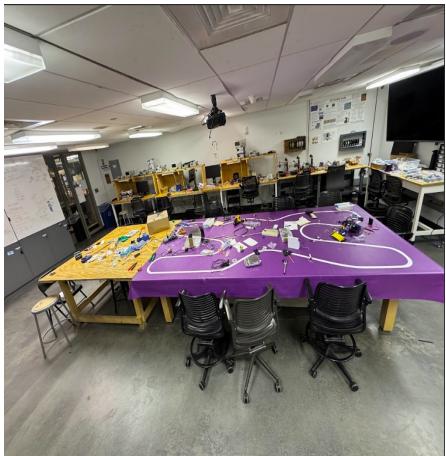


(a) Environment 2

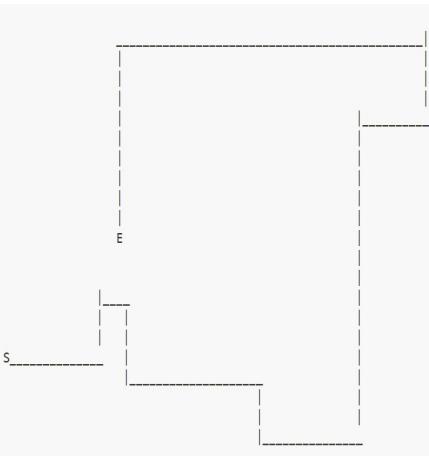


(b) Environment 2 computed map

Cluttered room: A room containing multiple obstacles such as boxes and objects placed at varying positions, creating narrow passages and irregular free space.

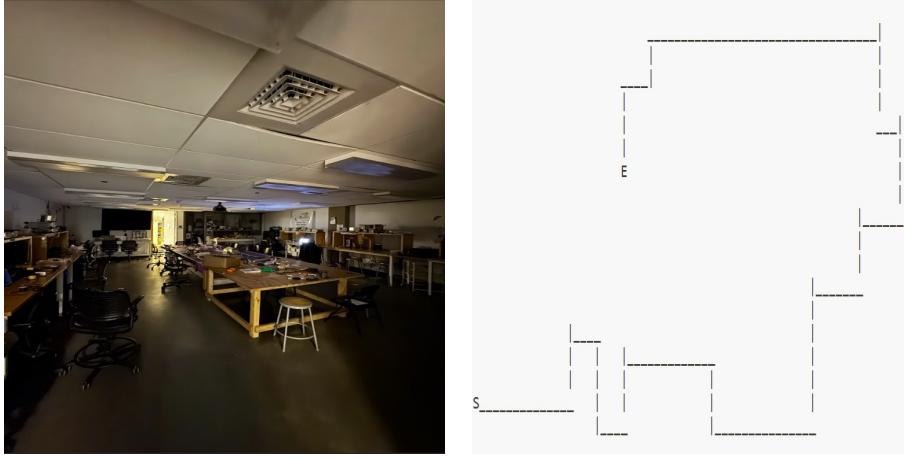


(a) Environment 3



(b) Environment 3 computed map

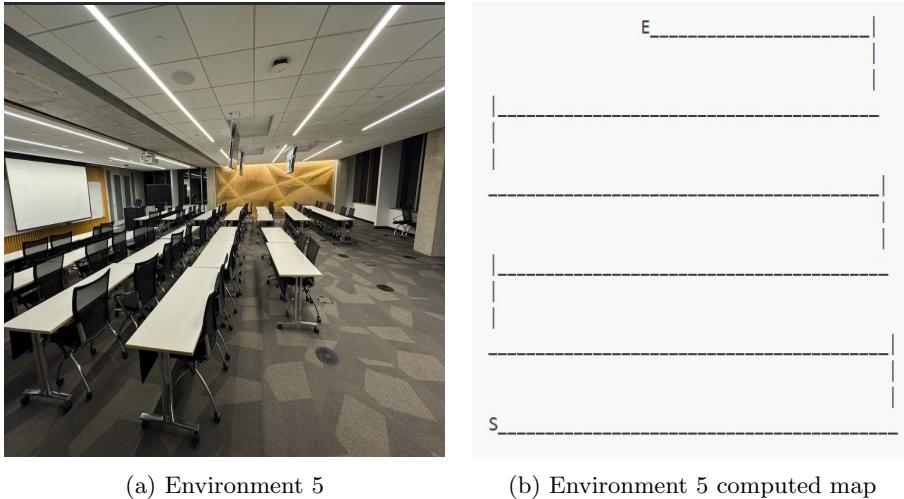
Poorly lit room: A similar layout to the empty room, but with low and uneven lighting, introducing shadows and reduced contrast in the camera images.



(a) Environment 4

(b) Environment 4 computed map

Moderately occupied room: A room with a few scattered obstacles (e.g., chairs, boxes, or furniture), representing a more typical indoor environment with both open areas and localized clutter.



(a) Environment 5

(b) Environment 5 computed map

3.2 Quantitative Test Data

The table below summarizes the quantitative performance of the system across the five test environments. For each environment, the total duration of the run, the number of collisions with obstacles or walls, and a qualitative coverage rating (high, medium, or low) indicating how much of the environment the robot was able to explore were recorded. This overview provides a compact comparison of how the system behaved under different spatial layouts and

lighting conditions.

Environment	Duration	Collisions	Coverage
1	20 s	0	High
2	37 s	0	High
3	133 s	7	Low
4	201 s	11	Low
5	120 s	2	Medium

Table 1: Quantitative data

4 Discussion

The results show that the vision-based control and mapping system performs well in simple, structured environments, but becomes less reliable as the environment gets more cluttered or visually challenging. In the simple corridor and empty room, the robot achieved high coverage with no collisions and relatively short run durations. These environments match the assumptions built into the software pipeline: the dominant edges in the processed camera images correspond clearly to walls and boundaries, and there are few competing features.

Performance degraded notably in the cluttered room and the poorly lit room. In the cluttered room, obstacles at different positions and orientations produced dense, irregular edge patterns across the image. This made it harder for the algorithm to distinguish a “safe” direction based on summed edge intensities which led to poor turning decisions. This is reflected in the higher number of collisions and lower coverage recorded in this environment. In the poorly lit room, the same pipeline was affected by low contrast and shadows. The texture-based segmentation and Canny edge detection became less reliable, occasionally missing important boundaries and amplifying noise. Since the control logic depends directly on these processed images, any visual failure immediately translated into suboptimal or unsafe actions.

Similarly, the quality of each map depended on how consistent and predictable the robot’s motion was. In the simpler environments, where the robot moved mostly straight with occasional clean turns, the accumulated path remained reasonably close to the true trajectory, producing maps that qualitatively resemble the physical layout. In more difficult environments, frequent turning, collisions, and small corrections introduced larger errors in the assumed step lengths and turning angles. Over the course of a long run, these small discrepancies accumulated into visible distortions in the maps, such as curved walls or misaligned obstacle regions.

Some of these limitations stem directly from the hardware design. The

differential-drive platform with two DC motors and a caster wheel provided basic maneuverability, but there was no odometry: the system had no encoders or other feedback to measure how far it actually moved or turned. The software therefore relied on idealized assumptions about motion, which are easily violated by battery voltage changes, floor friction, or wheel slip. Additionally, the single forward-facing OV2640 camera offered only a narrow field of view and no depth information, forcing the system to infer obstacle proximity purely from 2D edge density. This is a very lightweight sensing setup, but it is also inherently sensitive to lighting, texture, and viewpoint.

Despite these constraints, the project demonstrates that relatively simple algorithms on low-cost hardware can still produce meaningful autonomous behavior and coarse maps of real environments. The robot was able to explore and map multiple rooms without manual control, avoid complete failure modes, and generate maps that convey the general structure of each space. At the same time, the experiments clearly point toward future improvements. Adding basic odometry, incorporating an additional sensor such as an IMU or distance sensor, and refining the control logic beyond a fixed three-region threshold would likely reduce collisions and improve coverage. On the perception side, more robust or adaptive image-processing methods that handle varying lighting conditions would help address the issues observed in the poorly lit environment.

5 Conclusion

In conclusion, the purpose of this project was to design, build, and test a small vision-based mobile robot that uses an ESP32-S3, an onboard camera, and a Python control program to navigate and generate 2D maps of indoor environments. The system streamed images from the robot, processed them to extract obstacle information using a lightweight edge- and texture-based pipeline, and issued motion commands in real time. By logging each motion decision and reconstructing the trajectory, the robot produced simple 2D maps that captured the overall layout of the spaces it explored.

The results demonstrate that both the potential and the boundaries of what can be achieved with low-cost hardware and simple algorithms, and they point toward clear future improvements such as adding basic motion sensing, extra sensors, and more robust perception and control strategies.