Universidad de Monterrey
School of Engineering and Technologies

Final exam: Simulation forward kinematics of a Universal Robot 5 (UR5) in CoppeliaSim using Python & ZeroMQ Remote API

Course: Robotics

Lecturer: Dr. Andrés Hernández Gutiérrez

Learning objectives

After completing this exam, students would have achieved the correct simulation of forward kinematics of a Universal Robot 5 (UR5) in CoppeliaSim using Python and the ZeroMQ Remote API. To do so, students would have investigated the Denavit-Hartenberg[1] parameters of the UR5 robot to build the proper transformation matrices to capture the relationships between the individual links and joints of the robot. Figure 1 shows the UR5 robot used for this simulation.

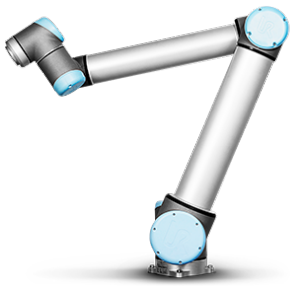


Figure 1: Universal Robot 5 (UR5) used for the simulation of the forward kinematics.

Introduction to Forward Kinematics[2]

Forward kinematics is a fundamental concept in robotics and mathematics that deals with determining the position and orientation of the end-effector (or tool) of a robot given the joint angles or joint displacements. It provides a way to calculate the pose of the robot's end-effector in the robot's coordinate system based on the joint values.

In simple terms, forward kinematics allows us to answer the question: "Where is the end of the robot arm located and how is it oriented, given the current configuration of the robot's joints?"

To perform forward kinematics, we start with a mathematical representation of the robot's structure, typically using transformation matrices or homogeneous transformations. These representations capture the relationships between the individual links and joints of the robot. Each joint is associated with a specific coordinate system, and these coordinate systems are linked together to form a chain, leading from the base of the robot to its end-effector.

By sequentially applying the transformations associated with each joint, we can propagate the coordinate system from the base to the end of the arm. The transformations involve translations and rotations that take into account the dimensions and geometry of the robot's links.

Once the transformations for all the joints have been combined, we obtain the final transformation matrix that represents the position and orientation of the end-effector relative to the base of the robot. This matrix contains information about the translation (x, y, z) and rotation (usually represented as a rotation matrix or quaternion) of the end-effector.

Forward kinematics is used in various robotic applications, including robot programming, motion planning, and control. By knowing the position and orientation of the end-effector, engineers and programmers can define desired trajectories, plan efficient paths, and control the robot's movements to perform specific tasks accurately.

It is worth noting that forward kinematics assumes that the robot has a fixed structure and that there are no external forces acting on the robot. Additionally, the accuracy of the forward kinematics solution depends on the accuracy of the robot's physical parameters and the precision of the joint angle measurements.

[1] This journal paper provides the Denavit-Hartenberg parameters of a UR5 robot.

[2] This section has been generated by ChatGPT3.0

Materials

- A personal computer.
- Internet connectivity, as you may need to install Python libraries and/or download files from Internet.
- Python $\geq$ 3.10 using the following libraries: `pyzmq`, `cbor`, `numpy`, `argparse`, `time`, `textwrap`.
- CoppeliaSim v4.3.0, (rev. 12).
- Jupyter Notebook
- Virtual Environment using Python $\geq$ 3.10
- Optionally: PyCharm, IDE, Visual Studio Code, Sublime, Atom, Gedit, etc.

Methodology

You may find the following procedure useful to simulate the forward kinematics of the UR5, nonetheless, feel free to propose and discuss your own methodology.

1. Work on the CoppeliSim simulator to become familiar with the virtualised UR5 robot.

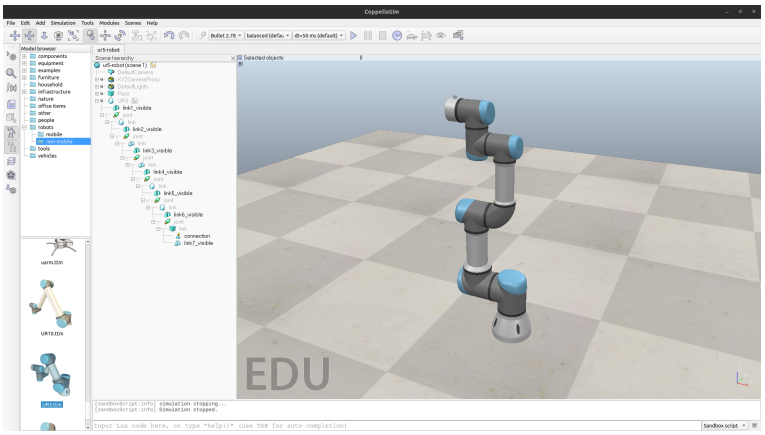
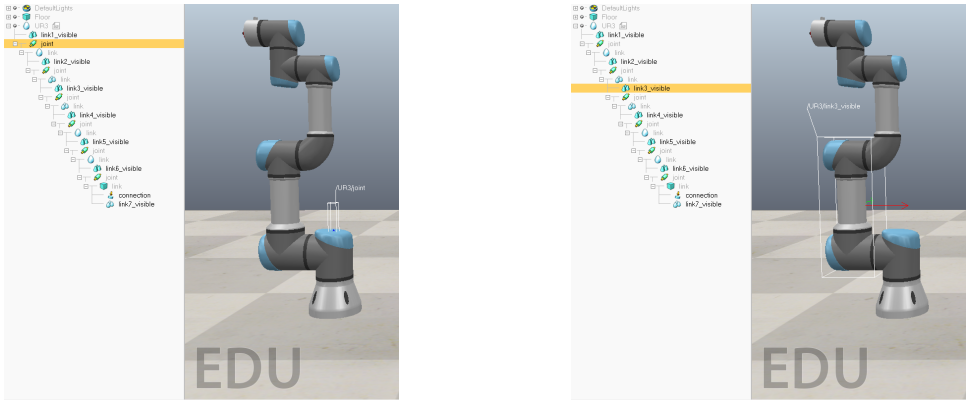


Figure 2: Virtual Universal Robot 5 model already available in CoppeliaSim v4.3.0.

2. From the *Scene hierarchy* window, identify the links and joints that are part of the UR5.



(a) (b)

Figure 3: Identification of joints and links of the virtual UR5 robot in CoppeliaSim. By selecting the different elements on the left-hand side panel, each link/joint is highlighted on the robot frame. Bear in mind that you may need to give each joint or link a unique name in order to be able to control the robot from Python using the Remote API.

3. Configure the Python Child Script.
4. Open your Jupyter Notebook.
5. Import the required Python libraries.
6. Create a client to get connected to the zmqRemoteApi from CoppeliaSim.
7. Get the handles of joints and links.
8. Start simulation.
9. Define the rotation angle of each joint.
10. Implement the forward-kinematics equations to predict the robot's end-effector position. Feel free to use the homogeneous matrix representation or the Denavit-Hartenberg description first then the homogeneous matrix representation.
11. Print the robot's end effector position with respect to the robot's base coordinate system.
12. Move the robot by applying the instruction `sim.setJointTargetPosition(jointx, anglex)` to every joint of the UR5 robot. `jointx` and `anglex` were defined in step 9.
13. Get the actual position of the robot's end-effector.
14. Compare the predicted and the actual robot's end-effector positions.
15. Discuss your results in detail.
16. Set all joints to a 0^o rotation angle.
17. Repeat steps 9 to 15 according to the following Table 1 (NB. Add as many code cells to your Jupyter Notebook as needed in order to test the following joints configurations):

Run	θ_1	θ_2	θ_3	θ_4	θ_5	Predicted position (X,Y,Z)	Actual position (X,Y,Z)
1	20^o	30^o	15^o	5^o	45^o		
2	0^o	-90^o	90^o	0^o	90^o		
3	0^o	10^o	20^o	30^o	40^o		

Table 1: Joints configurations and UR5 robot's end-effector position for four different simulations.

Submission guidelines

To complete this assignment:

- (20pts) Upload your CoppeliaSim scene to Blackboard. This should be ready for testing and be configure to be interfaced from Python using the Remote API.
- (20pts) Upload your Jupyter Notebook to Blackboard. It is expected to run properly when tested using Python 3.10.
- (20pts) Your Jupyter Notebook must address the above steps from 3 to 16.
- (20pts) Include a demo video (10min approx.), face recording is need, where you explain steps from 3 to 16.
- (10pts) Add your personal conclusions at the end of your Jupyter Notebook.
- (5pts) What would you like to learn/add to this assignment if you had to complete it just for fun and for the sole intention of learning more about this topic/simulator?
- (5pts) Add references.

Happy learning! - Andrés
"Knowing is not enough; we must apply. Willing is not enough; we must do" - J. W. von Goethe