Perception and Decision Making in Intelligent Systems
Homework 4: A Robot Manipulation Framework
Announcement: 12/6, Deadline: 12/27 23:55

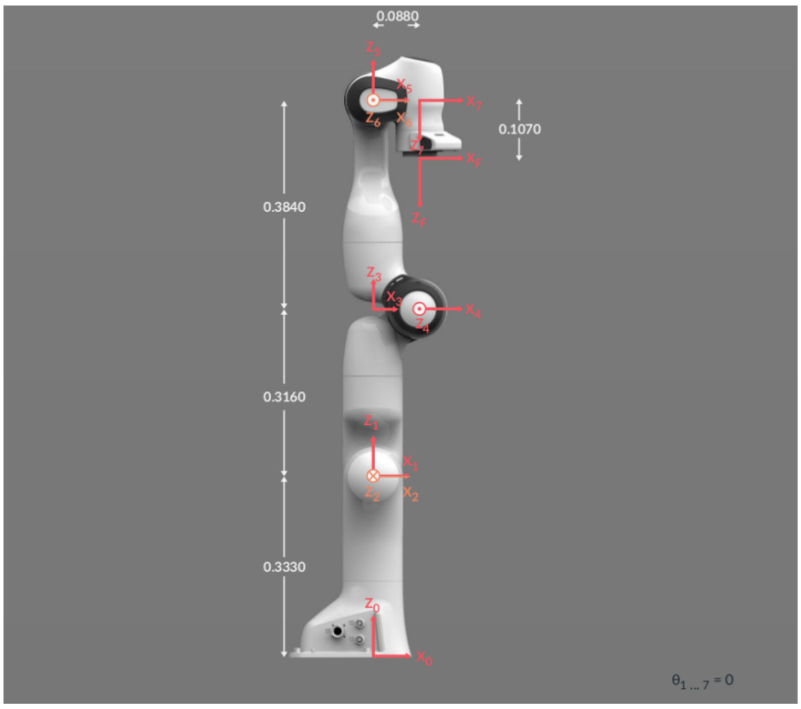
1. Introduction

In this homework, you are required to implement the forward kinematics (FK) and inverse kinematics (IK) functions for a 7-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.
- Answer questions regarding the manipulation pipeline.

We will use the PyBullet simulator for this homework. The robot arm used in this homework is Franka Emika PANDA, which is a 7-degree-of-freedom robot arm (containing 7 revolute joints).



The 7-degree-of-freedom robot arm we will use in this homework

2. Requirements

The requirement of the development environments:

- OS : ubuntu 18.04 , 20.04
- Python 3.7, 3.8 (You can use conda to create a new environment)
- **PyBullet simulator (you can install it by pip/pip3)**
- opencv-python
- Open3d

3. Detailed Descriptions

We will provide a project template (link) 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 conduct experiments and answer questions regarding the manipulation framework. Note that we have provided the basic implementation. [Bonus] You will need to modify our code to improve some parts of the manipulation framework and compare the results with our implementation.

The file structure of the project template:

```
PDMS-HW4
        | --- 3d_models/ # 3D object for the manipulation pipeline
        | --- data/ # Important information for the manipulation task
        | --- pybullet_planning/ # Motion planning algorithms
        | --- pybullet_robot_envs/ # contains all the important information of the robot arm
        | --- test_case/ # contains the public test cases to help you verify your code
        | --- utils/ # contains useful functions you may use
        | --- fk.py # contains the function that you need to implement for task 1
        | --- ik.py # contains the function that you need to implement for task 2
        | --- manipulation.py # the implementation for task 3
        | --- README.md
        | --- requirements.txt # contains all the packages you need
```

Please read the following the instructions in README.md **CAREFULLY**. Note that we highly recommend you develop this homework in the Ubuntu environment.

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

Task1: 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 7D array ranging from -π to π
    - The D-H parameters (following **Craig's 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 $R^{6 \times 7}$

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..m 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 a different convention
- geometric Jacobian in robotics

Task2: 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 9D joint parameters ranging from -π to π
        - 7D: for the revolute joints
        - 2D: for the gripper position of the arm
- Output:
    - The expected 9D joint parameters for the target 7D pose of the robot arm's end-effector
        - 7D: for the revolute joints (will be different from the input)
        - 2D: for the gripper position of the arm (you don't need to change it, keep it unchanged)

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.

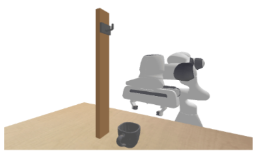
Similar to part 1, You need to modify **your_ik()** function in **ik.py** and make it output the expected results. **You cannot use any PyBullet or third-party functions to directly output the results. You will not get any points in this part if you do not follow the policy.** Besides, you can use a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

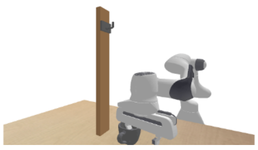
Hint: You may need your implemented **your_fk()** function in this task.

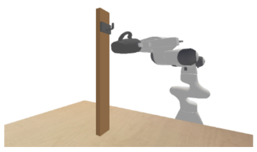
Task3: Answer questions regarding the manipulation pipeline.

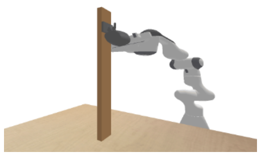
After finishing the previous two tasks, the robot arm can move autonomously! In this homework, we will go through a **hanging task** that the robot will hang a mug onto a hook (see **manipulation.py**). The manipulation framework contains 4 sub-tasks:

1. **reaching 1:** move the robot arm to the mug
2. **grasping:** grasp the mug
3. **reaching 2:** move the robot arm to the initial pose of hanging
4. **hanging:** hang the mug onto the hook


reaching 1


grasping


reaching 2

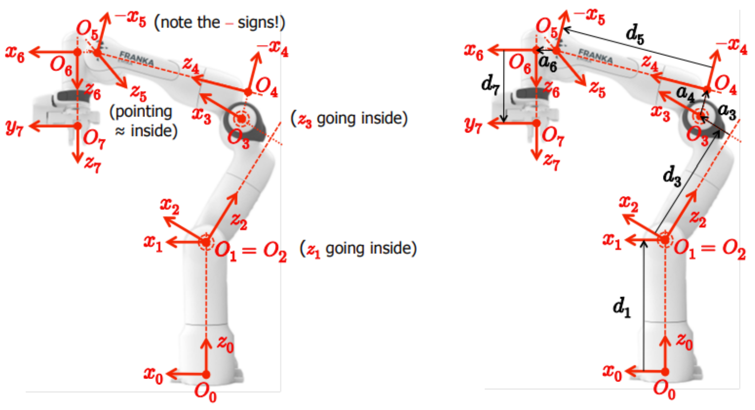

hanging

Please answer the following questions in your report:

1. About **task 1 (10%)**
   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? **(2%)**

   1.3 Complete the D-H table in your report following **D-H convention (5%)**



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 (15%)**

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 **manipulation.py (25%)**

This script uses the **your_ik()** function to control the robot and complete the hanging task. Besides, in the "**grasping**" sub-task, you need to annotate some keypoints to find the grasping pose via the **template pose matching method**, we have provided a basic implementation to do so using your annotated keypoints (see **get_src2dst_transform_from_kpts()** in **manipulation.py**).

3.1 Briefly explain how **get_src2dst_transform_from_kpts()** function works for pose matching and how **template_gripper_transform** work for gripper pose estimation in **manipulation.py**. (**5%**)

- (Hint: this function may be similar to some code in HW1)

3.2 What is the minimum number of keypoints we need to ensure the program runs properly? Why? (**5%**)

3.3 Briefly explain how to improve the matching accuracy **(5% + 5% bonus)**

- (You don't need to implement it. Of course, if you implement the **improved version** and compare the results with the original version, **you will get another 5 points!!!**)

3.4 Record the manipulation process in "**manipulation.mp4**" **(5%)**

- You can record your screen, and it needs to cover:
    - 1. the keypoint annotation process
    - 2. the whole manipulation pipeline in PyBullet window.
- See one example: https://drive.google.com/file/d/1mcTgRxkex5fFSwbcSLUp331M1xRxORCQ/view?usp=share_link

4. Submission

1. **File Format:**

```
studentID_Name_hw4
        | --- fk.py # contains your implementation of your_fk()
        | --- ik.py # contains your implementation of your_ik()
        | --- manipulation.py # the whole pipeline (you may need to modify it)
        | --- manipulation.mp4
        | --- report.pdf
        | --- other files or folders # only needed if you modify some files in other folders
```

   Please put all the files into the folder **studentID_name_hw4** and compress your folder into **one "zip" file**. Submit it to New E3 System with the format **studentID_name_hw4.zip.**

2. The **deadline** for this homework is **12/27 23:55.**
3. Late submission leads to **-20 points per day**.
4. **No Demo**. You don't need to demo your homework
5. **Wrong submission format leads to -10 point**
6. **NO CHEATING!! You will receive no credit if you are found cheating.**

5. Grading Policy

1. Correctness verification using test cases 60%
   - task 1: 20%
   - task 2: 40%
2. Report 40% + bonus 10%
   - about **task 1**: 10%
   - about **task 2**: 10% + 5% bonus
   - about **manipulation.py**: 20% + 5% bonus

6. References

- Project template of this homework:
  https://drive.google.com/file/d/1OiNQvW5Il4ZcJiaxnJjNcyB6AOUCQbPd/view?usp=share_link
- PyBullet quickstart guide:
  https://docs.google.com/document/d/10sXEhzFRSnvFcl3XxNGhnD4N2SedqwdAvK3dsihxVUA/edit#heading=h.2ye70wns7io3

- D-H parameters of Franka Emika PANDA robot: https://frankaemika.github.io/docs/control_parameters.html#denavithartenberg-parameters
- Inverse Kinematics using Jacobian methods: https://homes.cs.washington.edu/~todorov/courses/cseP590/06_JacobianMethods.pdf