writeup_template.md

Project: Kinematics Pick & Place

Writeup Template: You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.

Steps to complete the project:

1. Set up your ROS Workspace.
2. Download or clone the project repository into the ***src*** directory of your ROS Workspace.
3. Experiment with the forward_kinematics environment and get familiar with the robot.
4. Launch in demo mode.
5. Perform Kinematic Analysis for the robot following the project rubric.
6. Fill in the `IK_server.py` with your Inverse Kinematics code.

Rubric Points

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

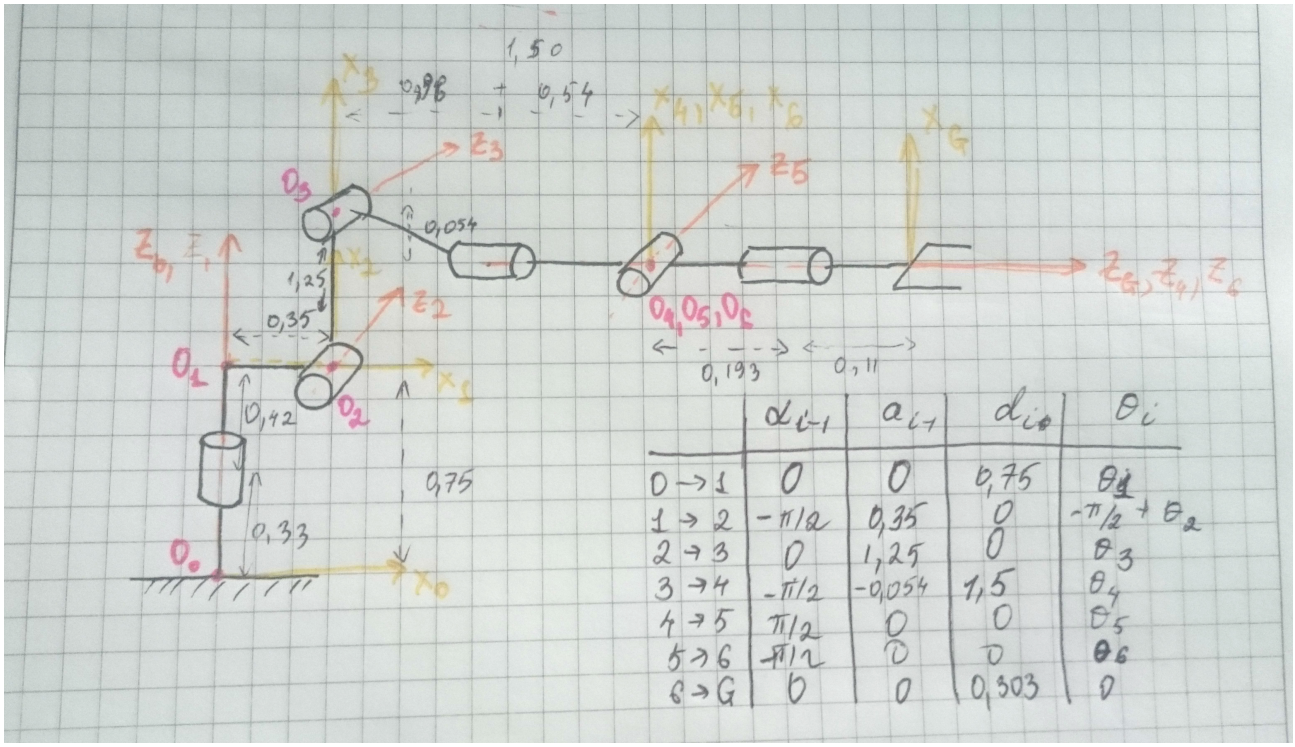
Writeup / README

1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf.

You're reading it!

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.



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.

Links	alpha(i-1)	a(i-1)	d(i)	theta(i)
0->1	0	0	0.75	q1
1->2	- pi/2	0.35	0	-pi/2 + q2
2->3	0	1.25	0	q3
3->4	- pi/2	-0.054	1.5	q4
4->5	pi/2	0	0	q5
5->6	- pi/2	0	0	q6
6->G	0	0	0.303	0

Individual transformation matrices

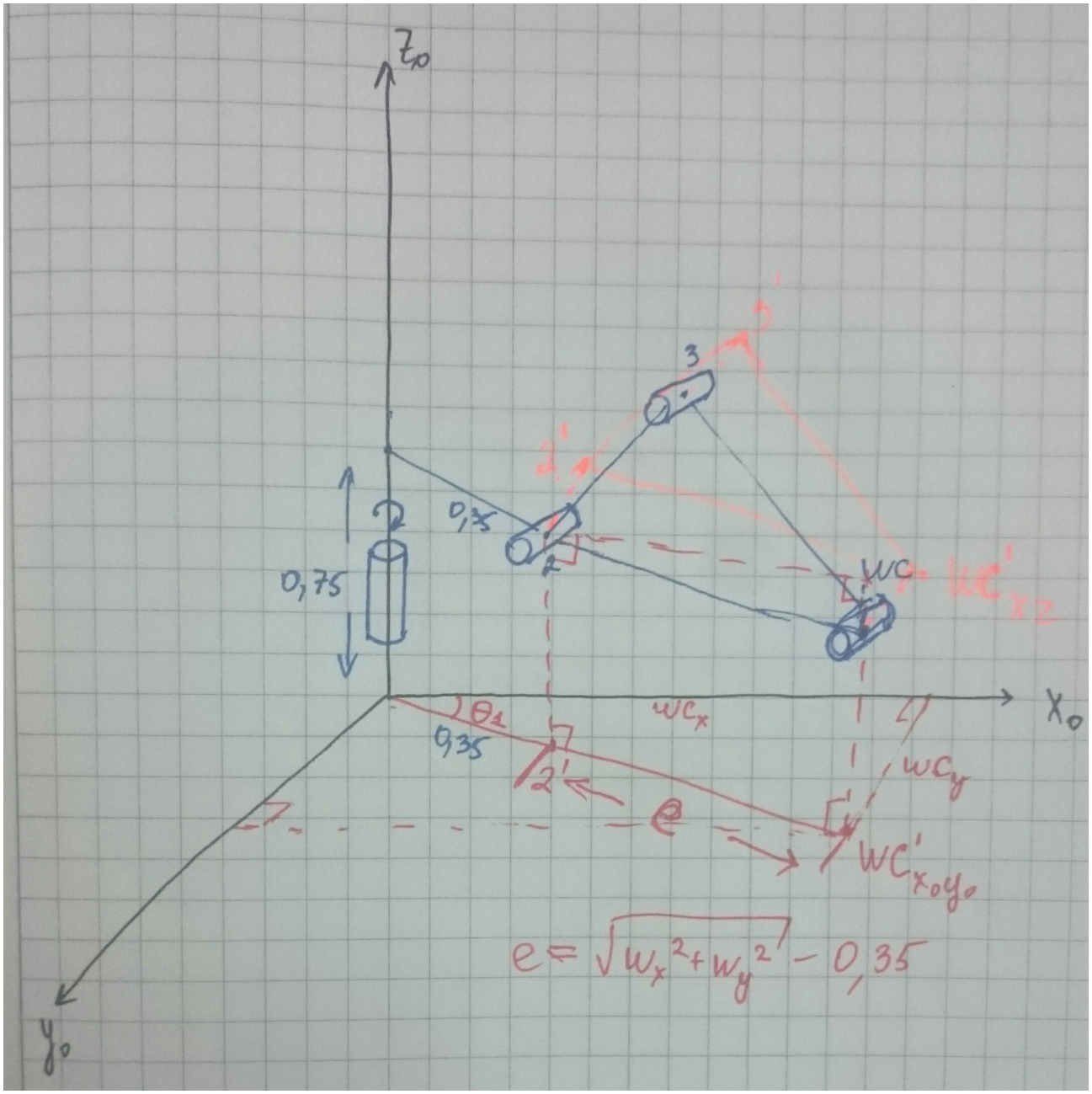
```
T0_1 = Matrix([[         cos(q1),        -sin(q1),        0,          a0],
               [ sin(q1)*cos(i0), cos(q1)*cos(i0), -sin(i0), -sin(i0)*d1],
               [ sin(q1)*sin(i0), cos(q1)*sin(i0),  cos(i0),  cos(i0)*d1],
               [               0,              0,        0,           1]])
T1_2 = Matrix([[         cos(q2),        -sin(q2),        0,          a1],
               [ sin(q2)*cos(i1), cos(q2)*cos(i1), -sin(i1), -sin(i1)*d2],
               [ sin(q2)*sin(i1), cos(q2)*sin(i1),  cos(i1),  cos(i1)*d2],
               [               0,              0,        0,           1]])
T2_3 = Matrix([[         cos(q3),        -sin(q3),        0,          a2],
               [ sin(q3)*cos(i2), cos(q3)*cos(i2), -sin(i2), -sin(i2)*d3],
               [ sin(q3)*sin(i2), cos(q3)*sin(i2),  cos(i2),  cos(i2)*d3],
               [               0,              0,        0,           1]])
T3_4 = Matrix([[         cos(q4),        -sin(q4),        0,          a3],
               [ sin(q4)*cos(i3), cos(q4)*cos(i3), -sin(i3), -sin(i3)*d4],
               [ sin(q4)*sin(i3), cos(q4)*sin(i3),  cos(i3),  cos(i3)*d4],
               [               0,              0,        0,           1]])
T4_5 = Matrix([[         cos(q5),        -sin(q5),        0,          a4],
               [ sin(q5)*cos(i4), cos(q5)*cos(i4), -sin(i4), -sin(i4)*d5],
               [ sin(q5)*sin(i4), cos(q5)*sin(i4),  cos(i4),  cos(i4)*d5],
```

```
           [              0,              0,        0,           1]])
T5_6 = Matrix([[         cos(q6),        -sin(q6),        0,          a5],
           [ sin(q6)*cos(i5), cos(q6)*cos(i5), -sin(i5), -sin(i5)*d6],
           [ sin(q6)*sin(i5), cos(q6)*sin(i5),  cos(i5),  cos(i5)*d6],
           [              0,              0,        0,           1]])
T6_G = Matrix([[         cos(q7),        -sin(q7),        0,          a6],
           [ sin(q7)*cos(i6), cos(q7)*cos(i6), -sin(i6), -sin(i6)*d7],
           [ sin(q7)*sin(i6), cos(q7)*sin(i6),  cos(i6),  cos(i6)*d7],
           [              0,              0,        0,           1]])
```

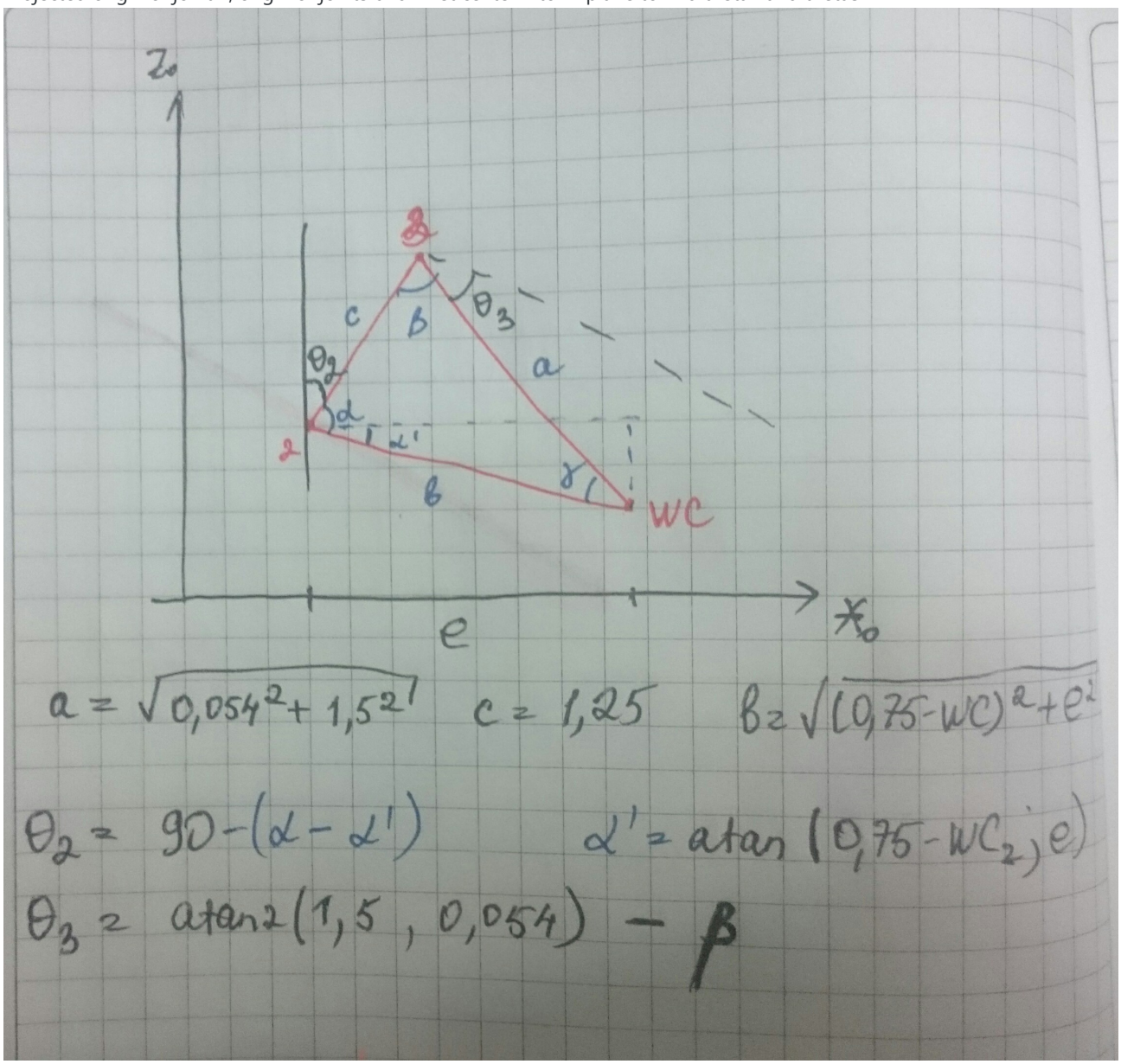
Generalized homogeneous transformation `T_total = T0_1*T1_2*T2_3*T3_4*T4_5*T5_6*T6_G`

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

Projected WC to x-y plane to find theta1



Projected origin of joint2 , origin of joint3 and wrist center into xz plane to find theta2 and theta3



To find theta2 and theta3 by using function atan2 used this formula

$$tan(\alpha) = \frac{4A}{b^2+c^2-a^2}$$
$$tan(\beta) = \frac{4A}{a^2+c^2-b^2}$$
$$tan(\gamma) = \frac{4A}{a^2+b^2-c^2}$$

Where A is Area of the triangle calculated using Heron's Formula

$$Area = \sqrt{S(S-a)(S-b)(S-c)}$$
$$S = \frac{a+b+c}{2}$$

Project Implementation

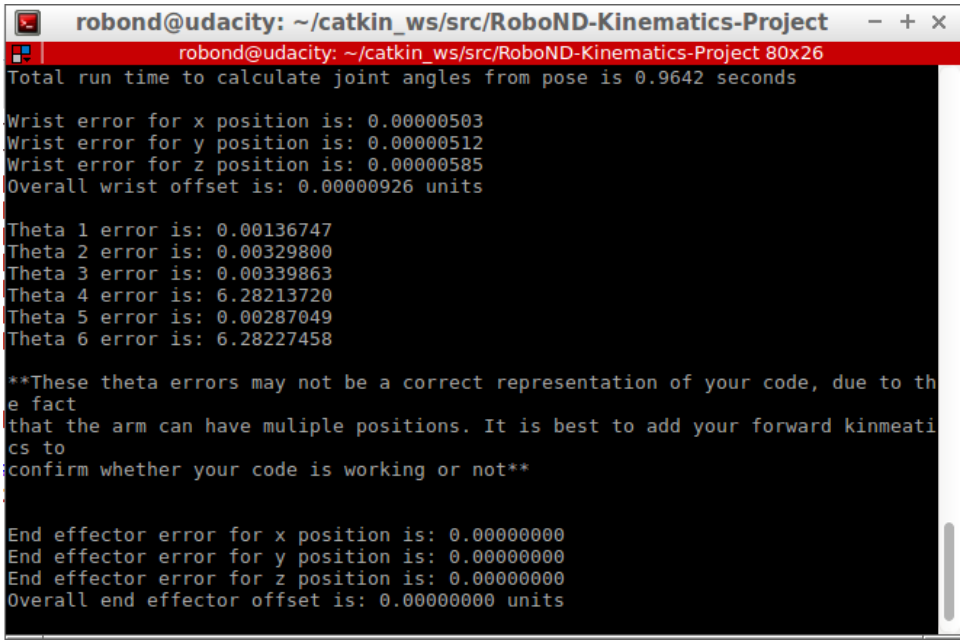
1. Fill in the `IK_server.py` file with properly commented python code for calculating Inverse Kinematics based on previously performed Kinematic Analysis. Your code must guide the robot to successfully complete 8/10 pick and place cycles. Briefly discuss the code you implemented and your results.

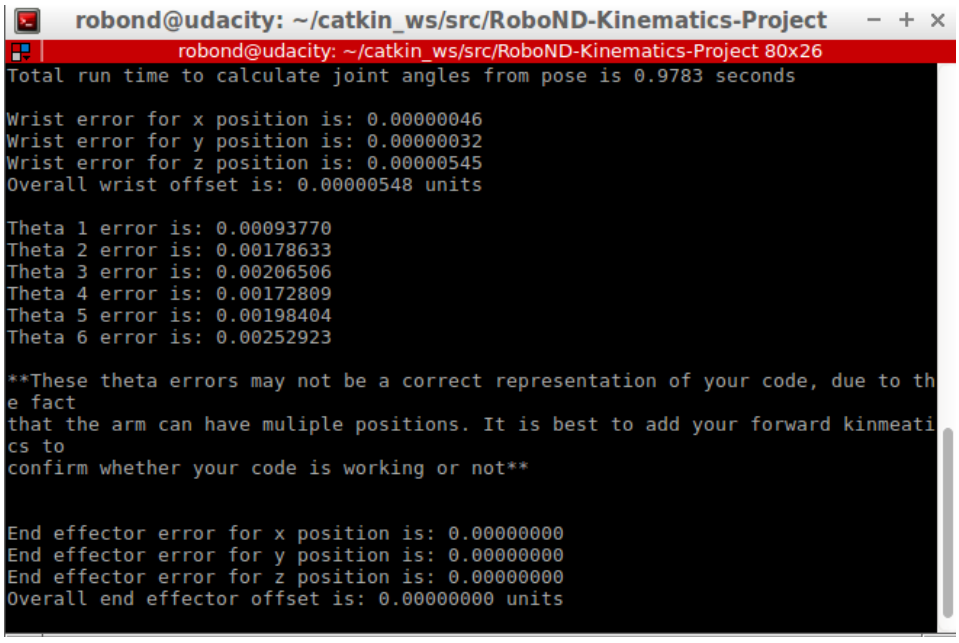
First, I obtained DH parameters and created function `transformation_matrix` which will return transformation matrix with passed parameters. Using the function I obtained individual transformation matrices. Then I defined rotational matrices around x,y,z axes, and rotation matrix for rotation sequence x-y-z to obtain rotation matrix of end effector.

For the IK , first multiplied rotation matrix to correction matrix to account of orientation difference of gripper in urdf vs DH convention. Then roll, pitch, yaw angles are substituted. After that, wrist center coordinates are obtained `WC = EE - (0.303) * R_EE[:,2]`

Now that wrist center coordinates are known, I calculated q1-3 using geometric IK method.Then using the angles rotation matrix from base link to joint 3 is calculated. From there we can get rot matrix from joint 3 to 6 `R3_6 = R0_3.inv("LU") * R_EE` Finally last three angles q4-6 are calculated using Eulers Angles from Rotation Matrix

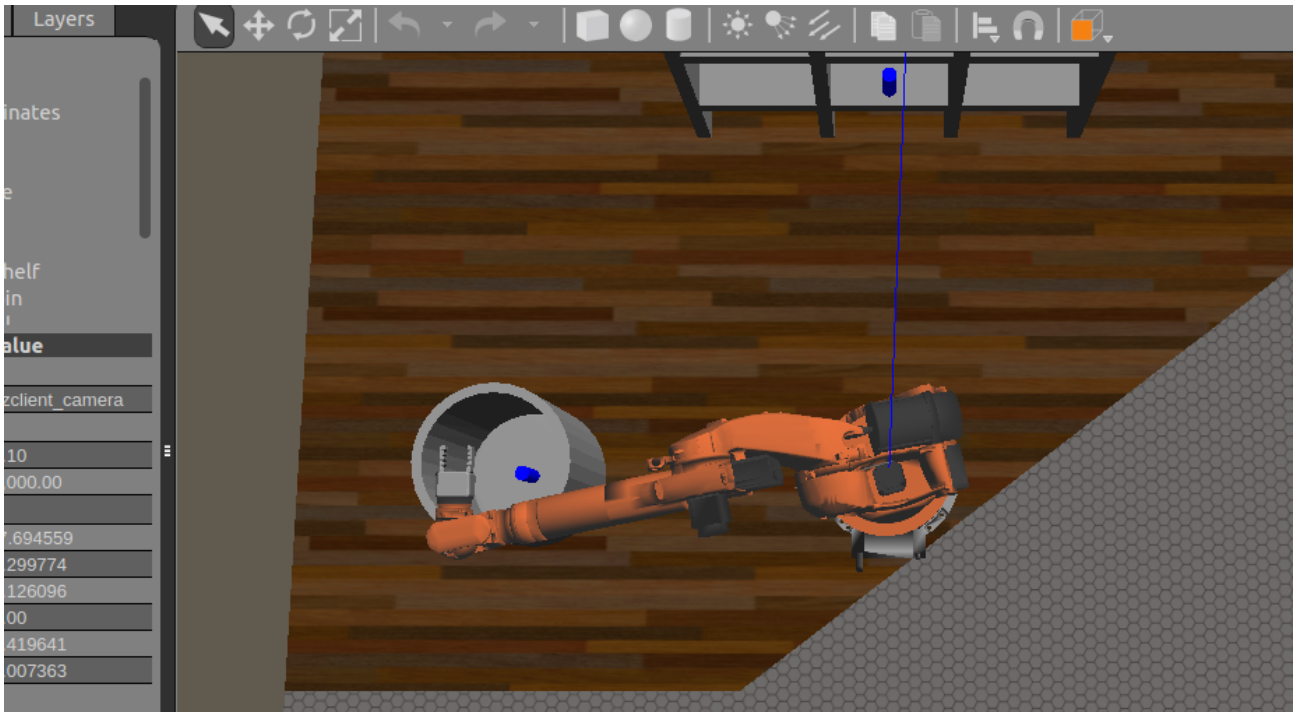
Here is some of the test results from IK_debug.py





The main challenge for me was definitely finding last three angles (q4-6). Finding correct rotational matrix took me a while to understand. Initially i thought it will be rotation from x-y-z planes, then tried out x-y-x. Then finally found out that we should be calculating rotational matrix using last three transformation matrices. Another thing that took me a long time to figure out was taking equation `R0_3 = T0_1[0:3, 0:3] * T1_2[0:3,0:3] * T2_3[0:3,0:3]` out of the loop incorrectly. That caused dropping into bin to fail every time, while grabbing the object worked correctly, and IK_debug.py always gave end effector offset 0.00000(because IK_debug didn't have a loop) . Then I realized I was defining R0_3 outside the loop and changing value of the same variable R0_3 inside the loop by substituting q1-3. After correcting this error finally bin drop worked correctly.

Some screenshots of grabbing object and dropping it into the bin



Overall pick and place worked mostly correctly, with occasional grabbing fail due to the trajectory knoking object off the shelf. I think the calculation speed can definitely be improved more , and angle calculation accuracy can be improved.