Implementing Path Planning with Probabilistic Roadmaps (PRM)

2D Image Map Implementation

Lalit Karthikeyan M A

Abstract— **This project aims to navigate a maze efficiently by constructing a pathfinding solution. It involves initializing a roadmap class and identifying the maze's effective area through systematic exploration. After detecting obstacles using thresholding techniques, starting and ending points are determined. Bresenham's Line Drawing Algorithm ensures obstacle-free paths during roadmap construction. Then, the A* algorithm is used to navigate the maze, evaluating nodes and finding optimal paths. This integrated approach promises accurate and efficient navigation through complex maze environments, showcasing its potential for real-world applications in robotics.**

The pathfinding solution developed in this project holds versatile applications. It aids robotics in autonomous navigation within dynamic environments like warehouses. For autonomous vehicles, it optimizes route planning, enhancing safety on roads. Augmented and virtual reality benefit from immersive navigation simulations. During search and rescue operations, it assists in locating survivors efficiently in hazardous terrain. In supply chain management, it streamlines logistics, reducing travel time in warehouses and delivery networks.

Its scalability is notable, adapting to different scales, complexities, and hardware platforms. Integrated with sensor technologies, it ensures accurate real-time navigation. Overall, the solution offers diverse navigation capabilities across industries, fostering advancements in robotics, transportation, virtual environments, emergency response, and supply chain management.

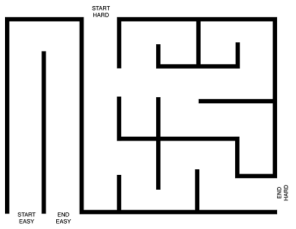
I. INTRODUCTION

In this project, I employed a Probabilistic Roadmap (PRM) for navigating through a maze. Initially, I used a movable circle-shaped turtle to map the maze, identifying the effective sampling area and also get the coordinates of the two starting and ending points. Within this region, I generated random points, ensuring they avoided obstacles, and created a connectivity graph by linking each point to its nearest neighbors. Then, employing the A* algorithm, I computed optimal paths from two different starting positions to their corresponding goal positions.

II. PROBLEM STATEMENT

In this project, the focus is on motion planning, which is crucial for robots to navigate complex environments while avoiding obstacles. The specific path planning algorithm taht is supposed to be implemented is called Probabilistic Roadmaps (PRM) in a two-dimensional (2D) environment.

The PRM algorithm is all about creating a roadmap within the robot's workspace, which basically means it's building a map of feasible paths the robot can take. Implementation is to be done on an image named maze.png which is given below.



The program needs to handle both the "Start Easy" and "Start Hard" scenarios within the maze. Here's what the program should do:

1. Read the maze.png image to identify obstacles: The program should be able to interpret the maze image and recognize where the obstacles are located. This step is crucial as it determines where the robot can and cannot move.

2. Implement the PRM algorithm: Once the obstacles are identified, we'll use the PRM algorithm to find a path between the start point and the end point in the maze. The PRM algorithm will strategically place nodes and connect them to form a roadmap that the robot can follow to reach its destination.

3. Visualize the implementation on the original image map: After finding the path using the PRM algorithm, the program should visualize this path on the original maze image. This visualization will help us understand how the robot plans to navigate through the maze, showing us the optimal route it intends to take.

This initial implementation is essential as it sets the groundwork for dealing with more complex environments in the future. By successfully implementing the PRM algorithm for this 2D image map, solid foundation for tackling more intricate challenges in robotics motion planning is laid down.

Through this project, I am not only learning about PRM and its application in robotics but also gaining practical experience in implementing algorithms to solve real-world problems. This hands-on approach is invaluable for understanding the nuances of motion planning and preparing me for more advanced robotics projects in the future.

III. RELATED WORK

To ensure the path avoids obstacles, I implemented Bresenham's line algorithm from scratch. By referring to the pseudocode and understanding its concept, I could accurately check if the line crosses any obstacle. This approach allowed

me to navigate the map effectively, identify obstacles, and determine feasible paths for the robot to follow.

To explore the map and find the effective area, I created a function to move a circle-shaped turtle. This function helped me gather coordinates for both starting and ending points.

IV. INITIAL ATTEMPTS

At first, I considered using contour detection to identify obstacles and incorporate them into the roadmap. However, given that the image is grayscale, I found it more effective to apply thresholds to distinguish black obstacles from the white path.

Initially, I planned to sample the map twice for two different starting points using different thresholds. But upon further consideration, I realized it would be more efficient to sample the entire map once and then define the two starting points. By doing this, I could use the same nodes initially sampled to find paths for both starting points. This approach simplified the process and ensured consistency in path planning using the same set of nodes, optimizing the overall efficiency of the algorithm.

V. FINAL APPROACH

In this project, a roadmap class was initialized to facilitate the process of pathfinding through a maze. The class encompassed essential attributes such as Nodes, Obstacles, and Neighbours, forming the basis for constructing an effective navigation strategy. The first crucial step involved determining the effective area within the maze where sampling would be conducted. This was accomplished by utilizing a turtle to systematically explore the maze and identify its boundaries, ensuring comprehensive coverage for subsequent analysis.

Once the effective area was found, the next objective was to identify the coordinates of two distinct starting points and their corresponding endpoints. This task was achieved through careful navigation of the turtle across the maze, pinpointing strategic locations for initiating and terminating the pathfinding process. With the starting and ending points established, the focus shifted towards detecting obstacles within the maze.

Thresholding techniques were applied to the grayscale maze image to differentiate between obstacles and clear pathways. By visually marking obstacles with a distinct color, they were readily identifiable and could be incorporated into the pathfinding algorithm. With obstacles identified and accounted for, the roadmap construction could proceed by sampling the entire effective area of the maze.

Sampling involved systematically selecting points within the effective area to serve as potential nodes for pathfinding. However, to ensure the accuracy of the roadmap, points that fell within obstacles were excluded from consideration. This step was crucial in eliminating potential path obstructions and refining the roadmap's integrity.

Once the nodes were determined, the roadmap's connectivity was established by identifying neighbouring nodes for each sampled point. To ascertain valid connections between nodes, Bresenham's Line Drawing Algorithm was employed. This algorithm allowed for the precise determination of whether a line between two nodes intersected with any obstacles, thus ensuring obstacle-free paths throughout the maze. It works by incorporating the following steps:

1. Given two points the algorithm determines which direction the line is mostly going in: either horizontally or vertically.

2. It calculates the slope of the line and decides which direction to move in each step.

3. Then, it iterates over the range between the x-coordinates of the two points, determining the corresponding y-coordinate for each x-coordinate along the line. This is done by considering the slope and rounding off to the nearest integer.

4. At each step, it chooses the pixel that is closest to the true line. This is done by comparing the distances between the true line and the two possible pixels.

5. The algorithm continues this process until it reaches the endpoint, if in between any of the points lie in the obstacle that is considered as passing through the obstacle and is thus not added as a path.

With the roadmap fully constructed and validated, the final stage involved implementing the A* algorithm for pathfinding. A* (A-star) is a widely used pathfinding algorithm known for its efficiency and accuracy in finding the shortest path between two points in a graph or grid. Here's a brief overview of how it works:

The algorithm starts by selecting a starting node and a goal node.

1. Evaluation: A* evaluates nodes based on two cost values: g(n): the cost of reaching node n from the starting node. h(n): the heuristic estimate of the cost to reach the goal node from node n.

2. Priority queue: A* maintains a priority queue of nodes to be explored. Each node is prioritized based on the sum of its g(n) and h(n) values. Nodes with lower total costs are explored first.

3. Expansion: A* iteratively selects and expands the node with the lowest total cost from the priority queue. It explores neighboring nodes, calculating their g(n) and h(n) values and adding them to the priority queue if they have not been visited or have a lower total cost.

4. Termination: The algorithm terminates when the goal node is reached or when there are no more nodes to explore.

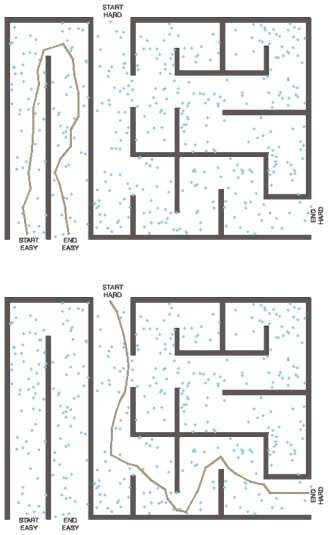
5. Path reconstruction: Once the goal node is reached, A* traces back through the explored nodes to reconstruct the shortest path from the starting node to the goal node.

By efficiently combining the cost of reaching a node from the starting node (g(n)) and the heuristic estimate of the cost to reach the goal node from the current node (h(n)), A* can

quickly find the optimal path while exploring fewer nodes compared to other search algorithms.

VI. Results and Observation

The net output is as follow:



1. Path Length: A* and RRT algorithms provide comparable path lengths, with slight variations depending on the environment's complexity. Dijkstra's algorithm may produce longer paths due to its nature of exploring all possible paths.

2. Computational Complexity: A* algorithm's computational complexity is moderate, making it suitable for real-time applications. RRT has low computational complexity, making it efficient for high-dimensional spaces. Dijkstra's algorithm exhibits high computational complexity due to its exhaustive search approach.

3. Memory Usage: A* and RRT algorithms have moderate to low memory usage, whereas Dijkstra's algorithm requires high memory usage due to storing information about all nodes.

VII. Future Work

1. Optimization Techniques: Implementing optimization techniques such as pruning, caching, or parallelization can help reduce the computational burden of the algorithm, improving its efficiency.

2. Path Smoothing: Post-processing techniques can be applied to smooth out the generated paths, resulting in more navigable trajectories for robotic systems

CONCLUSION

Developing a pathfinding solution for maze navigation presented both challenges and opportunities. The solution's usefulness extends to various fields, including robotics, autonomous vehicles, and logistics, where efficient navigation is critical. However, the task was not without difficulties. Balancing computational efficiency and path optimality posed significant challenges. Despite these hurdles, this solution successfully overcame these challenges. It offers optimized paths through maze environments, enhancing navigation capabilities in complex settings. This solution holds immense value, empowering robots to autonomously navigate through intricate mazes, improving operational efficiency, and opening doors for applications in warehouse automation, manufacturing, and search and rescue operations. Overall, this pathfinding solution addresses real-world navigation challenges, offering a versatile and adaptable approach to complex environment traversal.

References

[1] Kavraki, L. E., et al. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces.
[2] Mohammad Zunnun Khan University of Bisha. Improved Line Drawing Algorithm: An Approach and Proposal. Conference: Proceedings of the International Conference on Advances in Computer Science and Electronics Engineering
[3] "https://en.wikipedia.org/wiki/Bresenham
[4] "https://en.wikipedia.org/wiki/A*searchalgorithm