Collaborative Maze Navigation with e-puck Robots

Digital Control Laboratory Final Project

Course Instructor: Eng. AmirAli Setayeshi

Team Members:
Niusha Mohammadi Sedgh (ID: 40123078)

Spring 2025

Abstract

This project focuses on autonomous maze navigation utilizing a collaborative framework of two e-puck mobile robots programmed in Python within the Webots simulation platform. The fundamental objective is to design and evaluate an efficient digital control system alongside robust pathfinding and exploration strategies. Key operational tasks include the autonomous mapping and exploration of Zone 1 and Zone 2, visual detection of designated targets (activating an onboard LED upon identifying red tiles), smooth boundary handoff where the primary robot halts at the blue tile interface, and subsequent initiation of Zone 2 traversal by the secondary robot followed by a successful return sequence to the blue tile. Closed-loop motion stability and collision-free execution are guaranteed through empirically tuned discrete PID controllers integrated with custom graph search techniques.

Contents

1 Introduction and Problem Statement **3**

2 PID Controller and Digital Control Loops **3**
2.1 Standard PID Control Form 3
2.2 Discrete-Time Implementation 3
2.3 Gain Characterization and Tuning 3

3 Mapping and Maze Exploration Algorithms **3**
3.1 Breadth-First Search (BFS) 4
3.2 Depth-First Search (DFS) 4
3.3 Alternative Algorithms Evaluation 4

4 Algorithm Implementation Details **4**
4.1 Coordinate Transformation 4
4.2 Cell Classification and Ray Casting 4

1. Introduction and Problem Statement

Multi-robot coordination in unstructured or semi-structured environments requires real-time sensing, precise control execution, and efficient spatial mapping. In this project, two e-puck mobile platforms collaborate to systematically explore a complex maze divided into distinct functional regions (Zone 1 and Zone 2). The control architecture relies on discrete feedback loops to correct trajectory errors, while global and local pathfinding algorithms ensure continuous mapping, obstacle avoidance, and goal identification.

2. PID Controller and Digital Control Loops

To ensure stability during linear propulsion and rotational maneuvers, a Feedback PID control architecture is employed. The controller continuously evaluates the state trajectory against desired setpoints and modulates motor command outputs accordingly.

2.1. Standard PID Control Form

The continuous-time control law for the error signal $e(t)$ is expressed as:

$$u(t) = K_p\, e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \tag{1}$$

where K_p, K_i, and K_d denote the proportional, integral, and derivative gains, respectively.

2.2. Discrete-Time Implementation

Given a digital execution loop with sampling period T_s, numerical discretization (using forward Euler or Tustin transformations) yields the discrete update formula at time step k:

$$e[k] = x_{\text{setpoint}} - x[k] \tag{2}$$

$$I[k] = I[k-1] + e[k] \cdot T_s \tag{3}$$

$$D[k] = \frac{e[k] - e[k-1]}{T_s} \tag{4}$$

$$u[k] = K_p\, e[k] + K_i\, I[k] + K_d\, D[k] \tag{5}$$

2.3. Gain Characterization and Tuning

Proportional Gain (K_p): Provides direct response scaling relative to positional displacement (e_x, e_y). Elevating K_p accelerates systemic response time; however, excessive values induce oscillatory behavior and dynamic instability.

Integral Gain (K_i): Eliminates residual steady-state errors by accumulating past path deviations. High settings lead to phase lag and integral windup.

Derivative Gain (K_d): Damps overshoot and suppresses high-frequency transient variations by acting on the rate of error change. Over-amplification increases susceptibility to high-frequency sensor noise.

Rotational control utilizes a dedicated angular loop operating on the error metric $\Delta\theta = \theta_{\text{target}} - \theta_{\text{current}}$, bounded within $[-\pi, \pi]$.

3. Mapping and Maze Exploration Algorithms

The spatial mapping architecture relies on an occupancy grid structure where each discrete cell state is logged into a spatial hash table (`occupancy_grid`).

3.1. Breadth-First Search (BFS)

BFS performs level-order grid exploration utilizing a First-In, First-Out (FIFO) queue. It guarantees the absolute shortest path in unweighted planar graphs. The customized function `find_multiple_bfs_paths` isolates distinct trajectories using localized tracking structures (`visited_local`).

3.2. Depth-First Search (DFS)

DFS traverses graph structures deep along each branch before backtracking, managing nodes via a Last-In, First-Out (LIFO) stack structure. DFS proved highly effective for dense maze areas with narrow passages where complete coverage is preferred over local path minimality.

3.3. Alternative Algorithms Evaluation

- **A* Search:** Discarded due to initial map unpredictability and elevated memory storage requirements for open/closed lists prior to full environment discovery.
- **Wall-Following (Bug Algorithm):** Evaluated but rejected due to infinite loop conditions observed in symmetric, tightly spaced interior walls.

4. Algorithm Implementation Details

4.1. Coordinate Transformation

World frame coordinates are mapped to discrete grid indices using scaling transformations:

$$(i, j) = \text{world\_to\_grid}(x, y) = \left(\lfloor \frac{x - x_{\text{origin}}}{\Delta s} \rceil, \lfloor \frac{y - y_{\text{origin}}}{\Delta s} \rceil \right) \tag{6}$$

The inverse conversion grid_to_world(i, j) computes metric target centroids for navigation execution.

4.2. Cell Classification and Ray Casting

The occupancy grid tracks five structural states: *Current Position*, *Target*, *Obstacle*, *Free Space*, and *Unexplored*. Intervening cells between robot position vectors and distance sensor readings are mapped using **Bresenham's Line Algorithm**, updating occupancy states based on range-finder measurements.

References

[1] C. L. Phillips and H. T. Nagle, *Digital Control System Analysis and Design*, 4th ed. Englewood Cliffs, NJ: Prentice Hall.

[2] Cyberbotics Ltd., *Webots: Open source robot simulator*, Available: `https://www.cyberbotics.com`.

[3] OpenAI, *ChatGPT Language Model Architecture*, 2024.

[4] A. Setayeshi, *Digital Control Systems Laboratory Course Notes*, Department of Electrical Engineering, Amirkabir University of Technology.