Name: Sim Justin

Matric No: A0257926N

Project 2: Path Planning

# Initial Conditions

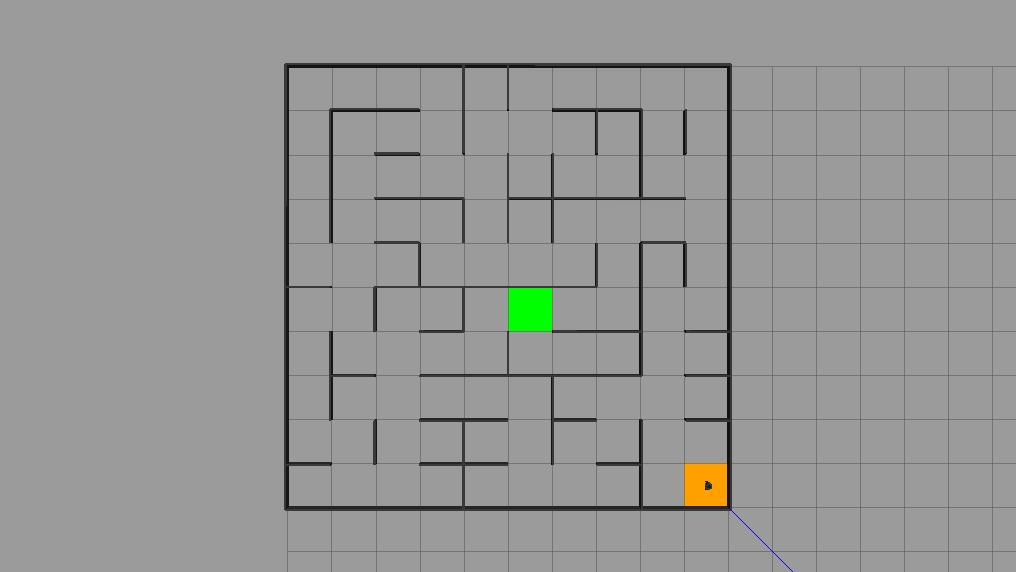


Figure I: Initial Maze Configuration

Starting Position: (0.5, 0.5)

Target Location: (4.5, 4.5)

# Path Planning

## How it Works

The path planning algorithm makes use of a BFS algorithm that explores all potential paths of the same length before exploring paths of a longer length.

This is done by forming a tentative path (stored in a queue) towards the goal with priority on translation along the x-axis followed by that along the y-axis. As the robot traverses along this tentative path, it will identify walls in its surroundings that will prevent it from following this path, updating the path based on this information until it reaches the goal.

This queue becomes empty when either (1) the robot reaches the goal or (2) the robot has navigated throughout all possible paths and found that there is no possible path to the goal.

Poll the Queue

This portion of the code identifies, based on the tentative path, the current cell of the robot and converts this to x, y indices.

Find the path if the goal is found and break the loop

This portion of the code checks if the robot’s current cell is the goal cell. If it is in the goal cell, the robot halts the path planning algorithm and stores the path taken in an array.

Search neighbors and queue them if cheaper

Check if there are walls between the current cell and each neighbor. For each neighbor, check if there are shorter paths to this neighbor and update if so. The tentative path is updated based on this.

## Path Planned

(1), In reverse



(2), In reverese



(3) Final Path Traversed

(0,0), (1,0), (1,1), (2,1), (3,1), (4,1), (5,1), (4,1), (3,1), (2,1), (2,2), (2,3), (2,2), (1,2), (1,3), (0,3), (0,4), (1,4), (2,4), (2,5), (2,6), (2,7), (3,7), (4,7), (4,6), (4,7), (3,7), (3,6), (3,5), (4,5), (4,4)

A screenshot of a game

Description automatically generated

Figure II: Path Traversed

(3) Shortest Path Found

(0,0), (1,0), (1,1), (2,1), (2,2), (1,2), (1,3), (0,3), (0,4), (1,4), (2,4), (2,5), (2,6), (2,7), (3,7), (3,6), (3,5), (4,5), (4,4)

A screenshot of a game

Description automatically generated

Figure III: Shortest Path Found

The paths are different as the path planned initially is tentative and does not account for the presence of the walls. As the robot traverses the maze, it can map out the position of the walls it encounters and replans the path based on this information, resulting in a different path being taken.

# Navigation and PID Control

## Code Added

A computer screen shot of a program code

Description automatically generated

Figure IV: Code Segment for PID

PID control was implemented as per in project 1. Trapezoidal Riemann Sum was used as opposed to Traditional Riemann Sum once again for improved accuracy of integral term.

## Control Gains

A screenshot of a computer screen

Description automatically generated

Figure V: Control Gains Used

The control gains were tuned as per in project 1. It was decided to forgo the integral gain for the angular velocity as the target angle would constantly be changing as the robot is traversing the maze. Including an integral control would result in delays in error correction and inducing excessive overshoot, resulting in instability of the locomotion (and crashing into the wall).

# Performance Enhancement

There are many avenues for the improvement of both the path planning algorithm and the locomotion algorithm. Changes to the locomotion algorithm were implemented and will be discussed in detail while improvements to the path planning algorithm were not carried out and will only be briefly mentioned within this report.

## (1) Locomotion: Heading Correction Priority

Unnecessary additional movements were noticed when making large heading corrections. A condition to slow down the linear velocity, by a factor of 4, while the robot is performing large heading corrections, of above 30 degrees in either direction, was added following the velocity saturation code. It was decided not to stop the robot (i.e. set the linear velocity to 0) to minimize the possibility of skidding which would induce overshooting as well. This was found to result in smoother negotiation of sharp corners, improving the overall locomotion of the robot as it traverses the maze.



Figure VI: Heading Correction Priority

The subsequent changes are discussed briefly as possible improvements but were not implemented.

## (2) Path Planning: Enabling Diagonal Movement

## (3) Path Planning: Heuristics-Based Path Planning

# ROS Structure

1. Initial conditions. Show the initial and assigned destination cells in the maze.
2. Path planning.
   1. Discuss how the path planning works, in particular the lines titled ”Poll the queue” and ”Find the path if the goal is found and break the loop”.
   2. Identify (1) the path planned at the start cell, (2) the path planned at cell O (refer to Figure 1) and (3) the eventual path. Discuss why they are the same or different.
3. Navigation. Discuss how the PID control is implemented, including:
   1. The code that you have added,
   2. The control gains.
4. Performance enhancement. Discuss what you planned to improve, how you did it and the results.
5. ROS structure. Include the graph of ROS nodes and topics of the simulation. Identify and describe the 3 main nodes and topics pertaining to those nodes.