Alex: An AI-Powered Personal Manager – Autonomous Navigation System

Pakeeza Anjum
Department of Computer Science
Gulam Ishaq Khan Institute of Technology
Swabi, Pakistan
Email: pakeezaanjum@gmail.com

***Abstract*—This project aims to design and implement an AI-powered personal manager, named **Alex**, with an autonomous navigation system. The primary focus is on simulating real-world autonomous navigation using a pathfinding algorithm, specifically Dijkstra's algorithm, along with the integration of IR sensors for location detection and route computation. The project also incorporates the visualization of internal data structures like adjacency lists and routing tables for educational purposes.**

***Index Terms*—Autonomous Navigation, Dijkstra's Algorithm, IR Sensors, Data Structures, Robotics.**

I. Project Title

Alex: An AI-Powered Personal Manager – Autonomous Navigation System

II. Problem Statement & Motivation

Autonomous navigation is a critical task in robotics. Robots must efficiently determine their position, understand their surroundings, and reach designated destinations without human intervention. The goal of this project is to create **Alex**, a personal assistant capable of autonomous decision-making and navigation. Specifically, Portion B of this project will focus on simulating how Alex determines its location using IR sensors, computes the shortest path using algorithms, and displays routing metrics clearly to the user. This will simulate a real-world robotic assistant in environments like hospitals, warehouses, and smart homes, where precise and optimized movement is crucial.

III. Project Objectives

The key objectives of the project are:

- Simulate real-world autonomous navigation using a pathfinding algorithm.
- Allow Alex to interpret sensor input, compute distances, and calculate the most optimal route.
- Visualize internal data structures such as adjacency lists and routing tables.
- Demonstrate the integration of algorithmic design with robotic movement logic, including tire rotations.

IV. Key Data Structures to be Used

To implement the project, the following data structures will be used:

- **Graph using Adjacency Lists**: Represents the environment, where nodes are locations and edges are paths with weights.
- **Priority Queue (Min-Heap)**: For efficiently selecting the node with the minimum tentative distance in Dijkstra's algorithm.
- **Arrays**: To store distances, parent references, and processed statuses for each vertex.
- **Vectors**: For dynamic list maintenance, especially for neighbors in the adjacency list.
- **Recursive Functionality**: To backtrack from a node to reconstruct the shortest path.

V. Algorithmic Strategy

The project will use **Dijkstra's Algorithm**, a greedy algorithm for computing the shortest path in a weighted graph. The steps are as follows:

1) Initialize all nodes with an infinite distance and unvisited status.
2) Set the source node's distance to zero.
3) Use a min-priority queue to process the node with the smallest tentative distance.
4) Update neighbors if a shorter path is found, and push them back into the queue.
5) Store and update parent pointers to reconstruct the path later.

Additionally, **Euclidean distance** will be used to calculate the distance between coordinates:

$$\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Tire rotations required are calculated by dividing this distance by the tire's circumference.

VI. Input and Output Design

A. Input Design

- Random (x, y) coordinates generated to simulate destination input via IR sensors.
- A predefined weighted graph with 6 nodes and various edge weights.

B. Processing Flow

1) Generate and display destination coordinates.
2) Compute the distance using the Euclidean formula.
3) Convert distance into tire rotations.
4) Run Dijkstra's algorithm to determine the optimal path.
5) Display routing tables after each node processing.
6) Reconstruct and print shortest paths.

C. Output Design

- Display destination coordinates from IR sensors.
- Output distance and tire rotations.
- Show adjacency list of the environment graph.
- Display Dijkstra's routing table with distance and parent updates.
- Final shortest paths with backtracking steps.

VII. Integration with Course Concepts

This project integrates core topics from the *Data Structures and Algorithms* course:

- **Graph Theory**: For representing the environment and connectivity.
- **Greedy Algorithms**: Dijkstra's approach is a greedy strategy for shortest paths.
- **Priority Queues**: Used for efficient node selection in pathfinding.
- **Backtracking and Recursion**: For reconstructing paths using parent relationships.
- **Complexity Analysis**: The time complexity of the algorithm is $O((V + E) \log V)$.

VIII. Future Scope & Extensions

The scope of the project can be extended by:

- Dynamically expanding the graph with live sensor data.
- Integrating with hardware platforms like Arduino or Raspberry Pi for physical navigation.
- Implementing more advanced algorithms like A* (A-star) for faster pathfinding.
- Introducing obstacles in the environment that require path replanning.

IX. Conclusion

In conclusion, *Portion B* of the ”Alex” project demonstrates autonomous robotic navigation, integrating data structures, algorithms, and real-world constraints. This proposal outlines a system that not only enhances understanding of theoretical concepts but also serves as a stepping stone towards creating intelligent robotic systems capable of practical real-world tasks.