Autonomous Mobile Robots

CDA 4621

Lab Report 4

Localization

Claude Watson

Objective

The goal of task 1 was to program the robot to navigate a grid-like maze environment, identify its location probabilistically, and visit a predefined set of cells using trilateration. The task integrated key algorithms for scanning, movement, and localization. The robot leveraged LIDAR and RGB camera data to achieve accurate localization and navigate through the maze while avoiding obstacles.

The objective of task 2 was to implement a probabilistic localization system and navigate a grid-like maze environment. The robot identified its most likely position using sensor model probabilities derived from LIDAR-based wall observations. The goal was to move across 15 cells while updating its localization.

Methodology

Task 1 – Localization Using Landmarks

The robot was programmed to navigate a grid-like environment and determine its location probabilistically using trilateration and Gaussian probability calculations.

1. Distance Measurement:
    - The robot scanned the environment using its RGB camera and measured distances to colored cylinders (red, green, and blue). These measurements were used for probabilistic localization.
2. Localization:
    - Precomputed distances to the landmarks for each grid cell were compared with measured values using a Gaussian function.
    - The cell with the highest probability was chosen as the current location.
3. Trilateration:
    - The robot calculated its position using the trilateration formula based on the distances to three distinct landmarks. The result was validated against GPS readings.
4. Movement Strategy:
    - The robot first moved to the first cell before beginning to calculate probabilities to ensure all cells would be accounted for and avoiding unnecessary revisits

- The robot scanned for obstacles (walls) using LIDAR and turned accordingly.

Key Functions:

1. Localization Functions:
    - scanLandmarks: Measured distances to landmarks and returned results.
    - calculateProbabilities: Calculated the likelihood of each cell being the robot's location based on Gaussian distribution.
2. Navigation Functions:
    - moveToNextCell: Controlled movement to the next cell in the grid.
    - rotate: Rotated the robot to a specific angle for alignment. Used when a wall was directly infront of the robot.
    - fixDirection: Ensured the robot's orientation was maintained.
3. Trilateration:
    - Calculated the robot's position using distances to landmarks.

Algorithms and Formulas

1. Trilateration:

$$x = \frac{C1 \cdot B2 - C2 \cdot B1}{B2 \cdot A1 - B1 \cdot A2}$$

$$y = \frac{C1 \cdot A2 - A1 \cdot C2}{B1 \cdot A2 - A1 \cdot B2}$$

2. Gaussian Probability:

$$P = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{(s-\mu)^2}{2\sigma^2}\right)}$$

Task 2: Probabilistic Localization and Navigation

1. Probabilistic Localization:

- The robot used LIDAR readings to observe the presence or absence of walls in four directions: North, East, South, and West.
- Observed wall configurations were matched with a pre-defined grid structure to calculate probabilities for all cells.

2. Sensor Model:
    - Probabilities were normalized across all cells to ensure accurate localization.
3. Movement Strategy:
    - After localizing, the robot aligned itself to face north using the compass and took lidar measurements to determine its surroundings.
    - It moved one cell forward unless an obstacle was detected.
    - If obstacles were present in multiple directions, the robot rotated accordingly to navigate around them.
4. Main Control Loop:
    - The robot continued localizing and navigating until it visited 15 unique cells or encountered a stopping condition.

Key Functions

1. Localization Functions:
    - takeLidarReadings: Mapped LIDAR readings to wall presence.
    - calculateProbabilities: Used the sensor model to compute and normalize probabilities for all grid cells.
2. Navigation Functions:
    - moveToNextCell: Controlled the robot's movement to the next cell.
    - rotate: Adjusted the robot's orientation when obstacles were encountered.
    - fixDirection: Ensured the robot maintained its initial heading after rotations.

Algorithms and Formulas

1. Sensor Model Probabilities:
    - Weighted Probabilty for each cell:
        - $P(z = 0|s = CN) = P(znorth = 0|s = CN) \times P(zsouth = 0|s = CN) \times P(zeast = 0|s = CN) \times P(zwest = 0|s = CN)$
    - Normalized Probability: $P(cell) = \frac{P(cell)}{\sum_{i=1}^{25} P(cell_i)}$
2. Compass-Based Rotation:

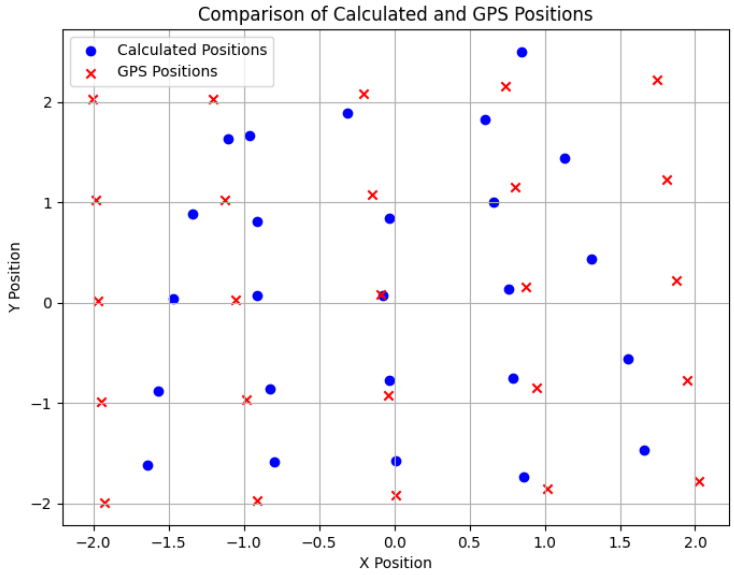
    Compass readings ensured alignment with cardinal directions (e.g., North).
3. Movement:

    Encoder readings ensured precise movements to adjacent cells.

Results

The robot was able to successfully complete both tasks:

- In localization with landmarks, the robot was able to successfully navigate to all the cells in the maze and correctly predict its current position.
- Because of the lawnmower technique used, cells were not unnecessarily revisited.
- The calculated x and y positions of the robot were compared to the actual x and y of the robot, obtained from the gps and are plotted in a graph below:



-

- The comparison between calculated positions (blue dots) and GPS positions (red crosses) shows that the trilateration algorithm performs well in estimating the robot's location, with close alignment in many areas. However, deviations are observed, particularly near the grid edges, likely due to sensor noise, measurement inaccuracies, or environmental factors. While the algorithm demonstrates reliability under optimal conditions, refining the distance measurements, improving the error model, and integrating additional data sources could further enhance accuracy and reduce discrepancies. Overall, the system provides a solid foundation for localization but leaves room for improvement in edge-case scenarios.

- In the localization using wall configuration and sensor readings, the robot was able to calculate and print the probability of each cell, indicating which cell it may be in by printing the cell with the highest probability.

- The robot was able to successfully navigate through 15 cells while displaying probabilities, and only revisiting cells if it encountered a dead end.

- If some cells had the same probability, those cells were printed to the console, otherwise, the cell with the highest probability was indicated in the console also.

Conclusion

Both tasks were able to be completed, however there were several difficulties throughout each task.

Task 1:

1. Determining how to navigate through the maze:
    - Several approaches were tried before the one implemented in the code. The main issue with these previous methods was getting the robot to determine and travel to unvisited cells. The robot would often revisit the same cells repeatedly and never reach these unvisited cells. To combat this, I thought of the idea to make sure the robot travels to the first cell, then travel in a zigzag pattern to travel to each cell

2. Incorrect cell probabilities:
    - During testing an incorrect would often be printed. I was able to fix this issue by properly indexing the print the correct cell number. I also tested the capability of the prediction function by starting the robot at random locations, and the accurate locations were always returned

Task 2:

1. Determining how to gather surrounding readings:
    - Due to how to data structure is implemented, it was a challenge to think of a way for the robot to accurately gather information about its surrounding walls while its orientation was continuously changing. I decided to have the robot face north at every cell, take the readings, and then go back to its orientation before facing north. This allowed me to accurately gather and compare the robots surrounds with the already known information

2. Robot not accurately returning to initial orientation:
    - I was faced with the issue of the robot not being able to return to its position before facing north. So, after the fixDirection function was called, it would face another way. I was able to fix this by including a threshold to accurately turn and was able to fix this error.

Videos

Task 1: Lab4Task1.mov

Task 2: Lab4Task2.mov

Author Statement

I developed all the code and written report with the following exceptions:

Task 1: Lines [68-79, 120 -150, 157-200, 200-278, 281-331]

Task 2: Lines [151-191, 194-235]

[Claude Watson: C.W., 12/1/2024]