ME 410 Robotics

Project #3, Maze solving robot, due via online meeting with the instructor from 11/9 to 11/13 (time is TBD)

For team projects, please ensure all team members are contributing to the project. There could potentially be a written peer review.

1 Overview

This project is on maze solving robot. Here is an example of a maze solving competition https://youtu.be/76blllun09Q. Here is a wikipedia article on maze solving algorithms https://en.wikipedia.org/wiki/Maze_solving_algorithm. We will focus on simply connected mazes where the robot can reach the target if it follows a wall successfully. Thus, you will first build code needed to follow a wall and demonstrate it in a simple scenario (Stage 1). Then you will use this code to follow a wall in a maze to reach the target (Stage 2). During grading time, besides demonstrating the results for the scenarios in Stage 1 and Stage 2, I may ask you to demonstrate your algorithm on another maze similar to the one in Stage 2.

2 Stage 1: Wall following

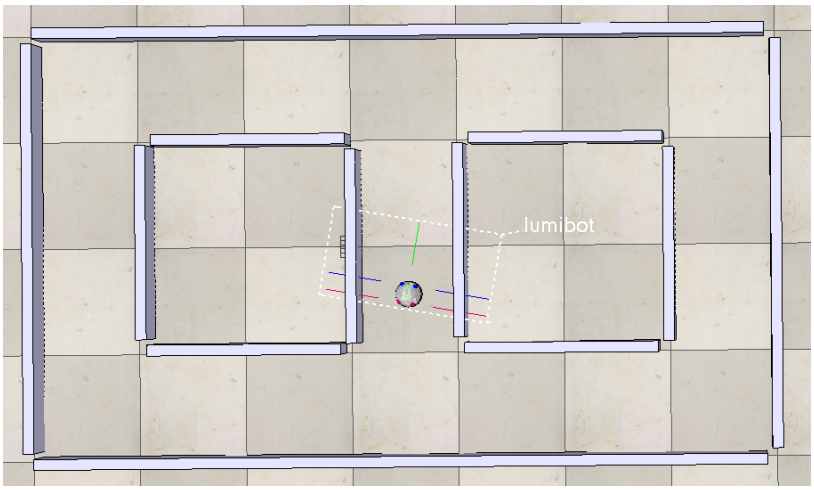

Figure 1: Stage 1: Wall following scene

2.1 Understanding the file given to you

I have modified the lumiBot.ttt from an earlier assignment to do wall following. The file is *lumiBotWall2.ttt*. I have added 5 proximity sensors, two on the left side, two on the right side, and one in front. These have a range of 0.25 m. You will write code to get the Bot to *either* follow the left wall or the right wall in the scenario shown in Fig. 1.

2.2 Stage 1: Specifications

You should demonstrate that the lumiBotWall.ttt or lumiBotWall2.ttt can follow the left OR right wall and complete one circular loop without bumping into any wall.

3 Stage 2: Maze solving

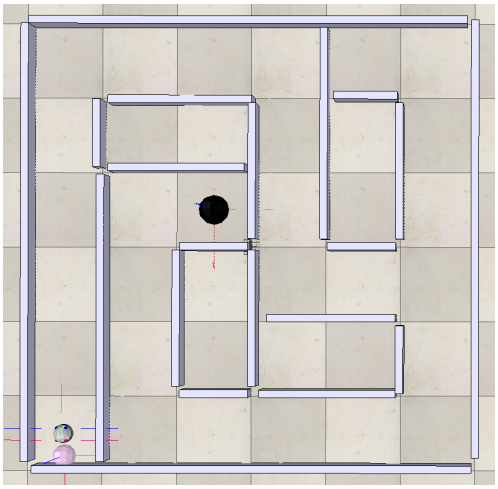

Figure 2: Stage 2: Maze solving scene

3.1 Understanding the file given to you:

I have modified the lumibotWall2.ttt from from Stage 1 to include a maze. The file is *lumiBot-Maze.ttt.*This file has essentially the same lumiBot.ttt and sensors, but with a simply connected maze. See the Fig. 2. Feel free to reuse the code developed from Stage 1.

3.2 Stage 2: Specifications

You should demonstrate the the lumiBot.ttt can go from the start (shown in pink) to the goal (shown in black): (1) without bumping into any wall; and (2) reach the goal in 60 seconds or less (I will use the time shown in the graph/simulator to keep measure the time taken)

4 Some suggestions

I have provided you with a functional code that gets data from the 5 sensors and inputs to the two motors. You only need to work on the logic to follow the left OR right wall.

1. Do not use all 5 sensors. Note that there are two sensors on the left side, two on the right, and one facing front. Use a subset depending on how you will solve the problem. More ambitious students can use all 5 sensors, but this is not required.

2. I suggest using only 2 sensors (at least to begin with): the two on the left side for left wall following OR using the right two sensors for right wall following. Depending on your logic and the performance of the bot, you might need to use the forward facing sensor. You would not need to use all 5 sensors if you are only following one of the two walls.

3. You only need to follow one wall. Choose one among the two throughout this project. More ambitious students can try switching walls to solve the maze faster, but this is more complicated and unnecessary.

4. Try to maintain a certain distance from the wall. You will have to decide what distance is ideal. The maximum distance cannot be greater than the range of the sensor which is 0.25 m and the minimum is defined by your ability to turn without bumping into the walls. You can use the distance from one of the left sensor or the average of the two left sensors to maintain the safe distance from the left wall and similar for the right wall.

5. Try to maintain the car orientation to be along the face of the wall. You can compare the values between the two left sensors to figure out the orientation to the left wall and similar for the right wall.

6. The forward facing sensor is helpful to steer away from the wall in case the heading is too much toward the wall. I recommend not using this sensor initially, but only after you have done thorough testing with the other two sensors.

7. Once you program the wall following scene (Stage 1), copy and paste the code to maze solving part (Stage 2). Modify the code only if it is not working as expected.

5 Grading (100 points as given below)

Grading will be done via zoom meeting with the instructor. Be prepared to show the simulation for the stages described below.

1. Stage 1: 30 points. Demonstrate left or right side wall following for the wall shown in Fig. 1.

2. Stage 2: 30 points. Demonstrate left or right side wall following for the maze in Fig. 2.

3. Stage 2 new course: 30 points. I will give you another simply connected maze like Fig. 2 during the zoom meeting. You can use either left or wall following to complete the course in 60 seconds without bumping into any wall.

4. Email submission: 10 points. The completed CoppeliaSim <filename>.ttt file should have the names of all students in the <filename> and also put author names as a comment inside the Lua script, preferably in the beginning. The .ttt file should be submitted no later than 11/13 via email to pranav@uic.edu.