HW 6 – Experimental analysis of RRT algorithms

Due: 2018.11.05 End of day

G. Fainekos – Intro to Robotics – Fall 2018

We have discussed in class that as the complexity of the workspace and the robot dynamics increases, solving the motion planning problem becomes a much harder problem. Thus, we introduced probabilistic roadmap methods which can provide good solutions to difficult motion planning problems. In certain cases, we also introduced different tricks and heuristics to improve the performance of such methods.

For this homework problem, you will perform a small scale experimental analysis of the RRT algorithm. You will be provided with a basic version of the RRT algorithm with collision detection (the version provided with the Robotics Toolbox does not support collision detection bug-free).

You will perform the following experiments:

A. Assess the probability of success of the RRT algorithm for a given environment when the budget is 1000 nodes. Success is defined as the RRT producing a tree node within distance 1 from the goal position on the x-y plane (we ignore the orientation of the robot in this analysis). The script `hw6_A_prob_of_success_data.m` creates the data for a specific scenario. If you run the script, it will display the environment, an RRT with 1000 nodes and the best path found.
B. Assess the performance of the basic RRT algorithm under different number of obstacles of the same size (the basic obstacle provided as input). Given an array with increasing number of obstacles [n_1 n_2 … n_m], where $n_i<n_j$ for $i<j$,
    a. for each number of obstacles n_i create an environment with randomly placed n_i obstacles of the same structure, a random start point and a random goal point
    b. run a number of tests of statistical significance in order to assess the performance of the algorithm for a fixed computation budget
    c. return the success ratio for each scenario along with the specific scenarios tested.
C. The RRTmap class can also receive as input the goal position which is currently unused for guiding the search. Modify the RRTmap plan method to bias the search towards the goal position. Compare the performance of the basic RRTmap with your modified version. Write a criterion for automatically establishing which version is better.

Files and templates provided
The following files are provided:

1. **`RRTmap.m`** is the class that implements RRT with obstacle collision checking for bicycle models. This is a class based on the RRT class of the Robotics Toolbox with some bugs fixed.
2. **`Polygon2OccMap.m`** is a function to translate polygonal environments to occupancy grids. You do not need to modify this function.
3. **`hw6_A_prob_of_success_data.m`** creates the data for a specific scenario. It calls the template **`hw6_prob_of_success.m`** which you are going to modify with your code.
4. **`hw6_scalability_data.m`** creates some preliminary data for your function to create artificial benchmarks and test the basic RRT algorithm. It calls the template **`hw6_scalability.m`** which you are going to modify with your code.
5. **`hw6_comparison_data.m`** creates some test scenarios for the comparison of the two RRT algorithms. It calls the template **`hw6_comparison.m`** which you are going to modify with

your code. The function **`hw6_comparison.m`** will also call your implementation of the goal guided RRT under the name **`RRTgoal.m`**.

Grading
Max points: 100
Criteria:

- Experiment 1: 70 pt
- Experiment 2: 20 pt
- Experiment 3: 10 pt

Bonus points:

- +5 : If you create an RRT algorithm that always outperforms the basic RRT version

Penalties:

- -3 per file if you do not update your information at the top of the templates

Notes/Remarks

- **Make sure that you update the Matlab paths before you start the homework assignment. This assignment depends on correctly computing differences between angles.**
- Make sure that you observe the interface requirements.
- Make sure that if you submit the modified RRT class, then it can be executed under the required naming convention **`RRTgoal.m`**.
- You should check whether the start or the goal positions are inside an obstacle. Notice that cases where the start and/or goal positions are inside an obstacle can be meaningful only for testing correctness of your implementation and not for comparing performance between different RRT algorithms.
- For each experiment, you are allowed to call the function that you have developed for any of the previous experiments.

You will be using the following classes:

- RRTmap: constructs an RRT and returns paths on the RRT
- PGraph : constructs graph which are topologically embedded. The latter means that the nodes have topological information regarding their coordinates and the edges are labeled with weights representing distances between nodes.
- Polygon : construct and manipulate polygons.

Deliverables
You will be submitting 4 Matlab files in a zip file named **`hw6_files_ASUAD.zip`**, where ASUAD is your ASU username. The zip file should contain:

1. **`hw6_prob_of_success.m`**
2. **`hw6_scalability.m`**
3. **`hw6_comparison.m`**
4. **`RRTgoal.m`**