Assignment 2: Kinematics

Robot Autonomy

Prof. Oliver Kroemer

1 Prerequisites

This homework will require ROS as we will be using Rviz for robot visualization. We will again be using Python 3 and suggest you use a virtual environment.

1.1 Ubuntu

1.1.1 Ubuntu 16.04

1. Install ROS Kinetic on your computer following instructions here: http://wiki.ros.org/kinetic/Installation/Ubuntu. (Make sure to install the Desktop-Full Install version of ROS Kinetic.)
2. Install libfranka and franka_ros using the following command:

```
sudo apt install ros-kinetic-libfranka ros-kinetic-franka-ros
```

3. Download and unzip the homework 2 code somewhere and again run:

```
pip install -r requirements.txt
```

4. Change the path argument on line 5 of hw2.launch to the path to your unzipped hw2 folder.

1.1.2 Ubuntu 18.04

1. Install ROS Melodic on your computer following instructions here: http://wiki.ros.org/melodic/Installation/Ubuntu. (Make sure to install the Desktop-Full Install version of ROS Melodic.)
2. Install libfranka and franka_ros using the following command:

```
sudo apt install ros-melodic-libfranka ros-melodic-franka-ros
```

3. Download and unzip the homework 2 code somewhere and again run:

```
pip install -r requirements.txt
```

4. Change the path argument on line 5 of hw2.launch to the path to your unzipped hw2 folder.

1.2 Windows and Mac

For Windows and Mac users, we have created a virtualbox image which is available to be downloaded here: `https://drive.google.com/file/d/1IuVqfBmtziWMbs6WIFBjZmb8SBc75Arn/view?usp=sharing`

1. You will need to download virtualbox from here: `https://www.virtualbox.org/wiki/Downloads`. (During installation, you must allow networking to pass through because we will be using this virtual machine in order to communicate with the robots during the labs and for your projects.)
2. You can load the virtualbox image by following instructions here: `https://www.maketecheasier.com/import-export-ova-files-in-virtualbox/`
3. The username for the Ubuntu 18.04 account is: `student`. The password is: `16-662`.
4. All files you will need are located in the hw2 folder in Documents. In addition a virtual environment with everything already prepared is in a folder called franka_env in your Documents folder. You can enter into the virtual environment by typing: sfranka. If you want to exit the virtual environment, simply type: deactivate.

2 Forward Kinematics

1) Parsing a Robot's URDF

On Slides 9 through 11 of Lecture 04 Kinematics, we covered parsing a URDF. You will be implementing a robot class that will first parse the URDF for the Franka Robot. Then you will implement a forward kinematics function that will take in the robot's joint positions and return the 4 x 4 homogeneous matrix of each joint location in cartesian space.

(1) [2 points] First, write a urdf parser in franka_robot.py that will extract the origins and axes for each joint into numpy arrays. As the flange to end-effector transformation is not in the urdf, I have provided it as the last row in both matrices. The urdf we will be using is in franka_robot.urdf.

(2) [5 points] Secondly, use the urdf parameters you loaded in and complete the forward_kinematics_urdf function in franka_robot.py that will use the origins and axes and the current robot's joint positions and calculate the homogeneous 4x4 matrices to each joint and the flange and end-effector. Note that the origin is the location of child / joint frame in the parent frame and axis represents the axis at which the joint rotates in the child / joint frame.

2) Denavit-Hartenberg Parameters

Another common way for robot manufacturers to communicate their robot's specifications is to define its Denavit-Hartenberg parameters.

(1) [5 points] In this question, you will be using the Denavit-Hartenberg parameters provided by Franka in the figure and table on the next page to again calculate the forward kinematics to each joint position. Then you can use this function to verify that both of your forward kinematics implementations match and are correct.

(2) [3 points] After you are done with both forward kinematics implementations, choose one and complete the end-effector function in franka_robot.py

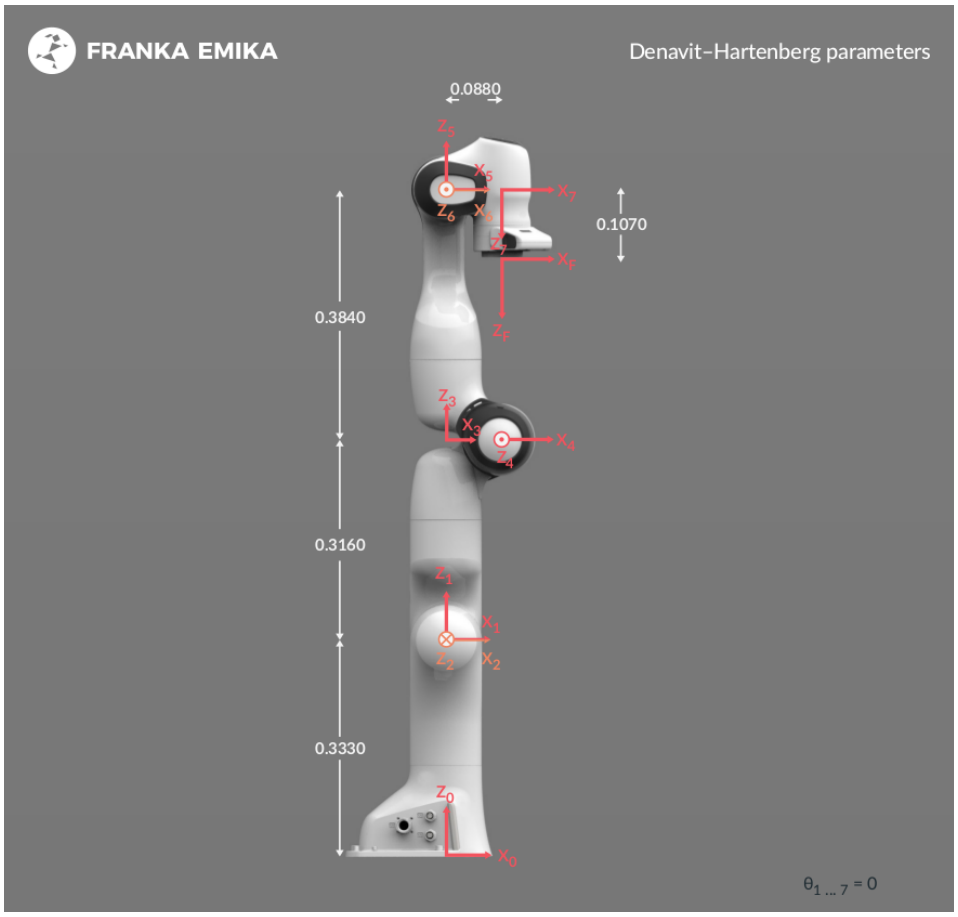
In order to implement forward kinematics using DH parameters, use the following equation to calculate the transformations:

$$\boldsymbol{T} = Rot_{x,\alpha}Trans_{x,a}Trans_{z,d}Rot_{z,\theta}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha & 0 \\ 0 & s_\alpha & c_\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & -s_\theta & 0 & 0 \\ s_\theta & c_\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $c_\alpha = \cos(\alpha)$, $s_\alpha = \sin(\alpha)$, $c_\theta = \cos(\theta)$, and $s_\theta = \sin(\theta)$.

* Note: Usually the correct transformation is $Trans_{z,d}Rot_{z,\theta}Trans_{x,a}Rot_{x,\alpha}$, but Franka defined their DH parameters differently so use $Rot_{x,\alpha}Trans_{x,a}Trans_{z,d}Rot_{z,\theta}$ in your implementation



Joint	a (m)	d (m)	α (rad)	θ (rad)
1	0	0.333	0	θ_1
2	0	0	$-\frac{\pi}{2}$	θ_2
3	0	0.316	$\frac{\pi}{2}$	θ_3
4	0.0825	0	$\frac{\pi}{2}$	θ_4
5	-0.0825	0.384	$-\frac{\pi}{2}$	θ_5
6	0	0	$\frac{\pi}{2}$	θ_6
7	0.088	0	$\frac{\pi}{2}$	θ_7
Flange	0	0.107	0	0
End-Effector	0	0.1034	0	0

3 Jacobian

1) Analytical Jacobian

   (1) [15 points] On Slides 12 and 13 of Lecture 04 Kinematics, we covered the Analytical end-effector Jacobian. Implement the Analytical end-effector Jacobian using the method discussed in class. You can test your Jacobian function out by running the simple test script test_jacobian.py.

4 Inverse Kinematics

1) Inverse Kinematics

   (1) [15 points] On Slides 20 through 25 of Lecture 04 Kinematics, we covered methods for calculating the iterative inverse kinematics of complex robots. Implement the jacobian transpose method in the inverse kinematics function and test it by trying out reachable end-effector poses and then running the computed joints through the end-effector function you implemented above to see if they are approximately equal.

5 Collision Checking

1) Box Box Collision Checking

   (1) [15 points] Implement the box-box collision checking as discussed in lecture. We have created boxes already for you, but you will need to determine how to use your forward kinematics to position the boxes so that they move with the robot. The box positions and sizes in the global frame are located in lines 209 to 258 in collision_boxes.py. To visualize the static boxes in the global frame, you should run the following commands: Open a new terminal without sourcing your virtual environment and run:

```
roslaunch hw2.launch
```

   Then in another terminal, run:

```
python test_collision_boxes.py
```

   Finally in another terminal, run:

```
python collision_boxes.py
```

   We have also provided the locations of the center of the boxes in the closest local link reference frame on the robot. The specifications are in the table below:

Box	Link	Translation (x,y,z)	Rotation (w,x,y,z)	Size (l,w,h)
1	1	(-0.04, 0, -0.283)	(1,0,0,0)	(0.23, 0.2, 0.1)
2	1	(-0.009, 0, -0.183)	(1, 0, 0, 0)	(0.13, 0.12, 0.1)
3	1	(0, -0.032, -0.082)	(0.9514, 0.3079, 0, 0)	(0.12, 0.1, 0.2)
4	1	(-0.008, 0, 0)	(1, 0, 0, 0)	(0.15, 0.27, 0.11)
5	1	(0, 0.042, 0.067)	(0.9514, 0.3079, 0, 0)	(0.12, 0.1, 0.2)
6	3	(0.00687, 0, -0.139)	(1, 0, 0, 0)	(0.13, 0.12, 0.25)
7	4	(-0.008, 0.004, 0)	(0.7071, -0.7071, 0, 0)	(0.13, 0.23, 0.15)
8	5	(0.00422, 0.05367, -0.121)	(0.9962, -0.08715, 0, 0)	(0.12, 0.12, 0.4)
9	5	(0.00422, 0.00367, -0.263)	(1, 0, 0, 0)	(0.12, 0.12, 0.25)
10	5	(0.00328, 0.0176, -0.0055)	(1, 0, 0, 0)	(0.13, 0.23, 0.12)
11	7	(-0.0136, 0.0092, 0.0083)	(0, 1, 0, 0)	(0.12, 0.12, 0.2)
12	7	(-0.0136, 0.0092, 0.1407)	(0.9239, 0, 0, -0.3827)	(0.08, 0.22, 0.17)

When you are finished with implementing your algorithm, run the following commands: Open a new terminal without sourcing your virtual environment and run:

```
roslaunch hw2.launch
```

Then in another terminal, run:

```
python test_collision_detection.py
```

Finally in another terminal, run:

```
python 6DoF_Box.py
```

Rviz should pop up with the Franka Robot and a rectangular box. You can move the cube around and test your box collision detection by seeing if a light turns red or green next to the robot. Once you are confident your implementation is working, take a video of your screen by pressing ctrl+alt+shift+r in Ubuntu 18.04 to start screen recording. Then move the cube around and show the collision detection at different joints. It will record your screen for 30 seconds and then save it to your videos folder. Include this video in your submission.

6 Submission Checklist

- ☐ Create a zip of `franka_robot.py`, your screenshot video, and any other necessary files.
- ☐ Upload the zip to Canvas.