Introduction to Scientific Computing

Indian Institute of Technology, Madras

Take Home End Semester

Maximum Marks: 100

Assigned: May 14, 2024
Deadline: **May 19**, 2024

General Instructions

- You can use any language to work on the assignment. You are supposed to do a write-up report for both the questions in LaTeX.
- For each question, create a folder called `question_i` in the `home_assn` directory. These folders should contain the relevant files for that `question_i`.
- Then, you are supposed to tar the folder home_assn and upload the tar file on Moodle.
- You are free to read through various resources. However, please ensure that you cite your sources to avoid plagiarism. Any detected instances of plagiarism will result in penalties.
- Please contact your assigned TA for any doubts or queries regarding this assignment.
- The **soft deadline** for this assignment is **11:59 PM** on **May 19**, 2024. Submissions after this deadline will only get linearly decreasing maximum marks.
- The **hard deadline** for this assignment is **11:59 PM** on **May 21**, 2024. Submissions after this deadline will not be evaluated.

[50 marks] 1. **Path planning:**
Context:

- RRT is a sampling-based path planning algorithm used in autonomous robotics systems when the map of the environment is available.
- The idea is to sample random points and add them to the nearest node till a point near the goal is sampled.
- The 'nearness' is defined by you as a tolerance parameter for the algorithm. You should also define a max distance parameter that will be used if the distance from the nearest node to the randomly chosen point is larger than the max distance value and a point at the max distance along the line joining the nearest node and the randomly chosen point.
- If an obstacle is between the nearest node and the randomly chosen point, discard this point and choose another point randomly.

Task:
Your task is to write a program that plots a rectangle of some arbitrary width and height, which acts as the walls of the environment, and place 3-4 obstacles, which are rectangles of around one-tenth the size of the bounding wall. The placement of the obstacles can be random or fixed. Choose two points - start and goal. With this, you are expected to do the following:

[10 marks] (a) Implement the RRT algorithm in this environment.

[10 marks] (b) Since RRT is a random sampling algorithm, the path produced by it is not smooth, which in real life is not a feasible trajectory. So, use the path from the previous part and do trajectory smoothing on it.

[10 marks] (c) RRT also doesn't consider the spacing necessary between obstacles and the path that a bot might follow. So, implement RRT such that the random points are more likely at a spacing away from the edges of the obstacle. This spacing is a parameter you can set as you see fit.

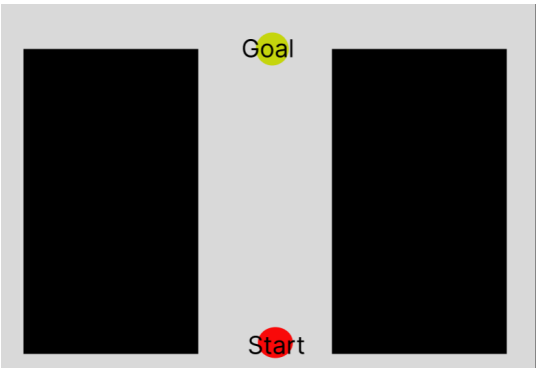

Figure 1: Test environment for Part 3

[10 marks] (d) A greedy approach would be choosing points nearer to the goal more likely. Implement this in RRT algorithm. The nearer here can be any logical distance function such as L1 or L2 norm.

[10 marks] (e) Write a report on your implementation of the previous parts and give plots on the average number of nodes needed to reach the goal - this is done by running the simulation a large number of times and taking the average; a good number would be 1000 for each variant. Setup an environment as shown Figure 1 to check your implementation of RRT from part 3 and write about it in the the report

[50 marks] 2. **Automata :**
Context:

- In this course, you have learned about Regular Expression and may have wondered how a regular expression matching works.
- Basically the regular expression is converted to a Finite state machine (FSM).
- FSM has several states and transitions that define the machine's behavior. It reads an input string one character at a time, and each makes the machine transition. That process is often called consuming a character because each character makes a single change in the machine.

Task:
In this task, you will implement a basic regex engine that converts a regular expression into a basic finite state machine, in which we pass a string and match the expression.

You need to do the following:

[10 marks] (a) Implement a program that converts a regular expression into an FSM. A suggestion would be to use OOPS to build the state machine. machine.

[10 marks] (b) Write a report on your implementation of a regular expression engine and give the finite state machine's state diagram for the below cases.

The FSM should be able to match:

[5 marks] (a) Direct matches

[5 marks] (b) Any number of characters

[10 marks] (c) Wildcard Character

[10 marks] (d) Multiple matches

Input:
Test case 1:
Regular Expression: abcd
Sample string: abcdef

Test case 2:
Regular Expression: a*c*
Sample string: baacc

Test case 3:
Regular Expression: a*
Sample string: baaccaa

Sample Output:
Output 1:
abcd
Output 2:
baacc
Output 3:
baaccaa
[1]

[1]The color of the matched part is up to you to choose.