Autonomous Mobile Robots
CDA 4621
Lab Report 5
Path Planning

Claude Watson

Objective

The objective of this lab is to explore and implement fundamental concepts of mapping and path planning in mobile robotics. Through the use of sensors such as encoders, LIDAR, and compass, this lab aims to enable a robot to represent its environment, localize within a structured grid, and calculate the shortest path to a specified goal. Data structures will be designed for maze representation, implement pathfinding algorithms to ensure the robot navigates accurately while maintaining and updating pose estimates.

Methodology

Task 1 – Path Planning

Path Planning Using Wavefront Algorithm

The robot was programmed to navigate a grid-like environment by implementing a Wavefront-based path planning algorithm to calculate the shortest path between a starting cell and a goal cell.

1. **Maze Representation**:
    - The environment was represented as a 5x5 grid with a predefined wall configuration.
    - A dictionary data structure (cellStructure) was used to store wall information for each cell in the maze. Each cell contained details about the presence or absence of walls in the four cardinal directions (North, East, South, West).
2. **Wavefront Algorithm**:
    - The goal cell was initialized with a value of 0, and all other cells were assigned infinity to indicate unvisited cells.
    - A queue-based propagation mechanism was implemented to assign incremental values to neighboring cells, representing the number of steps required to reach the goal.
    - Wall boundaries were validated using the maze representation, ensuring that path calculations avoided obstacles.
3. **Path Extraction**:
    - Starting from the initial cell, the algorithm followed decreasing wavefront values to construct the shortest path to the goal.
    - The calculated path was returned as a sequence of cells.

Robot Navigation

1. **Pose Estimation**:
    - The robot's initial position (x, y) and orientation (θ) were obtained using the compass and encoder readings.
    - The robot's starting position was mapped to a cell number using a coordinate-to-cell conversion function.

2. **Movement Strategy**:
    - The robot rotated to align with the target cell using the rotate() function, which adjusted the robot's angle based on compass readings.
    - The robot then moved forward by a fixed distance (equal to the size of one cell) using encoder readings to ensure precise navigation.
3. **Path Execution**:
    - The robot sequentially navigated through the cells in the calculated path, updating its pose at each step.
    - At each new cell, the robot printed its current pose in the format s = (x, y, cell, θ) for verification.

Key Functions

1. **Wavefront Path Planning**:
    - **calculate_wavefront()**: Assigned wavefront values to grid cells based on proximity to the goal.
    - **find_path()**: Extracted the shortest path from the wavefront grid.
2. **Robot Navigation**:
    - **rotate(radianAngle)**: Adjusted the robot's orientation to face the target direction.
    - **moveToNextCell(distance)**: Ensured precise forward movement based on encoder readings.
    - **move_to_cell(target_cell)**: Directed the robot to a specific grid cell, aligning its pose along the way.

Algorithms and Formulas

1. **Wavefront Algorithm**:
    - Incremental propagation: Each cell's value was updated to current_value + 1 for unvisited neighbors.
    - Valid neighbors were determined based on wall configurations in cellStructure.
2. **Pose Estimation**:
    - The robot's pose was updated using encoder readings:

      $$x_{new} = x + d \cdot \cos(\theta)\,, y_{new} = y + d \cdot \sin(\theta)$$

      where d is the distance traveled, calculated from the encoder.
3. **Rotation**:
    - Compass readings were used to align the robot's orientation to the desired angle.

Results

Test Overview

The robot was tested from 10 unique starting cells in the maze environment. For each test:

1. The robot initialized its position using the compass and encoder readings.
2. The Wavefront Planner calculated the shortest path to the goal cell (Cell 7).

3. The robot navigated the calculated path while continuously updating and printing its pose.

Calculated Paths
The table below summarizes the starting cells and calculated paths for each test.

Test Number	Starting Position	Computed Path
1	18	18→17→16→11→6→1→2→3→4→5→10→9→8→7
2	1	1→2→3→4→5→10→9→8→7
3	12	12→11→6→1→2→3→4→5→10→9→8→7
4	3	3→4→5→10→9→8→7
5	7	7 (no movement)
6	9	9→8→7
7	15	15→14→13→12→11→6→1→2→3→4→5→10→9→8→7
8	13	13→12→11→6→1→2→3→4→5→10→9→8→7
9	14	14→13→12→11→6→1→2→3→4→5→10→9→8→7
10	19	19→18→17→16→11→6→1→2→3→4→5→10→9→8→7
11	22	22→21→16→11→6→1→2→3→4→5→10→9→8→7
12	17	17→16→11→6→1→2→3→4→5→10→9→8→7

Notes:

- The calculated paths demonstrate the efficiency of the Wavefront Planner in identifying the shortest route.

Observations

1. **Path Accuracy**:
   - All calculated paths successfully led the robot to the goal cell (Cell 7).
   - The algorithm consistently avoided walls, ensuring obstacle-free navigation.
2. **Pose Printing**:
   - The robot accurately updated and printed its pose (x, y, cell, θ) upon entering each cell.
   - Pose data validated correct alignment and localization within the maze.
3. **Error Handling**:
   - In all tests, the robot handled corner cases (e.g., dead ends) effectively, reorienting itself to find the correct path.

Conclusion

The experiment successfully demonstrated the robot's ability to navigate a structured maze environment using the Wavefront Path Planning algorithm. Starting from 12 unique positions, the robot accurately calculated and followed the shortest path to the goal cell (Cell 7) while avoiding obstacles.

Key Outcomes:

1. **Path Planning**:
    - The Wavefront algorithm consistently provided optimal paths for all test cases, regardless of the starting cell.
    - Calculated paths were free of errors and ensured efficient navigation without revisiting cells unnecessarily.
2. **Navigation Performance**:
    - The robot successfully followed the computed paths and accurately updated its pose during navigation.
    - Encoder readings and compass-based rotations provided precise control, enabling smooth transitions between cells.
3. **Pose Estimation**:
    - The robot's pose was consistently printed in the correct format (x, y, cell, θ) at each step, validating its localization accuracy.

Challenges:

A few challenges were encountered when completing this task.

1. **Slight jerking when moving to next cell:**
    - After rotating to face the next cell, the robot would move forward and when placed in its center, it would slightly shift right, causing the rest of the projector to be thrown off. To fix this issue, the speed at which the robot rotated was decreases and its robots speed when traveling to the next cell was increased.
2. **Float Indexing in wavefront grid:**
    - The computed cell numbers were sometimes floats instead of integers. This caused errors when indexing the wavefront NumPy array, which requires integer indices. To fix this, I ensured all cell calculations returned integers using int() or integer division (//).
3. **Ignoring Walls in Pathfinding:**
    - During wavefront propagation, paths were calculated through walls, even though cellStructure clearly defined wall configurations for each cell. This is because the calculate_wavefront() method did not properly validate walls while propagating values to neighboring cells. To fix this,

Incorporate wall validation by checking the wall configuration in cellStructure:

Videos

Task 1: Lab5Task1.mov

Author Statement

I developed all the code and written report with the following exceptions:
Task 1: Lines [83-177, 181-279,]
[Claude Watson: C.W., 12/1/2024]