LAB 4: Path Planning

Due: Thursday, April 14th 11:59 pm

The purpose of lab 4 is to develop pathfinding capabilities for Cozmo. This lab consists of two parts. In the first part, you will implement the RRT algorithm for searching in a simulated environment, and in the next part you will incorporate this with Cozmo to help explore a real environment. You will submit both parts together.

Part I

For this part, you will complete the following methods in the file `rrt.py` :

> `step_from_to`: This method will take two nodes and return the second one if the distance between them is less than the `limit` (a threshold value), or it will return a new node along the same line but `limit` distance away from the first node.
>
> `node_generator`: This method will return a randomly generated node, uniformly distributed within the map boundaries while avoiding obstacles.
>
> `RRT`: This is the main method for which we have provided a framework. You must complete the main loop by generating random nodes and assembling them into a tree in accordance with the algorithm. Goal detection, tracking parents, and generating the final path is already done for you as part of the `cmap.py` method.

Note 1: The file `utils.py` contains some general useful methods, as well as the definition of the `Node` class, and the file `cmap.py` contains some useful methods related to the map representation, so it is worth familiarizing yourself with these files though you will not be editing them.

Note 2: We have provided you with 3 maps (in the `maps` folder) for testing purposes. You can run the algorithm either on one map with a graphical visualizer by executing `rrt.py` (you can change the map in the main method at the bottom of the file), or you can run the code on all 3 maps at once without the visualizer by executing `python autograder.py gradecase1.json`.

Evaluation: The first part will be auto-graded using the 3 maps we have given you, as well as 3 new maps (similar in layout to the one we have given you). You will receive 10 points per successful solution, for a total of 60 points (the first item in the evaluation table).

Part II

In this part, you will use the RRT implementation and use it to help Cozmo navigate its environment. We have given you an `rrt_robot.py` which includes a framework for running Cozmo alongside the visualizer. You can either copy your RRT function (from Part I) into it or copy the Cozmo functions into your `rrt.py` file. For this part, you should complete the followings:

`CozmoPlanning`: This method is executed by the RobotThread and should contain all your Cozmo behavior. The main objectives of this behavior include: (1) identifying a target cube, (2) using RRT to find a path to a specific face of the cube, (3) following the path found by RRT, and (4) replanting to avoid any obstacle cubes that are added during navigation. During the demo, obstacle cubes can be added at any time as the behavior function is running. The map should be updated accordingly to reflect any new cubes that Cozmo sees, however, we will not move the cubes once they are placed so you don't need to account for that.

Note 3: You will need to find a way to retrieve the path from RRT yourself as part of the path following. If the target cube is not visible from Cozmo's starting location, Cozmo should navigate to the center of the arena to look for it and navigate to it once it is seen. To run Cozmo's behavior simply execute the `rrt_robot.py` file.

Note 4: In this lab, you are not allowed to use high-level functions such as `go_to_pose` as it would largely defeat the purpose of this course. For robot's movements, you can pick one of these options:

- Use only `drive_wheels` (During lab3 some groups could not update Cozmo's pose which can lead to issues detecting cube positions correctly.)
- Use a combination of `drive_straight` and `turn_in_place`.

Note 5: We do not accept a combination of the above-mentioned three functions since it is not scientifically correct.

Note 6: To help identify specific cubes, the snippet `light_cubes[cozmo.objects.LightCube#Id].object_id` can be used to lookup matching object ids, where # is the number of a cube (1-3, it can be found in one of the hollow areas next to the battery screw).

Evaluation: Your robot behavior will be ~~demoed in class~~ executed and checked by the TAs, and your RRT implementation will be autograded based on your submission as described in part I. The rubric is as follows:

RRT implementation, autograded with 6 maps, 10 points per solved map	60 pts
The robot follows the path found by RRT	15 pts
The robot identifies a target cube and navigates to a specific face	10 pts
The robot replans to avoid obstacle cubes	10 pts
The map is updated to reflect newly seen cubes	5 pts

Submission: By 11:59pm on Thursday, April 14th 2020, submit both your final `rrt.py` and `rrt_robot.py` files as a single zip file on Blackboard. Make sure your code is entirely contained within these files. If you relied significantly on any external resources to complete the lab, please reference them in the submission comments.

~~**Note 7:** The demo will take place in class on the day the assignment is due. Please make sure to bring your Cozmo, phone, USB cable, and laptop as well as any chargers that may be required on this day~~. No in-class demo for COMP4500 - Spring 2020.