

## COM S 476/576 Midterm Exam

**Problem 1 [20 points for 476, 15 points for 576]:** We revisit the two-link robot described in Homework 3. Consider a robot consisting of 2 links,  $\mathcal{A}_1$  and  $\mathcal{A}_2$ . Each link has width  $W$  and length  $L$ . The distance between the two points of attachment is  $D$ .  $\mathcal{A}_2$  is attached to  $\mathcal{A}_1$  while  $\mathcal{A}_1$  is attached to the origin. Each link is allowed to rotate about its point of attachment. The configuration of the robot is expressed with 2 angles  $(\theta_1, \theta_2)$ , where  $\theta_1, \theta_2 \in [-\pi, \pi)$ . The first angle,  $\theta_1$ , represents the angle between the segment drawn between the two points of attachment of  $\mathcal{A}_1$  and the  $x$ -axis. The second angle,  $\theta_2$ , represents the angle between  $\mathcal{A}_2$  and  $\mathcal{A}_1$  ( $\theta_2 = 0$  when they are parallel).

The world is  $\mathcal{W} = \mathbb{R}^2$ . The obstacle region  $\mathcal{O} \subset \mathcal{W}$ , the link's parameters, and the initial and goal configurations are described in a json file, which contains the following fields.

- "O": a list  $[\mathcal{O}_1, \dots, \mathcal{O}_n]$ , where  $\mathcal{O}_i$  is a list  $[(x_{i,0}, y_{i,0}), \dots, (x_{i,m}, y_{i,m})]$  of coordinates of the vertices of the  $i^{th}$  obstacle.
- "W": the width of each link.
- "L": the length of each link.
- "D": the distance between the two points of attachment on each link
- "xI": a list  $[\theta_1, \theta_2]$  specifying the initial configuration  $x_I = (\theta_1, \theta_2) \in [-\pi, \pi) \times [-\pi, \pi)$ , and
- "xG": a list  $[\theta_1, \theta_2]$  specifying the goal configuration  $x_G = (\theta_1, \theta_2) \in [-\pi, \pi) \times [-\pi, \pi)$ .

Note that all of the above are the same as in Homework 3, except that there is a single goal configuration (as opposed to a list of goal configurations as in Homework 3) and  $x_I$  and  $x_G$  are configurations  $(\theta_1, \theta_2)$ , each specified in radian (as opposed to a grid cell as in Homework 3).

**Task (Solve the planning problem using RRT):** Instead of discretizing the C-space as in Homework 3, use the single-tree search outlined in Section 5.5.3 to compute a path in C-space from  $x_I$  to  $x_G$ . You should check periodically if the tree can be connected to  $x_G$ . This can be easily done by setting  $\alpha(i)$  as  $x_G$  with a certain probability  $p$ . For example, the book recommends  $p = 0.01$ . You should have this as a parameter in your code. Once  $x_G$  is successfully added to the tree, quit the loop and compute the path from  $x_I$  to  $x_G$ .

1. Plot both the resulting tree and path with the x-axis and y-axis corresponding to  $\theta_1$  and  $\theta_2$ , respectively. The result should be similar to Figure 1, which uses  $p = 0.1$ .
2. Generate a json file containing the following fields:

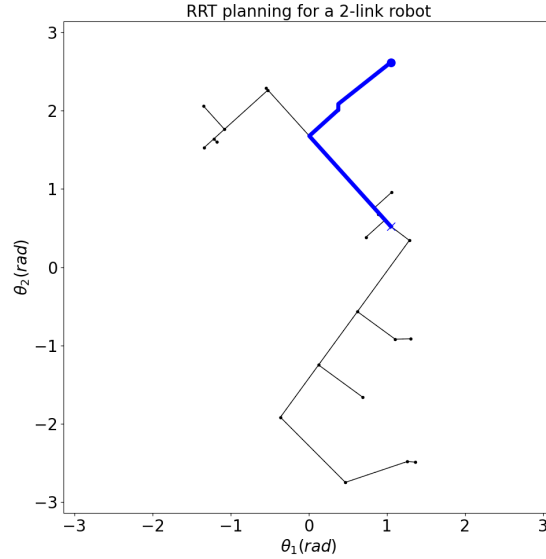


Figure 1: The result of RRT planning with  $p = 0.1$ , showing the tree and the path from  $x_I$  to  $x_G$ .

- **"vertices"**: the vertices in the graph, represented by a list of dictionaries with 2 keys: "id" and "config" such that the value of "id" is the id of the vertex and the value of "config" is the corresponding configuration of the robot. For example, "vertices": [{"id": 0, "config": [1.0, 0.5]}, {"id": 1, "config": [1.0, 1.0]}] means that there are 2 vertices. The id of the first vertex is 0 and its corresponding configuration is (1.0, 0.5). The id of the second vertex is 1 and its corresponding configuration is (1.0, 1.0).
- **"edges"**: the edges in the graph, represented by a list of  $[v_1, v_2]$ 's where  $v_1$  and  $v_2$  are the id of the origin vertex and the destination vertex, respectively. For example, "edges": [[2, 1], [2, 3]] means that there are an edge from a vertex with "id": 2 to a vertex with "id": 1 and an edge from a vertex with "id": 2 to a vertex with "id": 3 .
- **"path"**: the list of id's of the vertices specifying the path from  $x_I$  to  $x_G$ .

For example, if your code is `midterm.py`, running

```
python midterm.py midterm_world.json --out midterm_out.json
```

should output `midterm_out.json`, containing "vertices", "edges", and "path" for the problem described in `midterm_world.json`. Example of `midterm_out.json` and `midterm_world.json` can be found on Canvas.

**Submission:** Please submit a single zip file on Canvas containing the followings

- your code (with comments, explaining clearly what each function/class is doing), and
- a text file explaining clearly how to compile and run your code, including a list of necessary external libraries.

**Problem 2 [2 bonus points for 476, 5 points for 576]:** Suppose you are a reviewer of a conference paper on motion planning. The paper formulates a feasible motion planning problem in the following definition. Write down your feedback to the author on this definition. Is the problem correctly formulated? If not, point out the issues and offer some suggestion how to fix them.

**Definition 1 (A feasible motion planning problem)** *Let  $\mathcal{C}$  and  $\mathcal{C}_{obs}$  denote the configuration space and the configuration space obstacle, respectively. Additionally, let  $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$  denote the free space. For an initial configuration  $q_I \in \mathcal{C}_{free}$  and goal configuration  $q_G \in \mathcal{C}_{free}$ , a feasible motion planning problem attempts to find a feasible path  $\tau : [q_I, q_G]^T$  such that  $\tau$  is continuous and  $\tau \in \mathcal{C}_{free}$ . If such a feasible path does not exist, a motion planning algorithm should return failure.*