A Simple Solution to Escape Maze

Left Wall Following Algorithm

Pang (Jeff) Liu

October 27, 2024

Abstract

We present a sensor-based left wall following algorithm for autonomous robotic maze navigation. By combining front and left-side scans with a simple state machine, the robot successfully escapes arbitrary maze layouts in every test scenario. Key contributions include a two-stage obstacle detection strategy, bang–bang control for wall following, and fault-handling routines to manage dead ends and small exit rooms.

1 Introduction

Maze navigation remains a fundamental challenge in robotics, serving as a benchmark for path planning, sensor integration, and control strategies. The left wall following method is attractive due to its simplicity and near-universal success, making it suitable for resource-constrained platforms with limited computational power.

2 Background

The left wall following algorithm ensures escape by maintaining continuous contact with one maze boundary. Although it may not find the shortest path, its deterministic behavior guarantees exit in virtually all simply connected mazes. Prior work has explored variations using right-wall following and adaptive strategies for tighter corridors.

3 Sensor-Based Approach

Our implementation relies on two laser range scans:

- **Front scan ($-15°$ to $+15°$):** Detects obstacles directly ahead.
- **Left scan ($45°$ to $135°$):** Measures distance to the nearest left wall.

Figure 1 illustrates the scanning sectors used for obstacle detection and wall following.

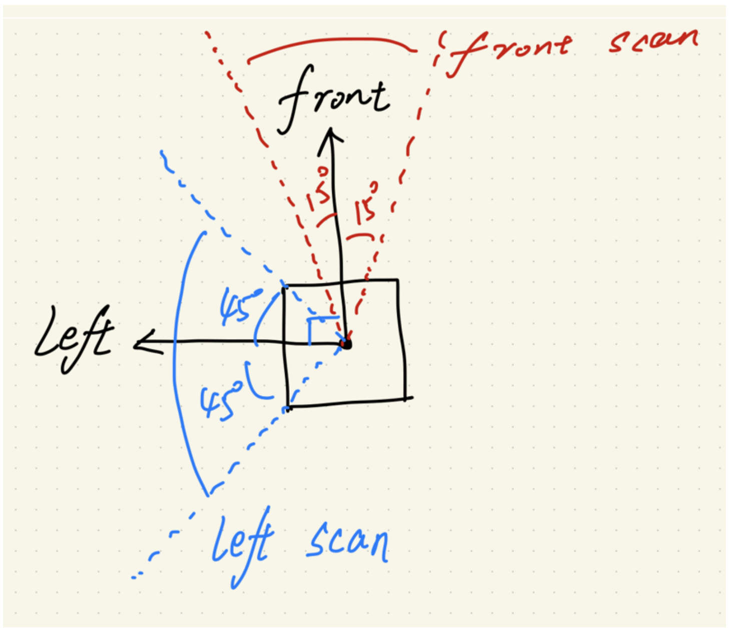

Figure 1: Front and left scan sectors.

4 Algorithm Logic

The decision process combines obstacle detection with a bang–bang control policy:

4.1 Obstacle Detection

If an obstacle is detected in the front scan:

- If the *left-turn state* is `True`, the robot executes a left turn and resets the state.
- Otherwise, it turns right and sets the *left-turn state* to `True`.

4.2 Wall-Following Policy

In the absence of a frontal obstacle, the robot evaluates its lateral position relative to three virtual lines:

1. **Dead line:** Too close to the left wall; turn right.
2. **Keep line:** Optimal following distance; move straight.
3. **Boundary line:** Too far from the wall; turn left.

Figure 2 shows example states and trajectories.

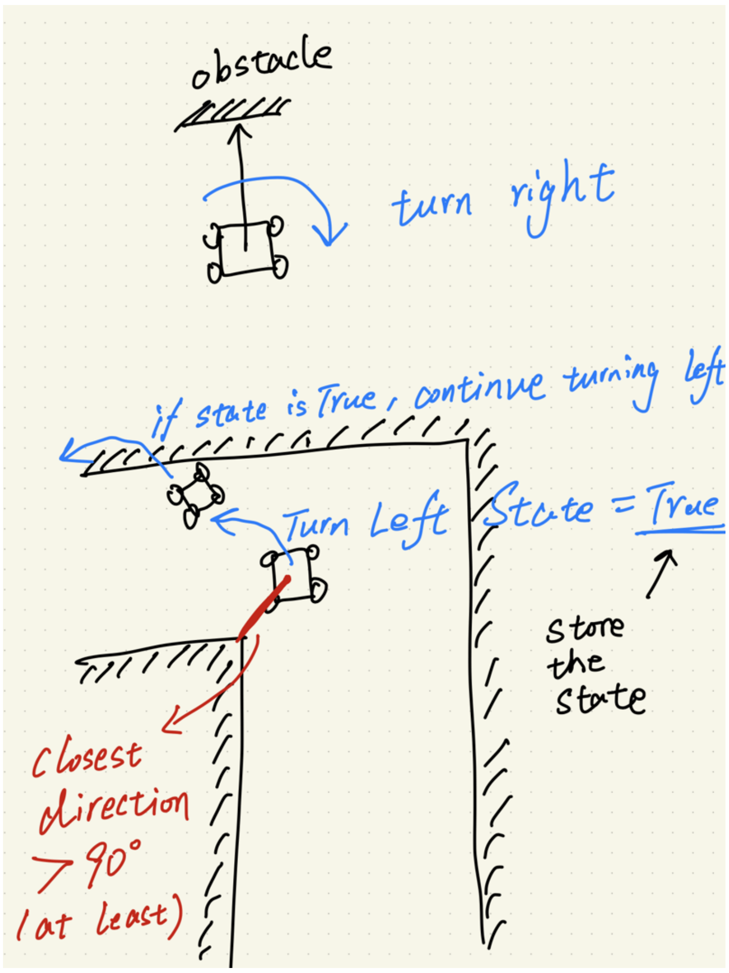
obstacle
turn right
if state is True, continue turning left
Turn Left
State = True
store the state
closest direction > 90° (at least)

Figure 2: State transitions and turning decisions.

5 Code Structure

The core algorithm is implemented in the `LeftWallFollower` class:

```
class LeftWallFollower:
    def __init__(self):
        # Initialize ROS node, publishers, and state
        self.left_turn_state = False

    def clst_dtc_and_dir(self, start_degree, end_degree):
        """Return distance and direction of the closest obstacle."""
        pass

    def scan_cb(self, msg):
        """Process incoming laser scan data."""
        pass

    def follow_left_wall(self):
        """Compute and publish velocity commands based on sensor data."""
        pass
```

6 Results

We tested the algorithm in 10 randomized maze configurations:

- **Perfect route:** 6 runs (60%), where the robot followed a consistent boundary path.
- **Imperfect route:** 4 runs (40%), involving minor backtracking but successful exit.

A sample simulation trace is shown in Figure 3.

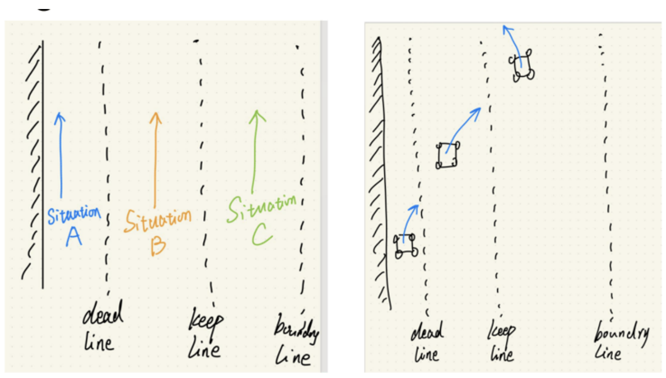

Figure 3: Simulation trajectories across test runs.

7 Fault Tolerance

Two common failure modes were addressed:

1. **Dead-end turns:** If a left turn leads into a dead end, the robot detects the blocked path and resumes right turns until a new left boundary is found.
2. **Exit room bypass:** Small alcoves near the exit may be skipped due to sensor field limitations; expanded scanning or SLAM mapping can mitigate this.

8 Conclusion

The left wall following algorithm offers a lightweight, reliable solution for maze escape. Future work includes PID control for smoother trajectories, integration with SLAM for dynamic environments, and machine learning to optimize turning decisions.