Path Planning for Autonomous Robot Navigation: Maximum Safety vs. Minimum Distance

Jiajun Li
Email: jil186@ucsd.edu
Department of ECE, UCSD

Tingyu Shi
Email: t5shi@ucsd.edu
Department of ECE, UCSD

***Abstract*—This paper presents the implementation of two distinct path planning algorithms for an autonomous robot navigating a predefined environment with obstacles. The objective was to design a path planner that balances two criteria: maximum safety and minimum distance. The maximum safety approach ensures that the robot follows a path that maintains the greatest possible distance from obstacles, while the minimum distance approach determines the shortest possible route between a start and goal point. We evaluate the performance of both methods in a controlled experimental setup and provide comparative analysis.**

I. Introduction

Path planning is a fundamental problem in robotics, impacting applications from autonomous navigation to industrial automation. In this work, we explore two contrasting approaches: maximizing safety versus minimizing distance. The environment consists of a 10ft $\times$ 10ft workspace with known landmarks and an obstacle. The start and goal points are placed diagonally across the workspace. This paper describes the design choices, algorithms, and results obtained through real-world experiments.

II. System Architecture

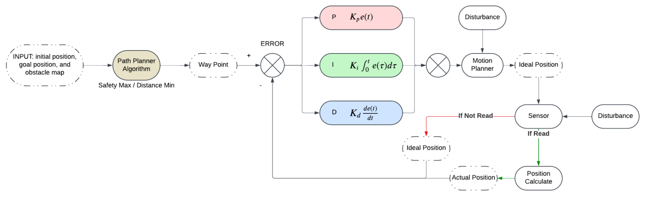


Fig. 1: Model Architecture

As illustrated in the figure above, the majority of the model remains consistent with the previous assignment. However, in this version, we do not directly provide waypoints to the robot. Instead, the waypoints are generated by a path planner algorithm. For this task, we have implemented two distinct path planning algorithms: the Maximum Safety Algorithm and the Minimum Distance Algorithm. The Maximum Safety Algorithm focuses on creating a path that keeps the robot as far from obstacles as possible, ensuring its safety. In contrast, the Minimum Distance Algorithm seeks to determine the shortest route from the starting point to the goal. The structure and details of each algorithm will be discussed further in the following sections of this report.

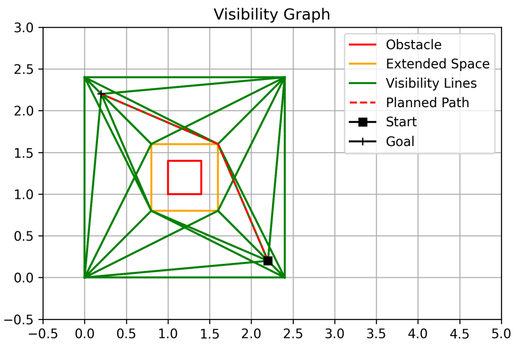


Fig. 2: Visibility Graph for Minimum Distance

III. Path Planning Algorithms

A. Minimum Distance Path Planning

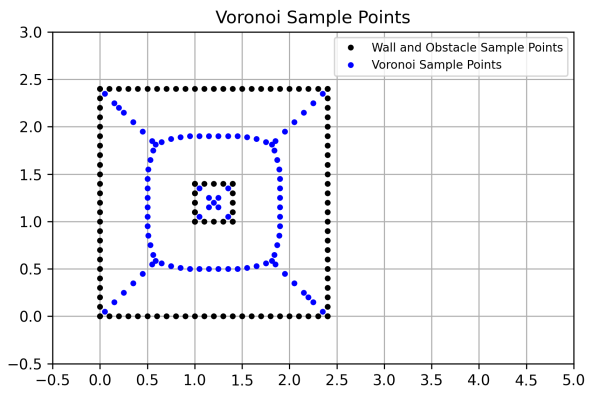
For minimum distance path planning, we used the visibility graph and Dijkstra algorithm as shown in Figure 2. We set an extended space(orange square) around the obstacle so that the car would not hit the obstacle. Then, we generated the visibility graph(represented by the green lines) using the following points:

- 4 corners of wall
- 4 corners of the extended space
- start point and end point

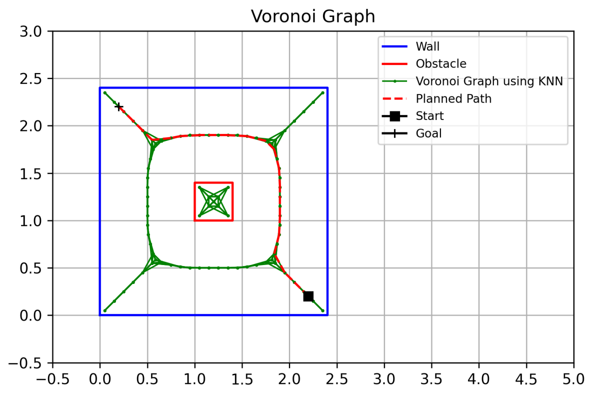
After getting the visibility graph, we used the Dijkstra algorithm to find the shortest path between the start and end points as the planned path. The red dashed line labels the planned path.

B. Maximum Safety Path Planning

For maximum safety path planning, we first used the Voronoi algorithm to generate sample points as shown in Figure 3(a). The black dots represent the wall and the obstacle, and the blue dots represent the Voronoi samples. Then, we used KNN(N = 5) to connect the blue dots to generate a graph, which is represented by green lines in 3(b). After this, use Dijkstra algorithm to find the shortest path between the start point and the end point as the planned path, which is represented by the red dashed line in 3(b).



(a) Voronoi Sample Points



(b) Voronoi Graph

Fig. 3: Voronoi-Based Maximum Safety Path Planning

IV. Experimental Results

A. Minimum Distance Path Execution

As shown in Figure 4, the red dashed line represents the planned path for minimum distance. The green dots represent the estimated car's position. Since there is always noise for car position estimation, the estimated car's positions are not exactly the same as the planned path.

B. Maximum Safety Path Execution

As shown in Figure 5, the red dashed line represents the planned path for maximum safety. The green dots represent the estimated car's position. Since there is always noise for car position estimation, the estimated car's positions are not exactly the same as the planned path. However, we can see that the car is roughly following the path.

V. Conclusion and Future Work

This work presents two contrasting approaches to path planning for autonomous navigation. The minimum distance algorithm effectively computes the shortest route using

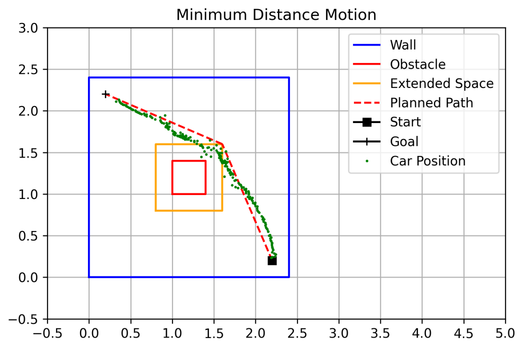


Fig. 4: Minimum Distance Car Motion

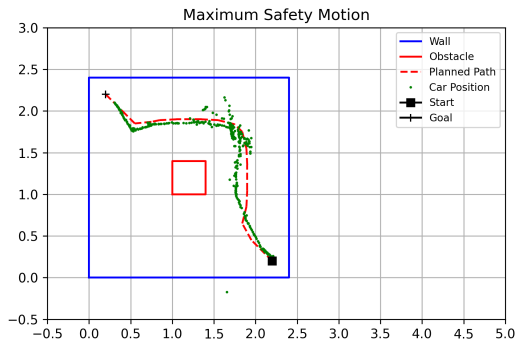


Fig. 5: Maximum Safety Car Motion

visibility graphs and Dijkstra's algorithm, while the maximum safety approach leverages Voronoi diagrams for obstacle avoidance. Experimental results validate both methods, demonstrating their effectiveness in different scenarios.

Future work will explore:

- Integrating dynamic obstacle avoidance mechanisms
- Enhancing localization accuracy using sensor fusion techniques
- Developing adaptive planning methods that balance safety and efficiency in real time

These improvements will further enhance the robustness of autonomous navigation systems.

References

[1] A. Sakai, "Visibility Road Map," PythonRobotics, 2024. Available: https://github.com/AtsushiSakai/PythonRobotics/tree/master/PathPlanning/VisibilityRoadMap.

[2] A. Sakai, "Voronoi Road Map," PythonRobotics, 2024. Available: https://github.com/AtsushiSakai/PythonRobotics/tree/master/PathPlanning/VoronoiRoadMap.