Homework 3

Intro to Robotics

Submission Instruction: Create a directory called HW3. Then create subdirectories called Q1, Q2 and so on. Place the answers of each question in the corresponding directories. Finally, zip the HW3 directory and submit.

1. (20 pts) Compute the Jacobian Matrix (formula with variables) of the robot defined by the following DH parameters

Link	a_i	α_i	d_i	θ_i
1	0	0	d_1	θ_1^*
2	0	-90	d_2^*	0
3	0	0	d_3^*	0

* variable

For the following programming questions, you should use Python.

2. (30 pts)

a. (10 pts) Program a function to compute the forward kinematic matrix based on UR5's DH parameters (Refer to Homework 2). You should have the following function:

A=FK(DH)

-- DH contains all the DH parameters of UR5

You should be able to find the DH parameters in the lecture slides.

b. (10 pts) Randomly generate 10 sets of join angles from the range of -60 to 60 degrees. Compute 10 A's for the 10 sets of join angles using your FK function.

c. (10 pts) Create a simulation in CoppeliaSim with a UR5 robot and a vacuum cup (as shown Figure 1) mounted on the wrist. Position the UR5 robot in the simulation based on the coordinate system 0. Manually set the robot joins to the 10 sets of join angles in 2.b. and record the vacuum cup position and orientation. Convert the positions and orientations into the rotation matrices and then compare them with the 10 A's you obtained in 2.b. Report your results in a table.

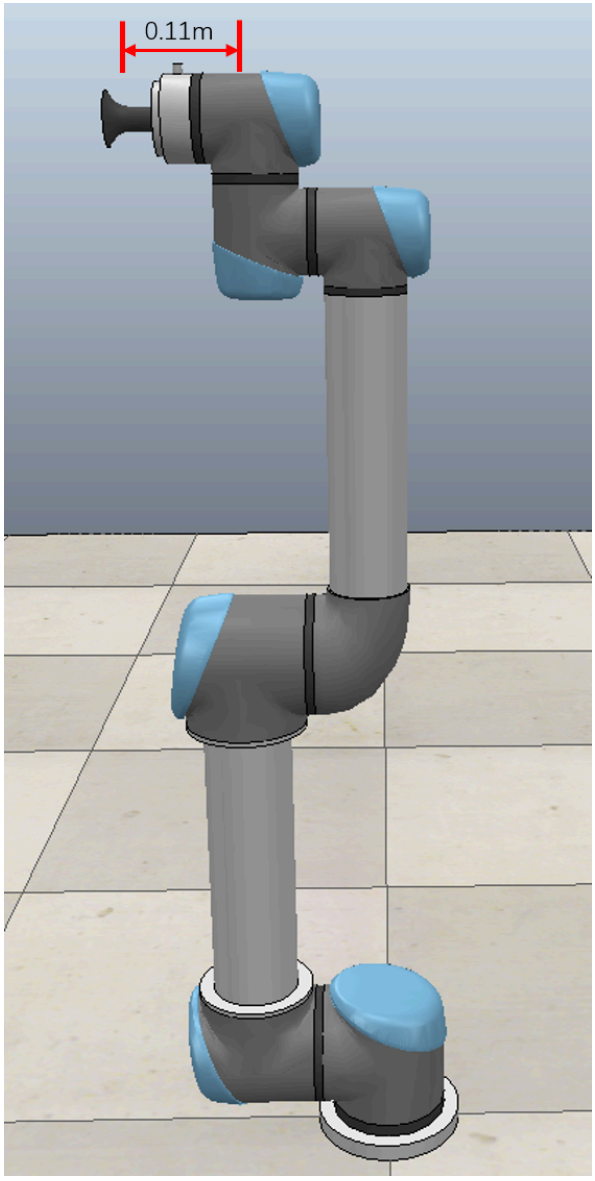

Figure 1.

3. (20 pts) Your program should randomly generate 100 poses of UR5. Each pose θ vector should be (θ_1, θ_2, θ_3, θ_4, θ_5, θ_6). Your program should check if any of the pose collide with the robot itself, table, and boxes or not. Output the ones are in the free space (not collide with anything). The set of poses in the free space is F.

4. (30 pts) Implement wavefront planner to generate a motion path in a 2D environment from a start point to a goal point without running into any obstacle:

```
yourpath = pathplanning(S, G, n, O)
```

The inputs of your function are

Start: S = (x, y)
Goal: G = (x, y)
Obstacle number: n
Obstacle coordinates: Oi = [x1 y1; x2 y2] and i =1 to n.
(All obstacles are rectangles)

The output of your function should be a serial of key points (milestones) that represent the path.

You should provide a test script to test your function.