Perception and Decision Making in Intelligent Systems
Homework 3: A Robot Manipulation Framework
Announcement: 10/22, Deadline: 11/11 23:59

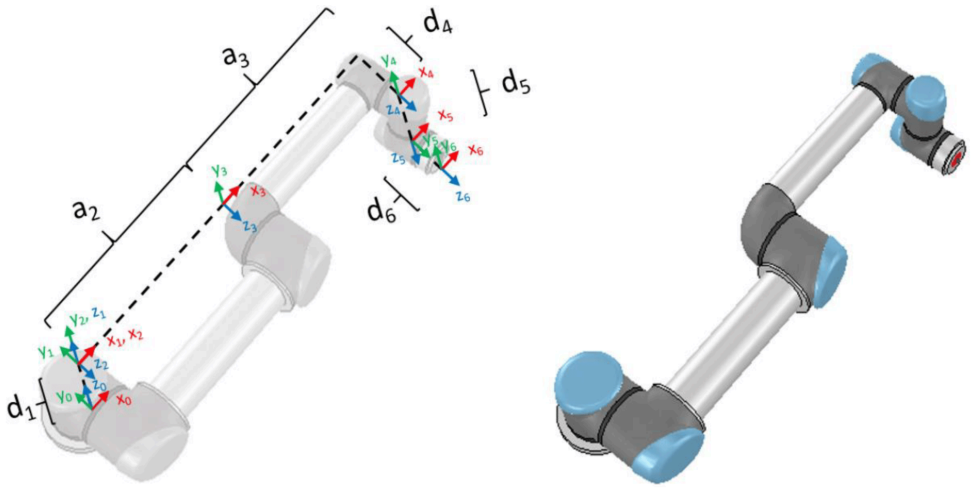
Introduction

In this homework, you are required to implement the forward kinematics (FK) ,./l;and inverse kinematics (IK) functions for a 6-degree-of-freedom robot arm. Besides, you need to answer questions about the robot manipulation framework developed in this homework.

There are three main tasks:

- Implement the forward kinematics function.
- Implement the inverse kinematics function.
- Using your IK implementation in the Transporter Network block insertion task.

We will use the PyBullet simulator for this homework. The robot arm used in this homework is ur5, which is a 6-degree-of-freedom robot arm (containing 6 revolute joints).



The 6-degree-of-freedom robot arm we will use in this homework

Requirements

The requirement of the development environments:

- OS : ubuntu 18.04, 20.04
- Python 3.7 (You can use conda to create a new environment)

Implementation

We will provide a project template of the robot manipulation framework. **Please complete this homework based on this template**.

For task 1 and task 2, you are required to write code to complete the functions that we have defined in the project template and make them output the expected results.

For task 3, you need to test the Transporter Network on the block insertion task meanwhile using your IK implementation.

The file structure of the project template:

```
hw3
 ├ pybullet_planning/ # Motion planning algorithms
 ├ pybullet_robot_envs/ # contains all the important information of the robot arm
 ├ ravens/ # contains the environment you need for task 3
 ├ test_case/ # contains the public test cases to help you verify your code
 ├ hw3_utils/ # contains useful functions you may use
 ├ ik.py # contains the function that you need to implement for task 1
 ├ fk.py # contains the function that you need to implement for task 2
 └ README.md
```

Please read the following instructions in README.md **CAREFULLY**.

[Important]
Please check all the **"TODO"** comments in **fk.py, ik.py, ravens/ravens/environments/environments.py**

Task 1: Forward kinematics function

The input and output of function **your_fk()** in **fk.py**:

- Input:
  - A set of joint states (the rotation angle of the robot arm) which is a 6D array ranging from $-\pi$ to π
  - The D-H parameters (following **classic convention**) of the robot arm which is a dictionary structure. We have already provided it in **fk.py**.
- Output:
  - The corresponding 7D pose of the robot end-effector:
    - 3D position: x, y, z in the world coordinate
    - 4D rotation: quaternion in (x, y, z, w) format
  - The corresponding Jacobian matrix in $R^{6 \times 6}$

You need to modify **your_fk()** function in **fk.py** and make it output the expected results. **You cannot use any PyBullet or third-party functions to directly output the results. You will not get any points in this part if you do not follow the policy**. Of course, the libraries about mathematical computing or matrix operations are allowed to use (e.g. numpy, quaternion, numba, scipy.spatial.transform …).

To verify the correctness of your implementation, you can use a subset of testing cases to check the corresponding scores. We will use the same scoring function to verify your results (import **your_fk()** in our scoring scripts).

The D-H table in the official specification is slightly different in this homework. Please follow the D-H parameters we provided in **fk.py** or you may fail to implement this function.

Hint: Prior Knowledge

- D-H parameters for classic convention
- geometric Jacobian in robotics

Task 2: Inverse kinematics function implementation

The input and output of this function **your_ik()** in **ik.py**:

- Input:
  - Target 7D pose of the robot arm's end-effector
    - 3D position: x, y, z in the world coordinate
    - 4D rotation: quaternion in (x, y, z, w) format
  - Current 6D joint parameters ranging from -π to π
    - 6D: for the revolute joints
- Output:
  - The expected 6D joint parameters for the target 7D pose of the robot arm's end-effector
    - 6D: for the revolute joints (will be different from the input)

In this task, you are required to implement the **iterative inverse kinematics (IIK)** using the Jacobian method (please check this slide). You need to implement the **pseudo-inverse method** to compute the desired joint parameters for the target robot arm's end-effector pose. Of course, it is welcome to implement other IK methods you like, you can compare the results with the pseudo-inverse method and observe the differences between them. You may get some extra points if you also try other methods to implement the IK function.

Similar to part 1, You need to modify **your_ik()** function in **ik.py** and make it output the expected results. **You cannot use any PyBullet or third-party functions to directly output the results. You will not get any points in this part if you do not follow the policy.** Besides, you can use a subset of testing cases to check the corresponding scores.

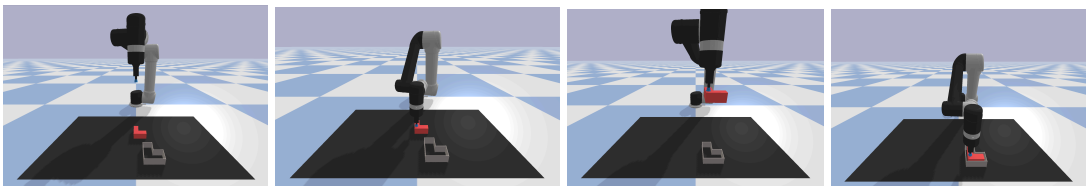
Hint: You may need to implement **your_fk()** function in this task.

Task 3: Transport Network manipulation pipeline

After finishing the previous two tasks, the robot arm can move autonomously! In this homework, we will go through a **block insertion task** where the robot will grasp a block and insert it into a fixture using a method called Transporter Network.

The manipulation framework contains 4 sub-tasks:

1. **to prepick:** move the robot arm to a prepick pose (above the block)
2. **grasping:** move the robot arm downward to grasp the block using suction cup
3. **to preplace:** move the robot arm to a preplace pose (above the fixture)
4. **insertion:** insert the block into the fixture



to prepick | grasping | to preplace | insertion

In this part we have provide you a model checkpoint for block insertion task and 10 test case. All you have to do is download them and run the testing code (see the readme file for instructions).

If your IK is correct you will be able to finish the task in all 10 trials.

References

- PyBullet quickstart guide: https://docs.google.com/document/d/10sXEhzFRSnvFcl3XxNGhnD4N2SedqwdAvK3dsihxVUA/edit#heading=h.2ye70wns7io3
- D-H parameters of ur5 robot: https://www.universal-robots.com/articles/ur/application-installation/dh-parameters-for-calculations-of-kinematics-and-dynamics/
- Inverse Kinematics using Jacobian methods: https://homes.cs.washington.edu/~todorov/courses/cseP590/06_JacobianMethods.pdf
- Transporter Networks: Rearranging the Visual World for Robotic Manipulation: https://transporternets.github.io

Report

Please answer the following questions in your report:

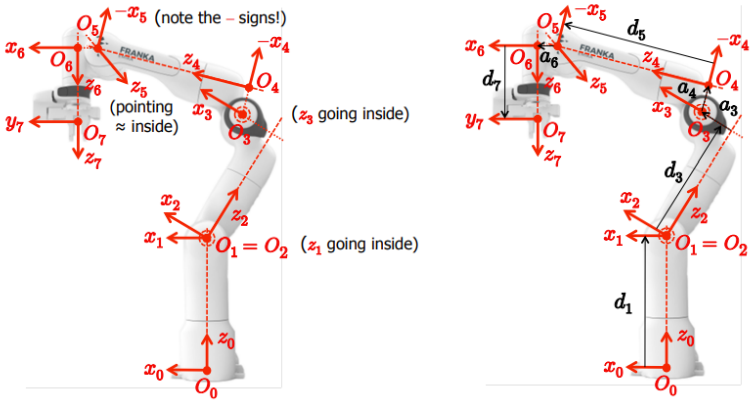
1. About task 1 (15%)

1.1 Briefly explain how you implement **your_fk()** function **(3%)**

- (You can paste the screenshot of your code and explain it)

1.2 What is the difference between D-H convention and Craig's convention (Modified D-H Conveition)? **(2%)**

1.3 Complete the D-H table in your report following **D-H convention (10%)**



The coordinate frames of the robot arm in this homework following D-H convention

i	d	α (rad)	a	θ_i (rad)
1				θ_1
2				θ_2
3				θ_3
4				θ_4
5				θ_5
6				θ_6
7				θ_7

A D-H table example format (please fill in it in your report)

2. About task 2 (10% + 5% bonus)

2.1 Briefly explain how you implement **your_ik()** function **(5%)**

- (You can paste the screenshot of your code and explain it)

2.2 What problems do you encounter and how do you deal with them? **(5%)**

2.3 Bonus! Do you also implement other IK methods instead of pseudo-inverse method? How about the results? **(5% bonus)**

3. **About task 3 (5%)**
   This part uses the **your_ik()** function to control the robot and complete the block insertion task.
   3.1 Compare your results between your_ik function and pybullet_ik

Grading Policy

1. **Correctness verification using test cases 70%**
   - task 1: 20%
   - task 2: 40%
   - task 3: 10%
2. **Report 30% + bonus 5%**
   - about **task 1**: 15%
   - about **task 2**: 10% + 5% bonus
   - about **task 3**: 5%

Submission

Due Date: 2024/11/11 23:59

Please directly compress your code files and report (.pdf) into {STUDENT ID}_hw3.zip and submit it to the New E3 System.

The file structure should look like:

```
📂{student_id}_hw3.zip
  ├ 📜 fk.py # contains your implementation of your_fk()
  ├ 📜 ik.py # contains your implementation of your_ik()
  ├ 📜 report.pdf
  └ 📜 other files or folders # only needed if you modify some files in other folders
```

Wrong submission format leads to -10 point
Late submission leads to -20 points per day