COM S 476/576 Homework 3: C-Space and C-Space Obstacles

This project is an extension of Homework 1 and 2, with the special case of $m = 2$.

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 (θ_1, θ_2), where $\theta_1, \theta_2 \in [-180°, 180°)$. The first angle, θ_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, θ_2, represents the angle between $\mathcal{A}_2$ and $\mathcal{A}_1$ ($\theta_2 = 0$ when they are parallel).

To turn the motion planning problem for the robot to a discrete planning problem, we discretize the C-space into 1-degree by 1-degree grid. So we'll have 360×360 grid, centered at $\{(i, j) \in \mathbb{Z} \times \mathbb{Z} \mid -180 \leq i < 180, -180 \leq j < 180\}$. The center of each grid cell represents a configuration (in degree). For example, the grid cell centered at $(0, 0)$ represents configuration $(0, 0)$, which corresponds to a configuration in which the two links lay flat horizontally, pointing to the right. With this discretization, we can define the configuration space for the robot motion planning problem as

$$\mathcal{C} = \{(i, j) \in \mathbb{Z} \times \mathbb{Z} \mid -180 \leq i < 180, -180 \leq j < 180\} \tag{1}$$

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 $[\mathcal{O}_1, \ldots, \mathcal{O}_n]$, where $\mathcal{O}_i$ is a list $[(x_{i,0}, y_{i,0}), \ldots, (x_{i,m}, y_{i,m})]$ 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 $[i, j]$ specifying the initial configuration $x_I = (i, j) \in \mathcal{C}$, and
- `"XG"`: a list of $[i, j]$'s, each corresponding to a goal configuration $x_G \in \mathcal{C}$.

Task 1 (C-space obstacles) [15 points for 476, 10 points for 576]: Compute the C-space obstacles $\mathcal{C}_{obs} \subseteq \mathcal{C}$, where $\mathcal{C}$ is defined in (1), for the robot based on the given obstacle region $\mathcal{O}$. In particular, implement the function `compute_Cobs(O, W, L, D)` in `hw3.py`, which can be found in the course repo. Here, `O` is a list of obstacles such that for each `i`, `O[i]` is a list `[(x_0, y_0), ..., (x_m, y_m)]` of coordinates of the vertices of the `i`th obstacle. `W`, `L`, and `D` are float that correspond to the width and length of each link and the distance between the two points of attachment on each link.

This function should return a list `Cobs` such that for each `i`, `Cobs[i]` is a configuration q of the robot such that $\mathcal{A}(q) \cap \mathcal{O} \neq \emptyset$.

Hint: For each of the 360×360 grid cells, you need to compute if the robot at the corresponding configuration is in collision with the obstacles. So you will need a function to detect the collision. If it is in collision, add the corresponding configuration to `Cobs`. Feel free to use any external library to check whether 2 rectangles overlap.

Task 2 (Free space) [5 points]: Compute the free space $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$, where $\mathcal{C}$ is defined in (1) and $\mathcal{C}_{obs}$ is computed in the previous task. In particular, implement the function `compute_Cfree(Cobs)` in `hw3.py`. Here, `Cobs` is the same object returned from `compute_Cobs(O, W, L, D)` in the previous task.

This function should return an instance of `Grid2DStates` class from Homework 1.

Once both tasks are completed, you should be able to run

```
python hw3.py hw3_world.json --out hw3_out.json
```

which generates a plot shown in Figure 1 as well as `hw3_out.json`, containing the following fields:

- `"Cobs"`: a list of configurations in $\mathcal{C}_{obs}$
- `"path"`: the list of cells specifying the path from x_I to X_G.

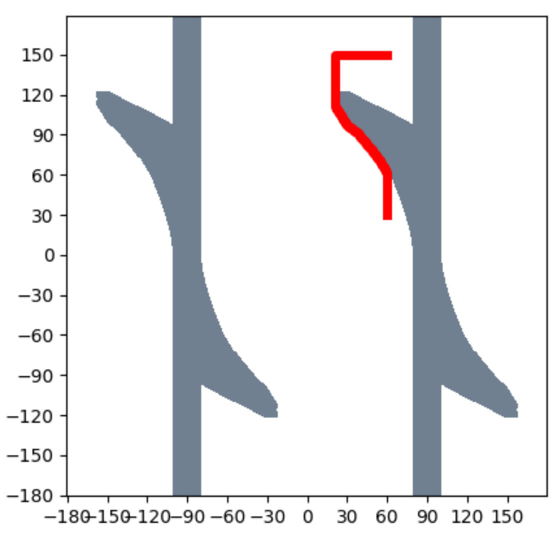

Figure 1: The plot generated after running `hw3.py` with `hw3_world.json`.

Example of `hw3_out.json` and `hw3_world.json` can be found on the course github repo.

Task 3 (Collision checking) [2 bonus points for 476, 5 points for 576]: Given a path as a list of configurations (i.e., the output of Task 2), construct a finer discretization of the path such that each θ_i does not increase more than 0.1 degree in 1 step, assuming linear interpolation between 2 consecutive configurations on the path. For example, suppose `path = [(0,0), (1,1), (1,2)]`. A finer discretization can be `fpath = [(0,0), (0.1, 0.1), (0.2, 0.2), ..., (1,1), (1, 1.1), (1, 1.2), ..., (1,2)]`.

Then, at each configuration in `fpath`, check whether the robot is in collision with the obstacle. Print out the list of configurations in `fpath` where the robot is in collision. You can simply add the printout to the code you run in Task 2.

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.