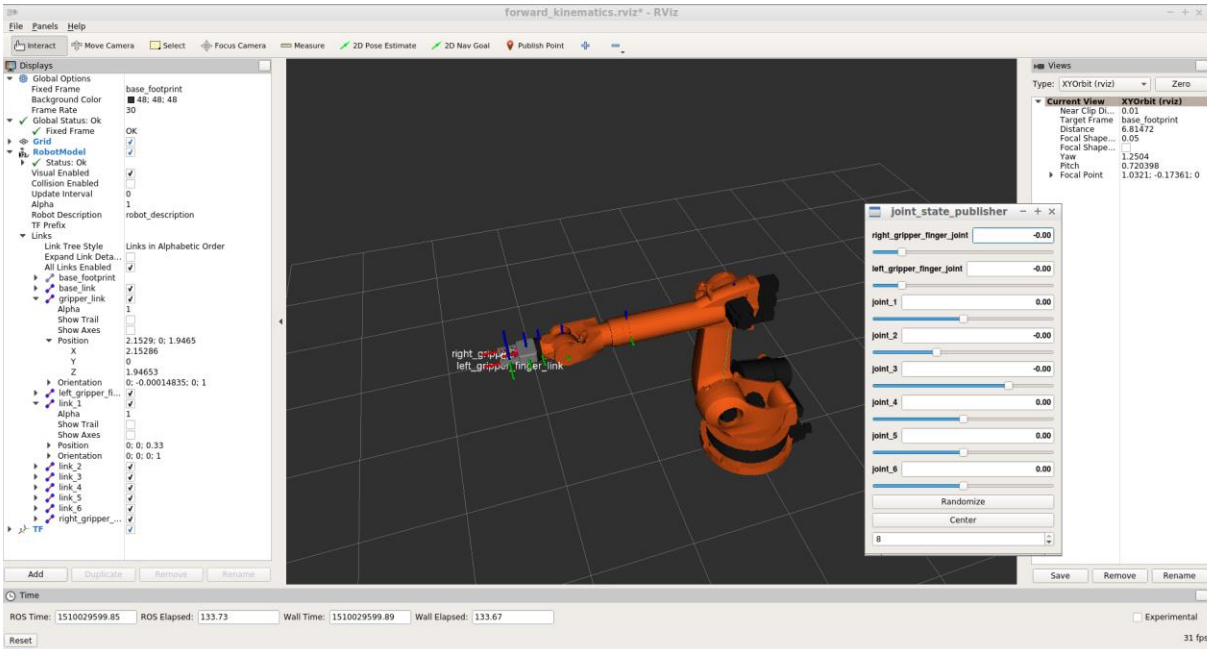
UDACITY – PICK AND PLACE PROJECT

RUBRIC POINT EVALUATION

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.

The forward_kinematics.launch will have position and orientation for each link. This helps is understanding the DH parameters, apart from the data present in the URDF file.



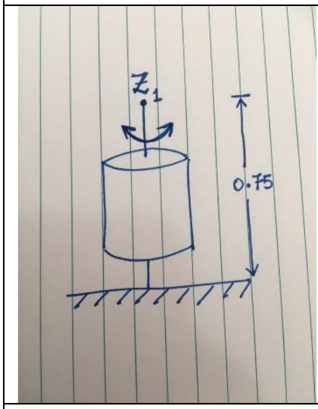
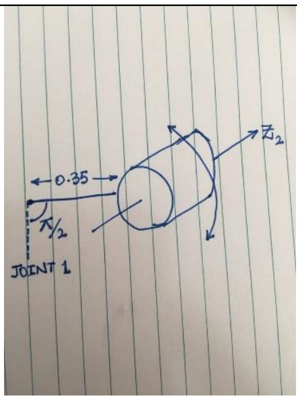
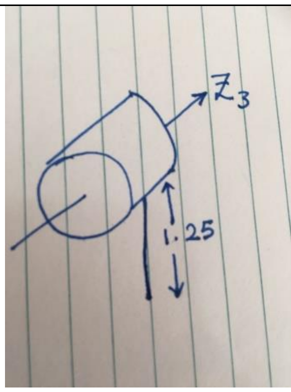
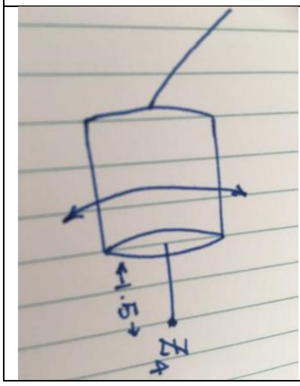
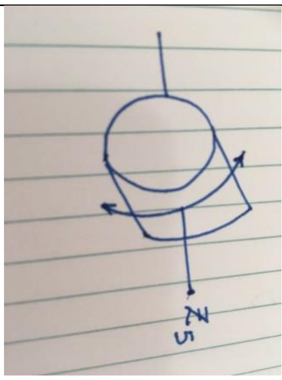
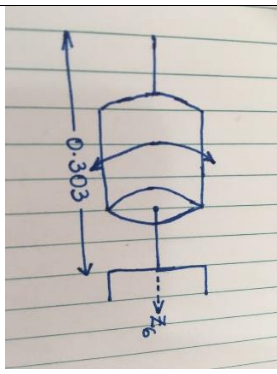

```
-<joint name="joint_2" type="revolute">
    <origin xyz="0.35 0 0.42" rpy="0 0 0"/>
    <parent link="link_1"/>
    <child link="link_2"/>
    <axis xyz="0 1 0"/>
    <limit lower="${-45*deg}" upper="${85*deg}" effort="300" velocity="${115*deg}"/>
 </joint>
```

This helps in determining all the DH parameters, to help in calculating forward kinematics.

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.

This is the DH parameter table derived:

```
DH_Table = { alpha0:      0, a0:      0, d1:   0.75, q1:         q1,
             alpha1:  -a/2., a1:   0.35, d2:      0, q2: -a/2. + q2,
             alpha2:      0, a2:   1.25, d3:      0, q3:         q3,
             alpha3:  -a/2., a3: -0.054, d4:    1.5, q4:         q4,
             alpha4:   a/2., a4:      0, d5:      0, q5:         q5,
             alpha5:  -a/2., a5:      0, d6:      0, q6:         q6,
             alpha6:      0, a6:      0, d7:  0.303, q7:          0}
```

JOINT 1	JOINT 2	JOINT 3
		
JOINT 4	**JOINT 5**	**JOINT 6**
		

All the above transformations were carried out with the below code for corresponding links:

```
alpha00 = alpha0.evalf(subs=DH_Table)
a00 = a0.evalf(subs=DH_Table)
d11 = d1.evalf(subs=DH_Table)
T0_1  = TF_Matrix(alpha00, a00, d11, q1).subs(DH_Table)

T0_EE = T0_1 * T1_2 * T2_3 * T3_4 * T4_5 * T5_6 * T6_EE
T0_EE = T0_EE.evalf()
```

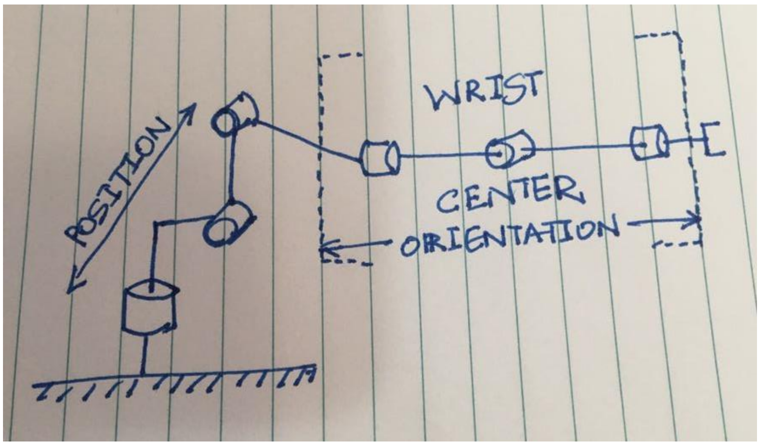
Where TF_Matrix is a subroutine with the DH matrix defined

FK = T0_EE.evalf(subs={q1: theta1, q2: theta2, q3: theta3, q4: theta4, q5: theta5, q6: theta6})

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

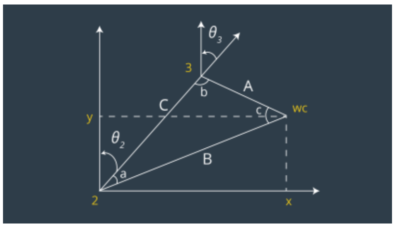
IK problem in this case = IK Position + IK Orientation

The analogy for this example is that of a wrist,



The position of the wrist centre is calculated by the following equations:

theta1 = atan2(WC[1], WC[0])
theta2 = pi/2. - angle_a - atan2(WC[2] - 0.75, sqrt(WC[0] * WC[0] + WC[1] * WC[1]) - 0.35)
theta3 = pi/2. - (angle_b + 0.036)
A graphical representation of the wrist center is as shown below by Udacity.



The rest of the angles are calculated as below:

theta4 = atan2(R3_6[2,2], -R3_6[0,2])
theta5 = atan2(sqrt(R3_6[0,2] * R3_6[0,2] + R3_6[2,2] * R3_6[2,2]), R3_6[1,2])
theta6 = atan2(-R3_6[1,1], R3_6[1,0])

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.

IK_server.py is populated with code to perform the following

1. Define DH parameters
2. Perform conversion for each link from base_link to subsequent link upto gripper_link
3. Perform a homogenous transformation from base_link to gripper_link using above step
4. Perform IK using the following steps
    a. Calculate wrist center
    b. Calculate the rest of the angles
5. Feed the calculated angles into the forward kinematics part

RESULTS

As guessed, the IK_server.py code takes a lot of time in IK rather than FK. The arm is in a stand still until the angles and hence the trajectory is calculated. It then drops off the sample into the bin. On a total of 10, the code initially had a success rate of 6. But once I used atan2 instead of atan and acos, the calculation was marginally faster and then improved to 8. Here's the You Tube link to a video of one of the runs - https://youtu.be/21eDa5Bw57E.

CONCLUSION

There are a few points of improvement that can be implemented for this requirement. The path from the sample to the drop off location can be stored and then a smoothing algorithm can be implemented on the path. This will enable optimal movement over time.

Comments provided by reviewer have been done wherever possible