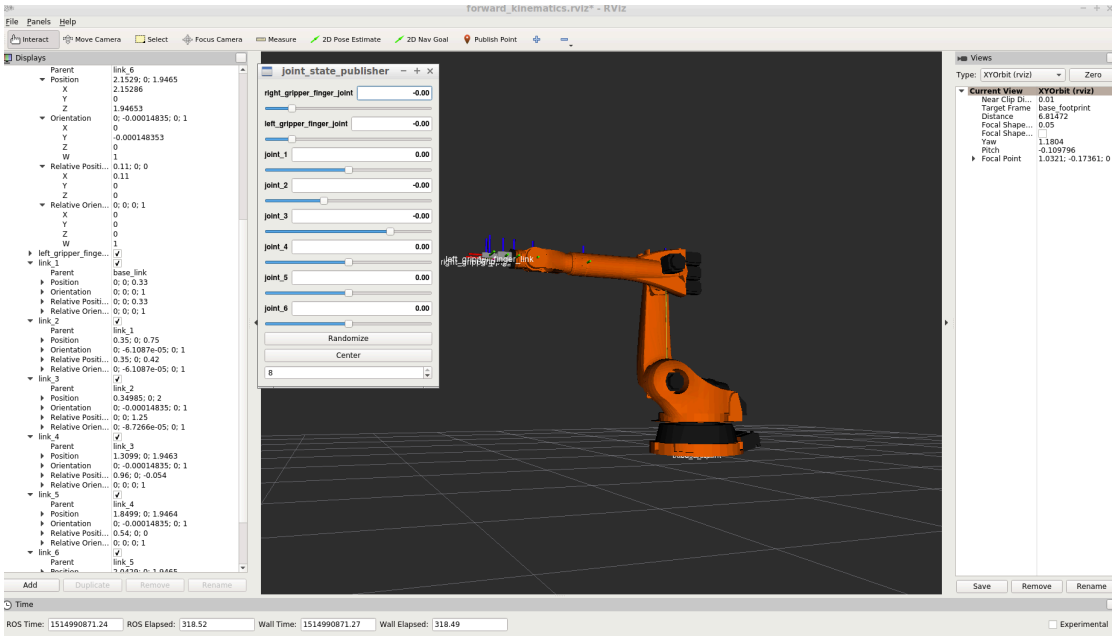
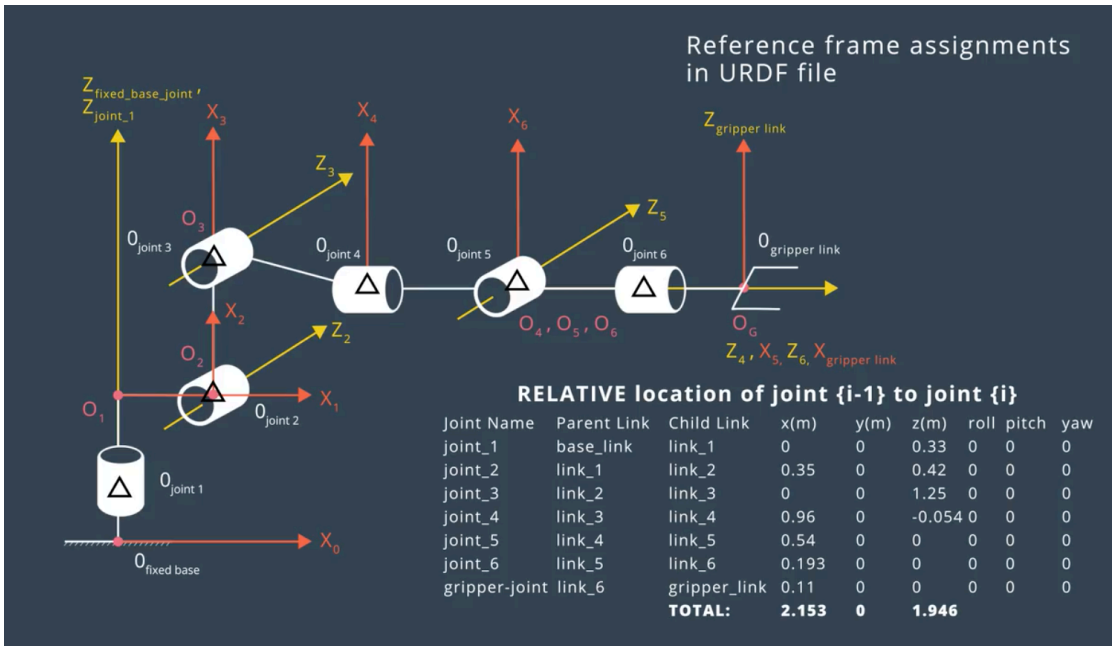
Kinematic Analysis

Zhanghanwei

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.



run the demo and analysis from `of kuka KR210 robot`



DH parameters were derived from the arm according to the assignments in urdf file.

. 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-1)	theta(i)
0->1	0	0	0.33+0.42	q1
1->2	- pi/2	0.35	0	q2:-pi/2 + q2
2->3	0	1.25	0	q3
3->4	- pi/2	-0.054	0.96+0.54	q4
4->5	pi/2	0	0	q5
5->6	- pi/2	0	0	q6
6->EE	0	0	0.193+0.11	q7:0

Create Modified DH parameters dict

```
# Define Modified DH Transformation matrix
s  = {alpha0: 0.,      a0: 0.,                  d1: 0.33+0.42,  q1: q1
    ,alpha1: -pi/2,   a1: 0.35,                d2: 0.,         q2: q2 - pi/2.
    ,alpha2: 0.,      a2: 1.25,                d3: 0.,         q3: q3
    ,alpha3: -pi/2,   a3: -0.054,              d4: 0.96+0.54,  q4: q4
    ,alpha4: pi/2,    a4: 0.,                  d5: 0.,         q5: q5
    ,alpha5: -pi/2,   a5: 0.,                  d6: 0.,         q6: q6
    ,alpha6: 0.,      a6: 0.,                  d7: 0.193+0.11, q7: 0.          }
```

Define Modified DH Transformation matrix

```
def getHomogeneousTransforms(alpha,a,d,q):
    T = Matrix([[            cos(q),           -sin(q),          0,             a],
                [  sin(q)*cos(alpha),  cos(q)*cos(alpha), -sin(alpha),  -sin(alpha)*d],
                [  sin(q)*sin(alpha),  cos(q)*sin(alpha),  cos(alpha),   cos(alpha)*d],
                [                  0,                 0,          0,             1]])
    return T
```

Extract rotation matrices from the transformation matrices, and generalized homogeneous transform between base_link and gripper using only the gripper pose

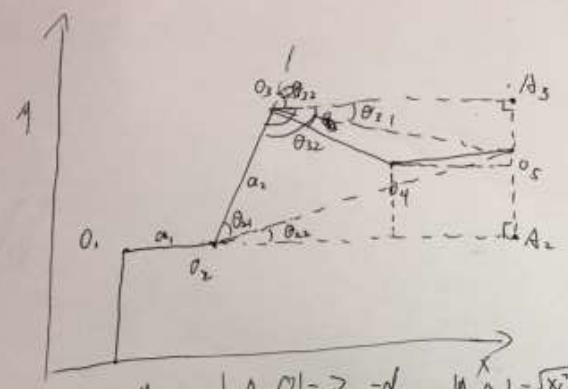
```
# Extract rotation matrices from the transformation matrices
T0_1 = getHomogeneousTransforms(alpha0,a0,d1,q1).subs(s)
T1_2 = getHomogeneousTransforms(alpha1,a1,d2,q2).subs(s)
T2_3 = getHomogeneousTransforms(alpha2,a2,d3,q3).subs(s)
T3_4 = getHomogeneousTransforms(alpha3,a3,d4,q4).subs(s)
T4_5 = getHomogeneousTransforms(alpha4,a4,d5,q5).subs(s)
T5_6 = getHomogeneousTransforms(alpha5,a5,d6,q6).subs(s)
T6_7 = getHomogeneousTransforms(alpha6,a6,d7,q7).subs(s)
T0_3 = simplify(T0_1 * T1_2 * T2_3)
T0_7 = simplify(T0_3 * T3_4 * T4_5 * T5_6 * T6_7)

R01 = rot_x(alpha0) * rot_z(q1)
R12 = rot_x(alpha1) * rot_z(q2)
R23 = rot_x(alpha2) * rot_z(q3)
R03 = simplify(R01 * R12 * R23)
R03 = R03.subs(s)
```

```
# Compensate for rotation discrepancy between DH parameters and Gazebo
R_corr = rot_z(pi) * rot_y(-pi / 2)
Rotation = rot_z(yaw) * rot_y(pitch) * rot_x(roll) * R_corr
```

R36 = simplify(R03.T * Rotation)

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



$$\theta_1 = \tan^{-1}\frac{y_c}{x_c} \qquad |A_2O_5| = z_c - d_1 \qquad |A_2O_2| = \sqrt{x_c^2+y_c^2} - a_1$$

$$\theta_2 = \theta_{21} + \theta_{22} = \tan^{-1}\frac{|O_5A_2|}{|O_2A_2|} + \cos^{-1}\frac{|O_2O_3|^2 + |O_2O_5|^2 - |O_3O_5|^2}{2|O_2O_3||O_2O_5|}$$

$$|O_2O_5| = \sqrt{|A_2O_5|^2 + |A_2O_2|^2} = \sqrt{(z_c-d_1)^2 + (\sqrt{x_c^2+y_c^2}-a_1)^2} = W$$

$$\theta_2 = \tan^{-1}\left(\frac{z_c-d_1}{\sqrt{x_c^2+y_c^2}-a_1}\right) + \cos^{-1}\frac{a_2^2 + W^2 - (a_3^2+d_4^2)}{2\cdot a_2\cdot W}$$

$$\theta_3 = \theta_{32} - \theta_{31} = \frac{\pi}{2} - \angle A_3O_3O_5 - \angle O_2O_3O_5 = \frac{\pi}{2} -$$

$$= \frac{\pi}{2} - \tan^{-1}\frac{|a_3|}{|d_4|} - \cos^{-1}\frac{|O_2O_3|^2 + |O_3O_5|^2 - |O_2O_5|^2}{2|O_2O_3|\cdot|O_3O_5|}$$

$$\cos^{-1}\frac{a_2^2 + a_3^2 + d_4^2 - W^2}{2a_2\sqrt{a_3^2+d_4^2}}$$

$$\frac{3}{6}R = \begin{pmatrix} \cos\theta_4\cos\theta_5\cos\theta_6 - \sin\theta_4\sin\theta_6 & \cdots & \cdots \\ \sin\theta_5\cos\theta_6 & -\sin\theta_5\sin\theta_6 & \cos\theta_5 \\ -\sin\theta_4\cos\theta_5\cos\theta_6 - \cos\theta_4\sin\theta_6 & \sim & \sim \end{pmatrix}$$

$$\theta_4 = -\tan^{-1}\frac{{}^3_6R(2,2)}{{}^3_6R(0,2)}$$

$$\theta_5 = \tan^{-1}\frac{{}^3_6R(1,0)/\cos\theta_6}{{}^3_6R(1,2}$$

$$\theta_6 = -\tan^{-1}\frac{{}^3_6R(1,1)}{{}^3_6R(1,0)}$$

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. Briefly discuss the code you implemented and your results.**

From the picture I draw out and show your math for the derivation of the theta angles, here is the code to calculating results. The robot track the planned trajectory and successfully pick and place operation. But sometimes robot hit the blue object to the floor before catch it.

```
l23 = a2
l35 = sqrt(a3**2 + d4**2)
D1 = sqrt(wcx * wcx + wcy * wcy) - a1
D2 = wcz - d1
ww = D1**2 + D2**2
w = sqrt(ww)

theta1 = atan2(wcy,wcx)

theta2 = -(atan2(D2, D1) + acos((l23**2 + ww- l35**2)/(2*l23*w)))
theta2 = theta2.evalf(subs=s)
theta2 = theta2 + pi/2

theta3 = pi/2 - acos((l23**2 - ww + l35**2)/(2*l23*l35)) + atan2(a3, d4)
theta3 = theta3.evalf(subs=s)

R36 = simplify(R03.T * Rotation)
R36 = R36.evalf(subs={q1: theta1, q2: theta2 , q3: theta3})

if R36[1,2] == 1:
    theta4 = 0
    theta5 = 0
    theta6 = atan2(-R36[0,1],R36[0,0])
elif R36[1,2] == -1:
    theta4 = 0
    theta5 = pi
    theta6 = atan2(R36[0,1], -R36[0,0])
else:
    theta4 = atan2( R36[2,2], -R36[0,2])
    theta6 = atan2(-R36[1,1],  R36[1,0])
    theta5 = atan2(R36[1,0]/cos(theta6), R36[1,2])
```

run the IK_debug.py

```
Wrist error for x position is: 0.00000046
Wrist error for y position is: 0.00000032
Wrist error for z position is: 0.00000545
Overall wrist offset is: 0.00000548 units

Theta 1 error is: 0.00093770
Theta 2 error is: 0.00178633
Theta 3 error is: 0.00206506
Theta 4 error is: 0.00172809
Theta 5 error is: 0.00198404
Theta 6 error is: 0.00252923

**These theta errors may not be a correct representation of your code, due to the fact
that the arm can have muliple positions. It is best to add your forward kinmeatics to
confirm whether your code is working or not**


End effector error for x position is: 0.00000000
End effector error for y position is: 0.00000000
End effector error for z position is: 0.00000000
Overall end effector offset is: 0.00000000 units
```