Technical Report: AI in Robotics

Florian HEGELE (ULMP)
Olivier MARAVAL (ULMP)
Jonathan SILVA

January 13, 2026

Abstract

This report documents the development and implementation of an autonomous navigation system using the WaveShare JetBot Professional. The project focuses on using various sensors for obstacle avoidance and line following behaviors. We describe the hardware setup, the software architecture, and the specific algorithms employed to process sensor data and control the robot.

1 Introduction

In this project, we explore autonomous navigation in a confined environment by leveraging sensor data for real-time decision making.

2 Lidar Integration and Obstacle Avoidance

One of the core components of our system is the integration of a 2D Lidar sensor (RPLidar) to enable deterministic environmental perception. This section details the hardware integration, software interfacing, and the obstacle avoidance algorithm implemented.

2.1 Hardware and Software Interface

The project utilizes an RPLidar connected to the JetBot via the serial port `/dev/ttyACM1`. To interface with the hardware, we use the `rplidar` Python library.

This library handles the low-level serial communication, allowing us to retrieve scan data efficiently.

The connection is established as follows:

```
from rplidar import RPLidar
lidar = RPLidar('/dev/ttyACM1', timeout=3)
lidar.start_motor()
```

We improved robustness by wrapping the connection logic in a try-except block to handle initialization failures or timeouts. A single retry would often be enough to start the Lidar.

2.2 Data Processing

The Lidar provides a continuous stream of measurements. We process these scans to extract relevant information for navigation. The raw data consists of tuples containing quality, angle, and distance (in millimeters).

We filter this data to focus on the area directly in front of the robot. We define a "Front Sector" covering angles from $330°$ to $360°$ (right-front) and $0°$ to $30°$ (left-front). For each scan, we identify the minimum distance to an obstacle within this sector:

```
min_front_dist = float('inf')
for (_, angle, dist_mm) in scan:
    dist_m = dist_mm / 1000.0
    if dist_m <= 0: continue # Filter invalid measurements

    # Check Front Sector (+/- 30 degrees)
    if angle >= 330 or angle <= 30:
        if dist_m < min_front_dist:
            min_front_dist = dist_m
```

2.3 Control Logic

The obstacle avoidance logic should be used only as a last resort. We define a safety distance threshold (`SAFE_DISTANCE`) of 0.3 meters. The control loop operates as follows:

- If `min_front_dist` < 0.3m: The robot detects an imminent collision and executes a turn command (turning left) to avoid the obstacle.
- If the path is clear: The robot goes back to his usual logic.

This simple logic allows the JetBot to avoid obstacles without complex path planning, suitable when the ai model fails to prevent a collision.

3 Conclusion

Coming soon...

References

[1] RPLidar Python Library, https://github.com/SkoltechRobotics/rplidar