ENG5009 Advanced Control 5 - Assignment

Introduction

This assignment is presented as a challenge within control engineering.

The first part of the assignment was completed as part of the Lab1 and Lab2 handouts and the results should be included as an appendix to the report for the assignment.

The second stage is to follow the steps below and capture the required data.

The final stage of the assignment is to create a controller based on either fuzzy logic, a neural network, a combination of both, or any other controller you feel is reasonable, that is able to guide a robot to a particular point (within 0.05m radius of the point) while avoiding obstacles.

The method you use is to be presented in a report with results presented and discussed. The report should cover a short description of the controller, the relevant input/output stages, and provide the results. A description of the testing undertaken should also be presented. Improvements to the system developed could also be mentioned.

The report should be a MAXIMUM of 10 pages (not including appendices).

A guide to the marking of the report is provided.

A recommended structure would be:

- Introduction
  - Including the approach taken
- Test Plan
  - Including any test tables
- Results
  - Including figures of the final output
- Discussion
  - Either discuss a successful outcome or discuss any challenges encountered
- Improvements
  - Either new additions or steps for further work
- Conclusions
- Appendices

Provided Resources

All resources that have been used in the labs to date will be available. You are free to use the Fuzzy Logic Toolbox.

Task 1

Develop a controller that is able to drive the robot to a selected point. The position of the robot can be assumed to be accurately known via `state(19)` and `state(20)`.

The first stage of developing this controller is to establish the direction that the robot is required to drive in. To achieve this you can use the following function to work out the required heading:

```
[booleanAtCheckpoint, newHeadingAngle] =
ComputeHeadingAngle( currentLocation, checkpoint, tolerance)
```

in which:
`currentLocation` – the current coordinates of the robot, you can use `state(timeStep,19:20)`
`checkpoint` – the coordinates of the checkpoint the robot is aiming for now
`tolerance` – if the robot is within an *x* meter radius of the target, it is considered accurate enough, you can use 0.05
`booleanAtCheckpoint` – returns 1 if the robot reached the checkpoint and 0 if not
`newHeadingAngle` – the heading angle to move towards the given checkpoint

Once implemented, provide a plot showing the robot driving to the points in the sequence (with no obstacles) provided in Table 1. The starting point is (0,0).

Table 1: Points to travel through

Point	X	Y
1	1	2
2	1.5	1
3	3	-2
4	-1	-2
5	-0.7	-0/7

Task 2

Develop a controller that is able to drive a robot to a known point (x = 1.5m y = 2m) via the points provided in Table 2, while avoiding obstacles.

Table 2: Points to travel through

Point	X	Y
Start	-1	0
1	2	3
2	1	4
3	3	-4
4	-1	-2
5	3	0
End	1.5	2

A map of the area is provided below.

Don’t forget to generate the extra walls using `WallGeneration.m`.

```
[wall_1, obstacleMatrix] = WallGeneration( -4, 0.5, 1, 1, 'h', obstacleMatrix);
[wall_2, obstacleMatrix] = WallGeneration( 2, 2,  -1,   1, 'v', obstacleMatrix);
[wall_3, obstacleMatrix] = WallGeneration( -2.5, -2.5, 2.5, 5, 'v', obstacleMatrix);
```