CSE 276A HW5

Tingyu Shi(A59023729)
Jiajun Li(A16635772)

December 7, 2024

1 Video URL

Without Using SLAM: https://youtu.be/sCGQzoF28NY

2 Environment Setup

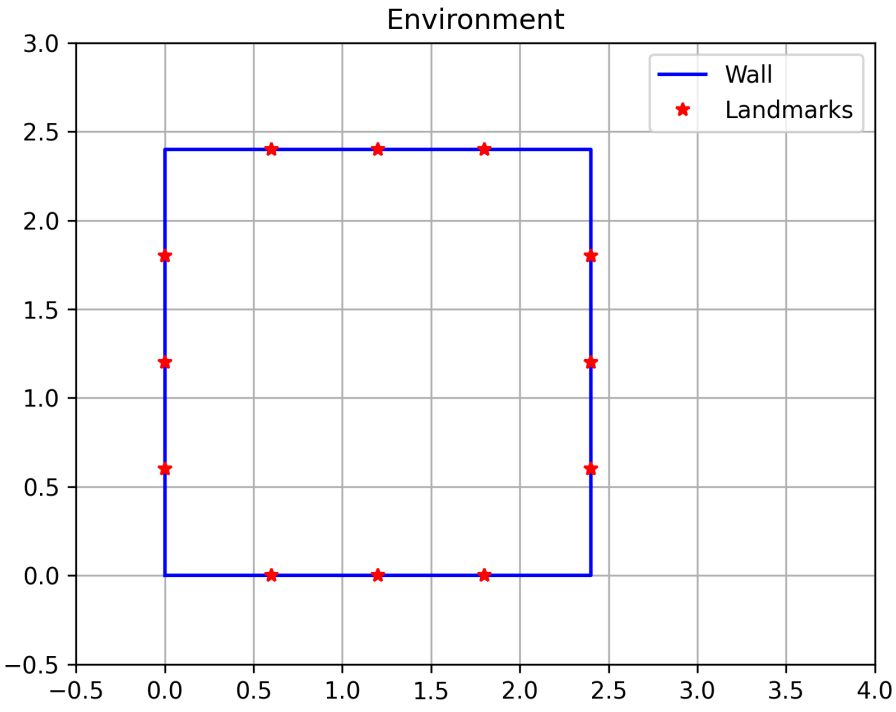

Figure 1: Experiment Environment

Figure 1 shows the experiment environment. The boundary is a square with a side length of 2.4 meters.

3 System Architecture

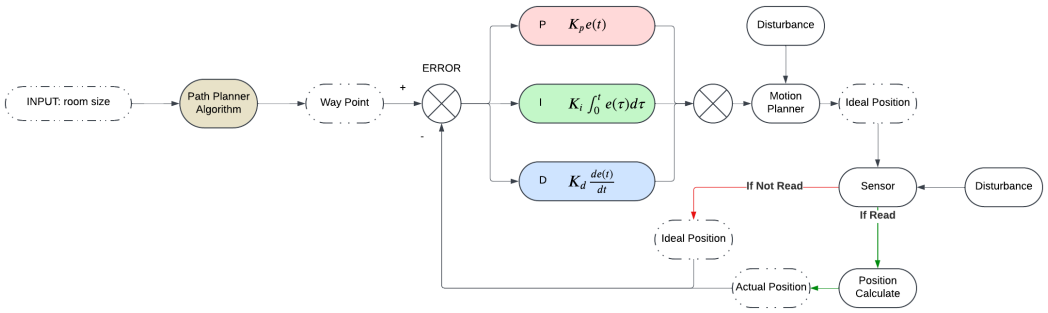

Figure 2: System Architecture

Figure 2 illustrates the architecture of our system, which shares similarities with the one from our previous homework. However, the objective of this project differs significantly. Instead of finding the safest or shortest route to a specific goal, our focus is on designing a path that efficiently covers the entire area. While this new objective introduces a distinct challenge, the remaining components of the architecture remain consistent with those used previously.

4 Module Description

4.1 Path Planning Module

Please refer to section 5 for the coverage path planning algorithm.

4.2 PID Control and Motion Module

For each waypoint, the desired location and pose are first set. The error is then calculated using either the position and pose estimated by the april detection node or those estimated by the algorithm. This error is processed by the PID controller, which generates a twist signal. This signal serves as input for the MPI twist control node and also provides feedback for the closed-loop system when no tag is detected. The MPI twist control node calculates and sends PWM signals to the four motors using the kinematic model. These motors generate the robot's real-world motion, which is captured by the camera. Finally, if the april detection node detects at least one tag, the current state is estimated based on the results provided by the april detection node.

5 Coverage Path Planning Algorithm

5.1 Algorithm Description

For the path planning algorithm, we used the Grid-based sweep algorithm. The first step is to discretize the space into different grids. Then, we use the center point of each grid to represent this grid. After discretizing the space, it can be considered as a 2D matrix, assume this matrix has N rows and M columns.

Algorithm 1 Grid-Based Sweep CPP Algorithm

Require: x_coords : *2D matrix with shape* $(N \times M)$ *representing x coordinates of all the grids*
Require: y_coords : *2D matrix with shape* $(N \times M)$ *representing y coordinates of all the grids*
$visited \leftarrow$ *2D boolean matrix with shape* $(N \times M)$ *full of False values*
$row\_idx, col\_idx \leftarrow N - 1, 0$ ▷ Assume the car starts from left bottom and move upward
$waypoints \leftarrow []$
while $True$ **do**
  $visited[row\_idx, col\_idx] \leftarrow True$ ▷ Mark Visited Grid
  $x\_coord \leftarrow x\_coords[row\_idx,\ col\_idx]$ ▷ Extract X coordinate
  $y\_coord \leftarrow y\_coords[row\_idx,\ col\_idx]$ ▷ Extract Y coordinate
  $waypoints \leftarrow waypoints + [[x\_coord,\ y\_coord]]$ ▷ Update Waypoints
  if Robot Can Move to the Right Grid **then**
    $col\_idx \leftarrow col\_idx + 1$
  else if Robot Can Move to the Left Grid **then**
    $col\_idx \leftarrow col\_idx - 1$
  else if Robot Can Move to the Up Grid **then**
    $row\_idx \leftarrow row\_idx - 1$
  else
    $Break$
  end if
end while

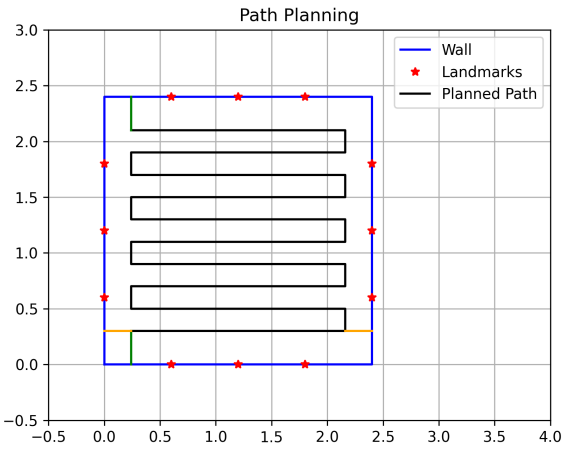

Figure 3: Planned Path

Algorithm 1 shows the pseudo-code of our coverage path planning algorithm. The car can only move to a certain grid if that grid has never been visited.
Figure 3 shows the planned path. The generated path is in a zigzag shape. Due to the noise in the localization, we also set certain distances between the path and the wall so that the car does not hit the wall. In figure 3, the green line represents the safe distance between the path and the up/bottom wall, they are all 0.3m. The orange line represents the safe distance between the path and the left/right wall; they are all 0.24m.
The black line in figure 3 represents the planned path, the distance between any consecutive horizontal lines is 0.2m, which is the same as the car's width. By doing this, we can maximize our area coverage.

5.2 Coverage Guarantee

As shown in algorithm 1, the algorithm stops if and only if the car cannot move left, right, and up. By designing in this way, the algorithm can guarantee that every discretized grid can be visited. Assume that the square space has a side length of S, the car's width is W, and the square space is vertically separated into N rows, then as long as we can ensure that $\frac{S}{N} \leq W$, we can say that this algorithm can reach 100% coverage. However, this needs accurate motion control and localization.

6 Result

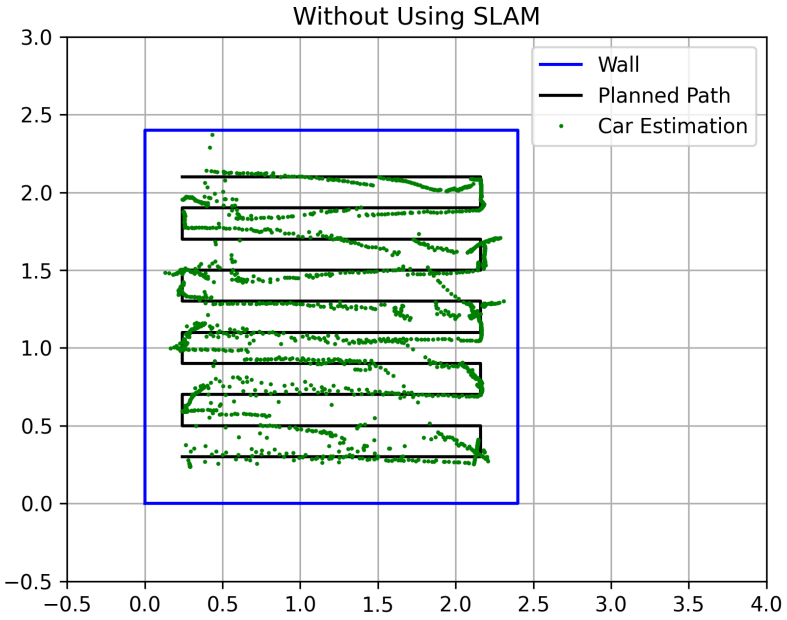

Figure 4: Car Motion

Figure 4 shows the car's position estimations and the planned path. We can see that the car basically follows the planned path. However, due to inaccurate localization, the car's estimated position is not exactly as the planned path.