Project: Kinematics Pick & Place

Rubric Points
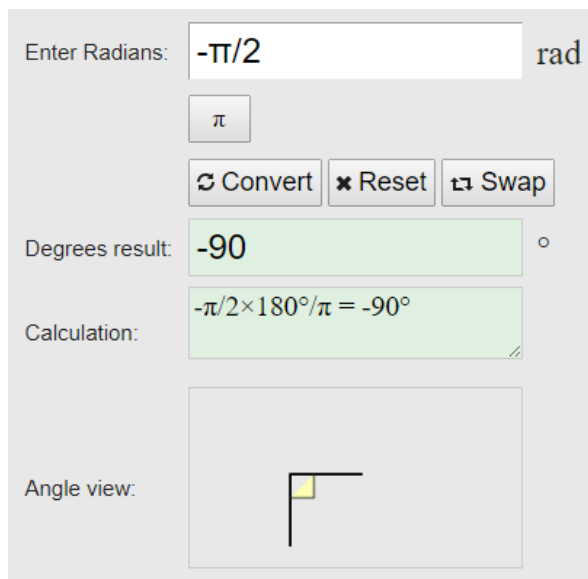
Here I will consider the rubric points individually and describe how I addressed each point in my implementation.


Kinematic Analysis

1. Run the forward_kinematics demo and evaluate the kr210.urdf.xacro file to perform kinematic analysis of Kuka KR210 robot and derive its DH parameters.

- **Twist angle (α)** – is defined in *<!--Links-->* section. Example is line 27 "*<origin xyz="0 0 0" rpy="0 0 0"/>*". Rpy= is where twist angle defined. As the result our twist angles are: alpha0 : 0, alpha1 : -pi/2, alpha2 : 0, alpha3 : -pi/2, alpha4 : pi/2, alpha5 : -pi/2, alpha6 : -pi/2. In this case all twists are around X-axis and define twist angle between Z-axis and next Z-axis.

- **Link length (a)** – is defined in *<!-- joints -->* section. Example is line 323 "*<origin xyz="0 0 0.33" rpy="0 0 0"/>*". Xyz= here is what gets our interest. I followed same sketch from the video (picture bellow). In link length, we are interested in distance between Z-axis and next Z-axis. As the result we get: a0 : 0, a1 : 0.35, a2 : 1.25, a3 : -0.54, a4 : 0, a5 : 0, a6 : 0.
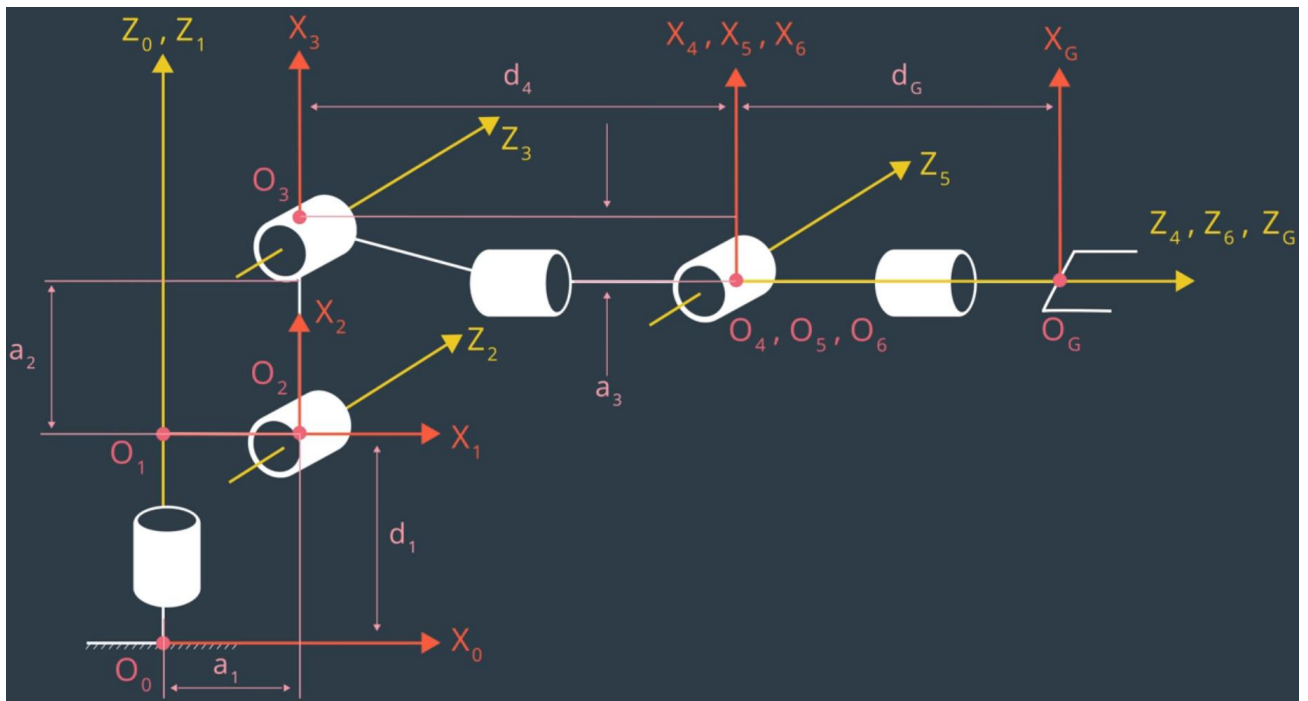
- **Link offset (d)** – also defined in *<!-- joints -->* section. But here we record different axis of coordinates according to picture above. As the result: d1 : 0.75, d2 : 0, d3 : 0, d4 : 1.5, d5 : 0, d6 : 0, d7 : 0.303. D7 is dg on the picture.
- **Joint angle (q)** – angle between X-axis and next X-axis. Not defined in URDF file (or did not find it). The only non-zero joint angle is between X1 and X2 (90 degrees). As the result: q1 : 0, q2 : q2-pi/2, q3 : 0, q4 : 0, q5 : 0, q6 : 0, q7 : 0.

2. Using the DH parameter table you derived earlier, create individual transformation matrices about each joint. In addition, also generate a generalized homogeneous transform between base_link and gripper_link using only end-effector(gripper) pose.

Transform matrixes:

Base to first:

cos(q1)	-sin(q1)	0	a0
sin(q1)*cos(alpha0)	cos(q1)*cos(alpha0)	-sin(alpha0)	-sin(alpha0)*d1
sin(q1)*sin(alpha0)	cos(q1)*sin(alpha0)	cos(alpha0)	cos(alpha0)*d1
0	0	0	1

First to second:

cos(q2)	-sin(q2)	0	a1

sin(q2)*cos(alpha1)	cos(q2)*cos(alpha1)	-sin(alpha1)	-sin(alpha1)*d2
sin(q2)*sin(alpha1)	cos(q2)*sin(alpha1)	cos(alpha1)	cos(alpha1)*d2
0	0	0	1

Second to third:

cos(q3)	-sin(q3)	0	a2
sin(q3)*cos(alpha2)	cos(q3)*cos(alpha2)	-sin(alpha2)	-sin(alpha2)*d3
sin(q3)*sin(alpha2)	cos(q3)*sin(alpha2)	cos(alpha2)	cos(alpha2)*d3
0	0	0	1

Third to forth:

cos(q4)	-sin(q4)	0	a3
sin(q4)*cos(alpha3)	cos(q4)*cos(alpha3)	-sin(alpha3)	-sin(alpha3)*d4
sin(q4)*sin(alpha3)	cos(q4)*sin(alpha3)	cos(alpha3)	cos(alpha3)*d4
0	0	0	1

Forth to fifth:

cos(q5)	-sin(q5)	0	a4
sin(q5)*cos(alpha4)	cos(q5)*cos(alpha4)	-sin(alpha4)	-sin(alpha4)*d5
sin(q5)*sin(alpha4)	cos(q5)*sin(alpha4)	cos(alpha4)	cos(alpha4)*d5
0	0	0	1

Fifth to sixth:

cos(q6)	-sin(q6)	0	a5
sin(q6)*cos(alpha5)	cos(q6)*cos(alpha5)	-sin(alpha5)	-sin(alpha5)*d6
sin(q6)*sin(alpha5)	cos(q6)*sin(alpha5)	cos(alpha5)	cos(alpha5)*d6
0	0	0	1

Sixth to gripper

cos(q7)	-sin(q4)	0	a6
sin(q7)*cos(alpha6)	cos(q7)*cos(alpha6)	-sin(alpha6)	-sin(alpha6)*d7
sin(q7)*sin(alpha6)	cos(q7)*sin(alpha6)	cos(alpha6)	cos(alpha6)*d7
0	0	0	1

Since all links are represented relative to the previous one, the answer would be dot product of all matrixes from base to gripper.

Generalized homogenous transform between link zero and gripper would be dot product of matrixes + offset of px, py and pz.

cos(pitch) * cos(yaw)	-sin(yaw) * cos(pitch)	sin(pitch)	px
sin(pitch) * sin(roll) * cos(yaw) + sin(yaw) * cos(roll)	-sin(pitch) * sin(roll) * sin(yaw) + cos(roll) * cos(yaw)	-sin(roll) * cos(pitch)	py
-sin(pitch) * cos(roll) * cos(yaw) + sin(roll) * sin(yaw)	sin(pitch) * sin(yaw) * cos(roll) + sin(roll) * cos(yaw)	cos(pitch) * cos(roll)	pz
0	0	0	1

Px, py, pz, roll, pitch, yaw - are given in IK_server.py as the part of inverse kinematics and represent end position and orientation of the gripper.

3. Decouple Inverse Kinematics problem into Inverse Position Kinematics and inverse Orientation Kinematics; doing so derive the equations to calculate all individual joint angles.

Please find comments in code (IK_server.py)

Project Implementation

1. Fill in the `IK_server.py` file with properly commented python code for calculating Inverse Kinematics based on previously performed Kinematic Analysis. Your code must guide the robot to successfully complete 8/10 pick and place cycles. Briefly discuss the code you implemented and your results.

I made comments in code. If further clarification needed, please point in review. In the result, I always shoot higher then needed (overshoot on Z or something). But weirdly enough, that actually helps not kicking things off, when arm start turning unexpectedly.

I understand that it is "you got a bug that fix another bug" kind of situation, but at this point I am completely out of ideas how to debug it further. It is also takes over 10 minutes to execute one pick and place on pretty much top of the line PC, so something is way off with performance as well.

You can find resulting video here https://youtu.be/kEEyYU_SL9g

You can also see what I am talking about when telling that getting higher helps at 2:40