CSE 276A HW4

Tingyu Shi(A59023729)
Jiajun Li(A16635772)

November 25, 2024

1 Video URL

- Maximum Safety: https://youtu.be/FdMfx6F-wpE
- Minimum Distance: https://youtu.be/pjOQJUsrepY

2 Environment Setup

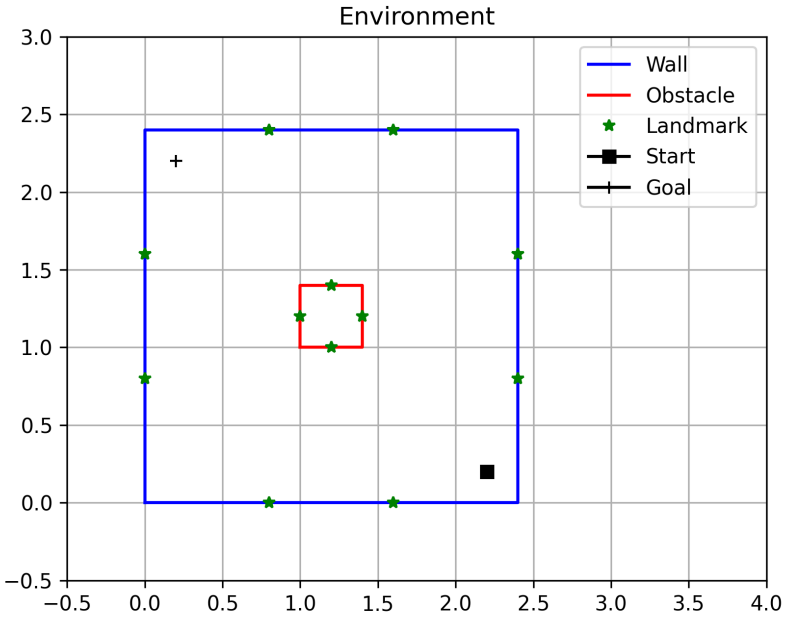


Figure 1: Environment Diagram

Our experiment environment is shown in figure 1. The wall is a square with a side length of 2.4m. The obstacle is a square with a side length of 0.4m. The start point is located at (2.2, 0.2), and the end point is located at (0.2, 2.2). We put 8 landmarks evenly on the wall and 4 landmarks evenly on the obstacle.

3 Architecture

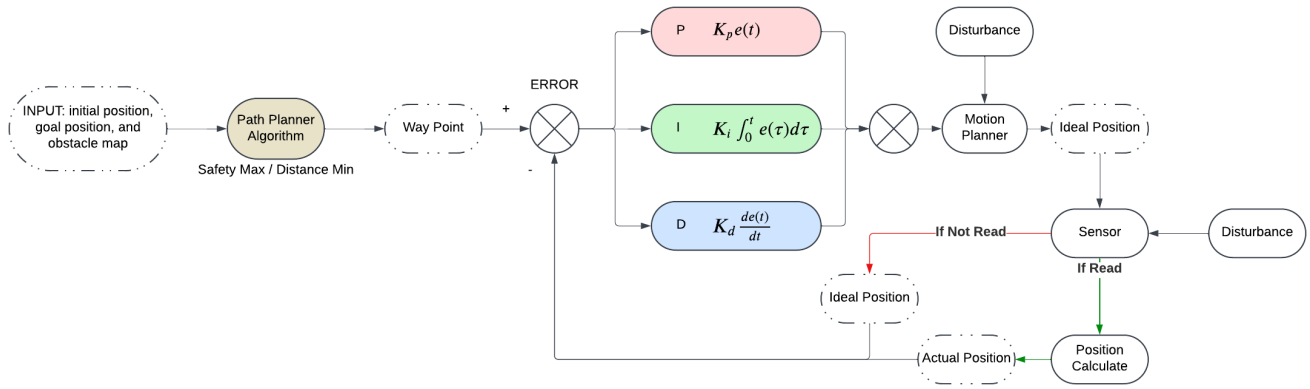


Figure 2: Model Architecture

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.

4 Representation and Algorithm

4.1 Minimum Distance

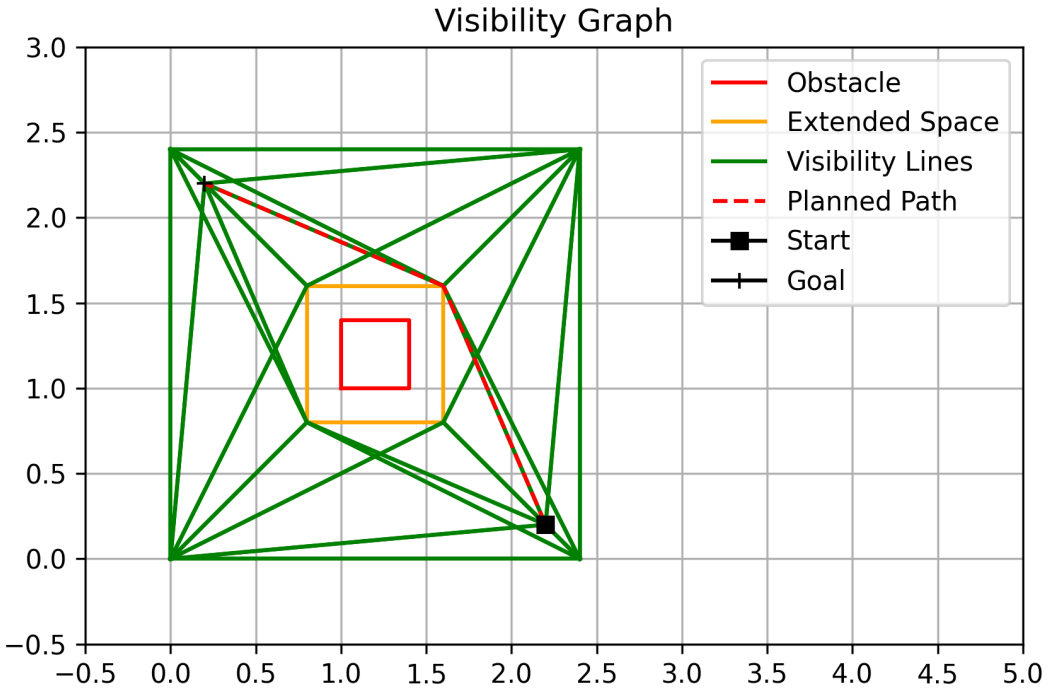


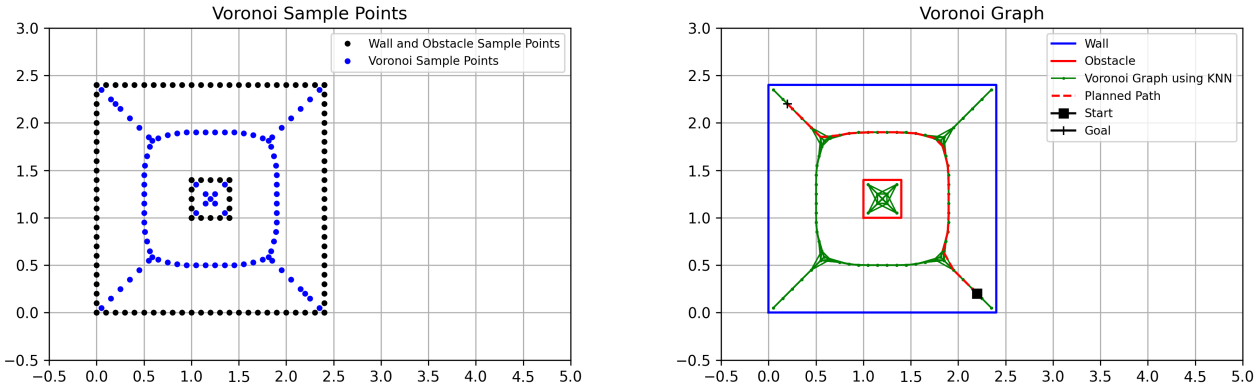
Figure 3: Visibility Graph

For minimum distance path planning, we used the visibility graph and Dijkstra algorithm as shown in figure 3. 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.

4.2 Maximum Safety



(a) Voronoi Sample Points

(b) Voronoi Graph

Figure 4: Voronoi Graph

For maximum safety path planning, we first used the Voronoi algorithm to generate sample points as shown in figure 4(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 4(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 4(b).

5 Results

5.1 Minimum Distance

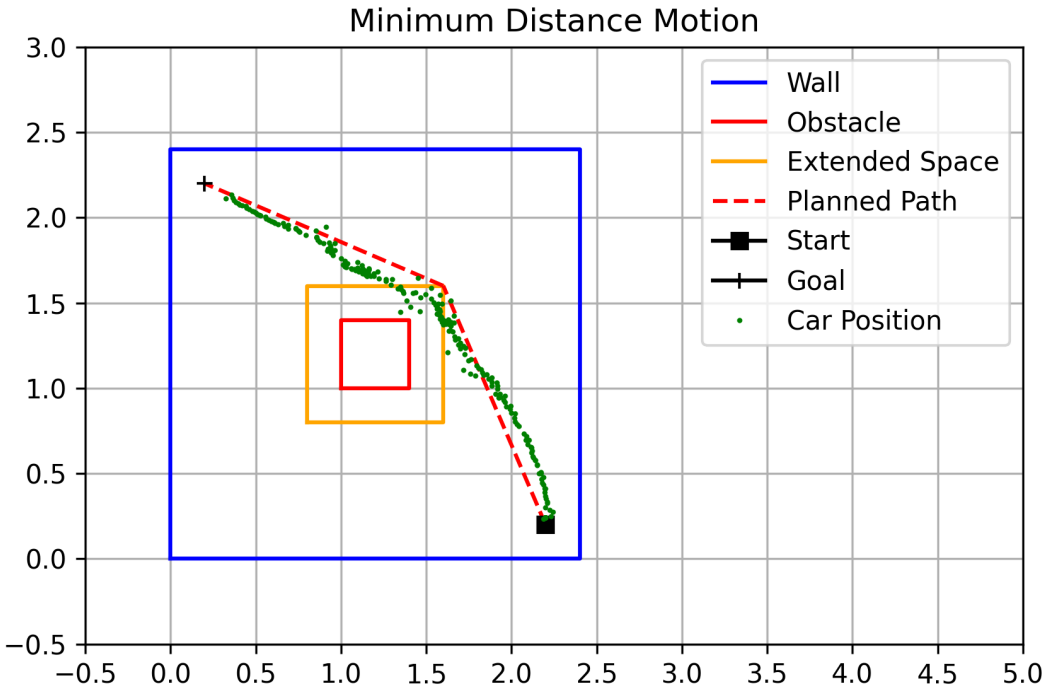


Figure 5: Minimum Distance Car Motion

As shown in figure 5, 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.

5.2 Maximum Safety

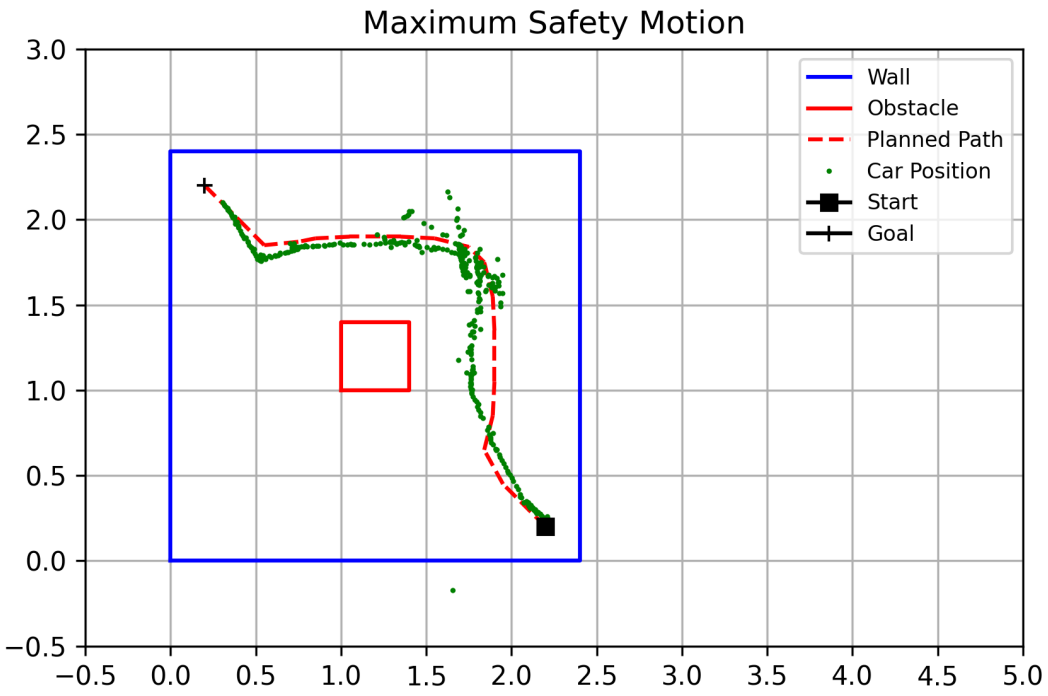


Figure 6: Maximum Safety Car Motion

As shown in figure 6, 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.

6 Reference

- VisibilityRoadMap from PythonRobotics.

  VoronoiRoadMap from PythonRobotics.