RBE 550 - Standard Search Algorithms Implementation

Overview

In this assignment, you are going to implement **RRT** and **RRT*** algorithms. This template is provided to you as a starting point. After you finish coding, you would be able to run these algorithms to find a path in a map, and visualize the result.

Files included:

RRT.py is the file where you will implement a RRT class for RRT and RRT*.

main.py is the script that provides helper functions that load the map from an image and call the classes and functions from **RRT.py**. You are not required to modify anything but you are encouraged to understand the code.

WPI_map.jpg is a binary WPI map image with school buildings. You could replace it with some other maps you prefer.

Instruction

Before starting any coding, please run the code first:

```
python main.py
```

The **main.py** loads the map image **WPI_map.jpg** and calls classes and functions to run planning tasks. As you haven't written anything yet, there should be no path shown in the graph, but only the map image, start point and end point.

Please keep in mind that, the coordinate system used here is **[row, col]**, which is different from [x, y] in Cartesian coordinates. In README and the code comment, when the word '**point**' is used, it refers to a simple list [row, col]. When the word '**node**' or '**vertex**' is used, it refers to the Node class in RRT.

RRT

For simplicity, this template uses a class 'Node' and a list 'vertices' in class 'RRT' as a tree structure. If you prefer to use other tree structure, please feel free to do so.

You would code RRT in the function `RRT`. In each step, get a new point, get its nearest node, extend the node and check collision to decide whether to add or drop this node. When you add a new node to the tree, remember to set the cost and parent of the new node, and add the new node to the list 'vertices'. You will also need to check if it reaches the neighbor region of the goal. If so, connect to the goal directly and set the found flag to be true.

You would code RRT* in the function `RRT_star`. The first few steps are pretty much the same as RRT. Besides, when a new node is added, you will need to rewire the new node AND all its neighbor nodes. Even a path is found, the algorithm should not stop as adding new nodes will possibly optimize the current found path.

Read the description of the functions for more details before implementing.

Until now, I hope you have a basic understanding of the template code and what to do next.

This template is only provided as a start point, feel free to make any modification of the codes or code structures if needed. After you finish coding, your algorithms should produce similar results as the images in **demo** folder.

Rubrics

- (5 pts) Your RRT and RRT* are implemented correctly
    - Get proper new nodes in each step
    - Connect and rewire (RRT*) new nodes
    - Find a path if feasible

- (2 pts) Documentation

  Besides the code, you should also include a documentation with the following content:

  - Briefly answer the following questions
  - For RRT, what is the main difference between RRT and RRT*? What change does it make in terms of the efficiency of the algorithms and optimality of the search result?
  - Compare RRT with the results obtained with PRM in the previous assignment. What are the advantages and disadvantages?
  - Algorithm results and explanation

  Run your code with `python main.py` and **save your results as png images**. Briefly explain why your algorithms would produce these results. Why RRT and RRT* result in different trees? How different sampling methods lead to different sample sets in the graph?

  - Reference paper and resources if any

  Include the documentation as a pdf file, or if you prefer, a md file.