

## Project: Kinematics Pick & Place

### Rubric Points

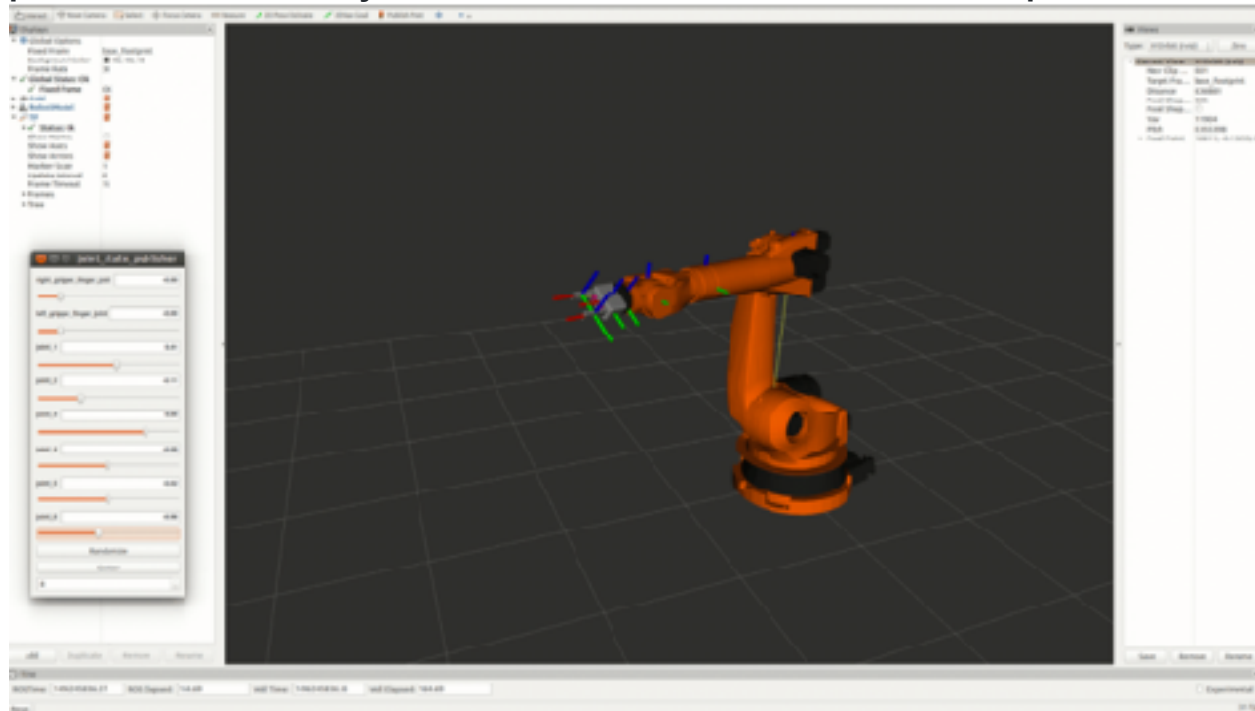
Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

### Writeup / README

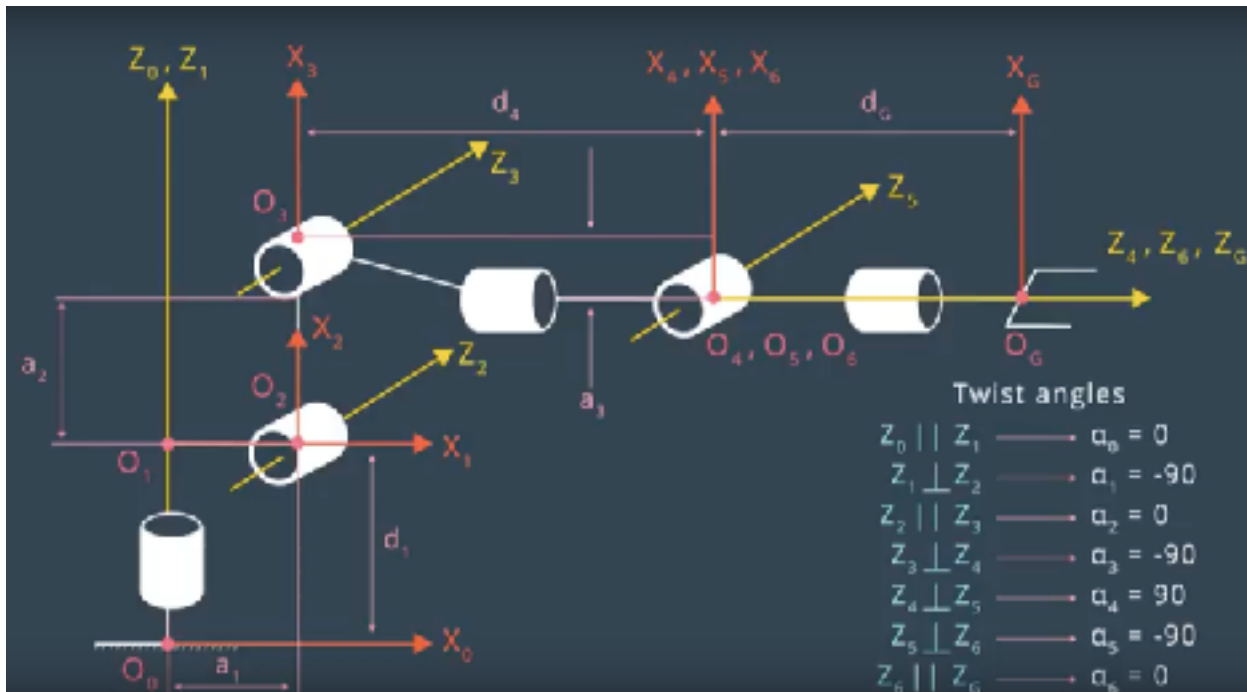
1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. You're reading it!

### Kinematic Analysis

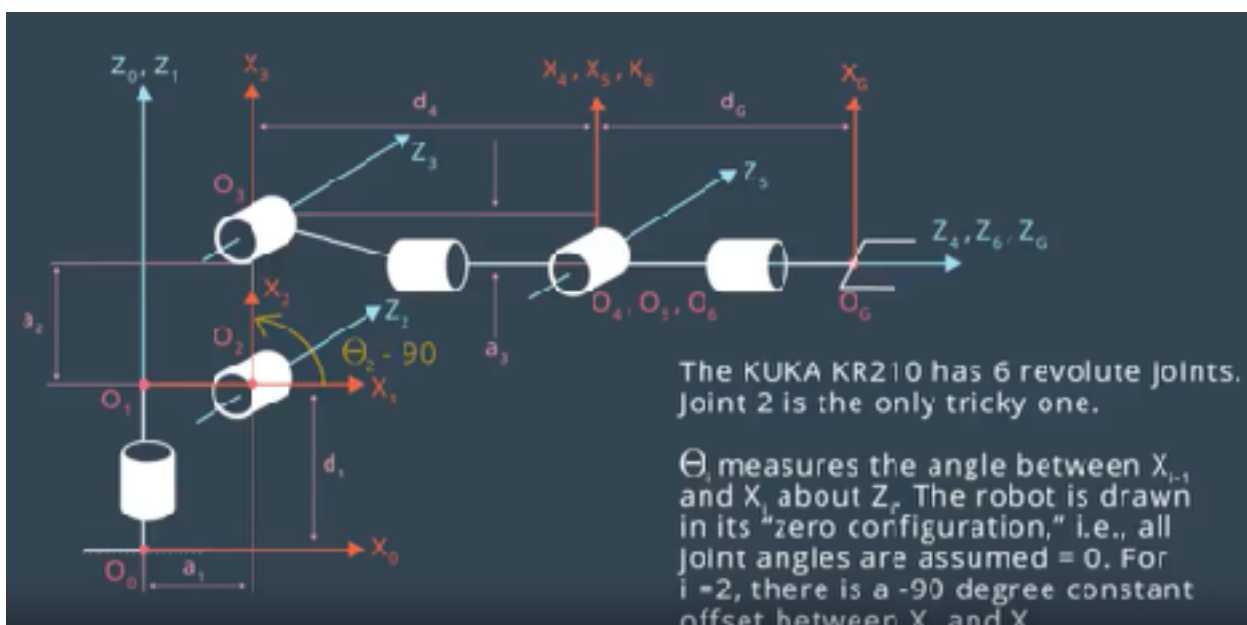
1. Run the forward\_kinematics demo and evaluate the kr210.urdf.xacro file to perform kinematic analysis of Kuka KR210 robot and derive its DH parameters.



## DH Parameters



- 1/ The twist angles  $\alpha_i$  is the angle between  $z_{i-1}$  and  $z_i$  measured about  $x_{i-1}$  in (+ in anti-clockwise direction)
- 2/ The link length  $a_{i-1}$  is the distance from  $z_{i-1}$  to  $z_i$
- 3/ The link offset  $d_i$  is the signed distance between  $x_{i-1}$  and  $x_i$  measured along  $z_i$  axis.
- 4/ The joint variable  $Q_i$  is the angle between  $x_{i-1}$  and  $x_i$  about  $z_i$



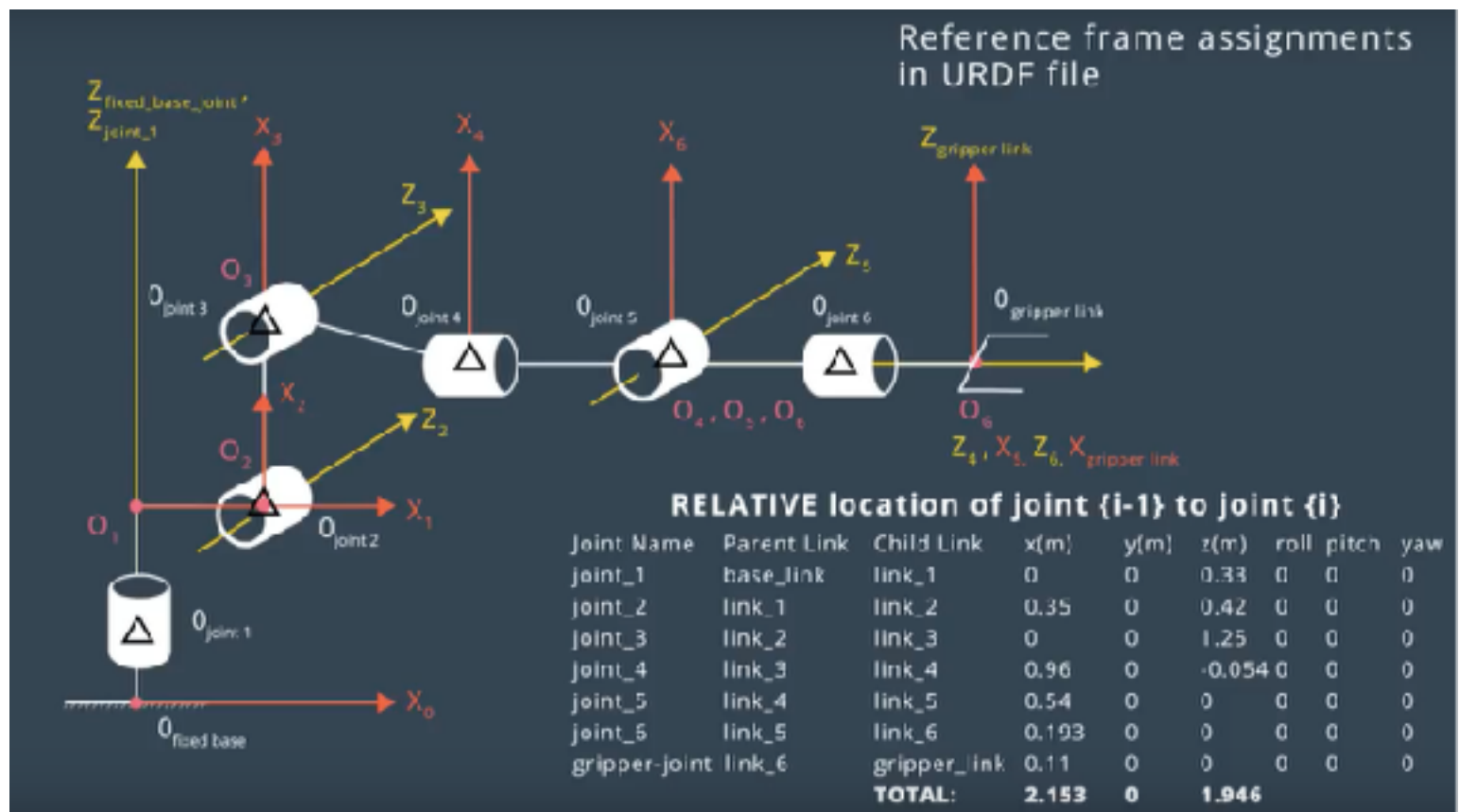
In zero configuration all angles are  $Q_i$  angles are zero only  $X_1$  and  $X_2$  are perpendicular and have an angle of  $-90$  degree.

The URDF file was used to come up with numerical values of 'a' and 'd'

From following lines we can see X, Y, Z of every joint w.r.t its parent link

<code>&lt;joint name="joint_1" type="revolute"&gt;</code>	
	<code>&lt;origin xyz="0 0 0.33" rpy="0 0 0"/&gt;</code>

The following snapshot shows how x,y and z for every joint can be collected from URDF file. Since these x,y,z are relative we need to add them as per our DH picture to come up with DH parameters



For example the  $d_1$  = distance from base\_link to link\_1 (z-axis)+ distance from link\_1 to link\_2(z-axis)

$$D_1 = 0.33 + 0.42 = 0.75$$

$A_1$  = distance of  $z_2$  to  $z_1$  along x-axis

$$A_1 = 0.35$$

Joint	$\alpha(i-1)$	$a(i-1)$	$D_i$	$Q_i$
1	0	0	0.75	$Q_1$
2	$-\pi/2$	0.35	0	$Q_2 - \pi/2$
3	0	1.25	0	$Q_3$
4	$-\pi/2$	-0.054	1.5	$Q_4$
5	$\pi/2$	0	0	$Q_5$
6	$-\pi/2$	0	0	$Q_6$
G	0	0	0.303	$Q_7$

**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.**

The transformation matrix is given as follows

$$TF = \begin{bmatrix} \cos(q), & -\sin(q), & 0, & a \\ \sin(q)\cos(\alpha), & \cos(q)\cos(\alpha), & -\sin(\alpha), & -\sin(\alpha)*d \\ \sin(q)\sin(\alpha), & \cos(q)\sin(\alpha), & \cos(\alpha), & \cos(\alpha)*d \\ 0, & 0, & 0, & 1 \end{bmatrix}$$

$$T_{0_1}: \begin{bmatrix} \cos(q_1), & -\sin(q_1), & 0, & 0 \\ \sin(q_1), & \cos(q_1), & 0, & 0 \\ 0, & 0, & 1, & 0.75 \\ 0, & 0, & 0, & 1 \end{bmatrix}$$

$$T_{1_2}: \begin{bmatrix} \sin(q_2), & \cos(q_2), & 0, & 0.35 \\ 0, & 0, & 1, & 0 \\ \cos(q_2), & -\sin(q_2), & 0, & 0 \\ 0, & 0, & 0, & 1 \end{bmatrix}$$

```

T2_3:      [cos(q3), -sin(q3), 0, 1.25]
            [sin(q3),  cos(q3), 0,   0]
            [   0,      0, 1,   0]
            [   0,      0, 0,   1]
T3_4:      [ cos(q4),  -sin(q4), 0, -0.054]
            [   0,      0, 1,   1.5]
            [-sin(q4),  -cos(q4), 0,   0]
            [   0,      0, 0,   1]

T4_5:      [cos(q5), -sin(q5), 0, 0]
            [   0,      0, -1, 0]
            [sin(q5),  cos(q5), 0, 0]
            [   0,      0, 0, 1]

T5_6:      [ cos(q6), -sin(q6), 0, 0]
            [   0,      0, 1, 0]
            [-sin(q6), -cos(q6), 0, 0]
            [   0,      0, 0, 1]

T6_G:      [1, 0, 0, 0]
            [0, 1, 0, 0]
            [0, 0, 1, 0.303]
            [0, 0, 0, 1]

```

```

T0_G = Matrix([(((sin(q1)*sin(q4) + sin(q2 + q3)*cos(q1)*cos(q4))*cos(q5) +
sin(q5)*cos(q1)*cos(q2 + q3))*cos(q6) - (-sin(q1)*cos(q4) + sin(q4)*sin(q2 +
q3)*cos(q1))*sin(q6), -((sin(q1)*sin(q4) + sin(q2 +
q3)*cos(q1)*cos(q4))*cos(q5) + sin(q5)*cos(q1)*cos(q2 + q3))*sin(q6) +
(sin(q1)*cos(q4) - sin(q4)*sin(q2 + q3)*cos(q1))*cos(q6), -(sin(q1)*sin(q4) +
sin(q2 + q3)*cos(q1)*cos(q4))*sin(q5) + cos(q1)*cos(q5)*cos(q2 + q3),
-0.303*sin(q1)*sin(q4)*sin(q5) + 1.25*sin(q2)*cos(q1) - 0.303*sin(q5)*sin(q2
+ q3)*cos(q1)*cos(q4) - 0.054*sin(q2 + q3)*cos(q1) +
0.303*cos(q1)*cos(q5)*cos(q2 + q3) + 1.5*cos(q1)*cos(q2 + q3) +
0.35*cos(q1)], [((sin(q1)*sin(q2 + q3)*cos(q4) - sin(q4)*cos(q1))*cos(q5) +
sin(q1)*sin(q5)*cos(q2 + q3))*cos(q6) - (sin(q1)*sin(q4)*sin(q2 + q3) +
cos(q1)*cos(q4))*sin(q6), -((sin(q1)*sin(q2 + q3)*cos(q4) -
sin(q4)*cos(q1))*cos(q5) + sin(q1)*sin(q5)*cos(q2 + q3))*sin(q6) -
(sin(q1)*sin(q4)*sin(q2 + q3) + cos(q1)*cos(q4))*cos(q6), -(sin(q1)*sin(q2 +
q3)*cos(q4) - sin(q4)*cos(q1))*sin(q5) + sin(q1)*cos(q5)*cos(q2 + q3),
1.25*sin(q1)*sin(q2) - 0.303*sin(q1)*sin(q5)*sin(q2 + q3)*cos(q4) -
0.054*sin(q1)*sin(q2 + q3) + 0.303*sin(q1)*cos(q5)*cos(q2 + q3) +
1.5*sin(q1)*cos(q2 + q3) + 0.35*sin(q1) + 0.303*sin(q4)*sin(q5)*cos(q1)], [-
(sin(q5)*sin(q2 + q3) - cos(q4)*cos(q5)*cos(q2 + q3))*cos(q6) -
sin(q4)*sin(q6)*cos(q2 + q3), (sin(q5)*sin(q2 + q3) - cos(q4)*cos(q5)*cos(q2
+ q3))*sin(q6) - sin(q4)*cos(q6)*cos(q2 + q3), -sin(q5)*cos(q4)*cos(q2 + q3)
- sin(q2 + q3)*cos(q5), -0.303*sin(q5)*cos(q4)*cos(q2 + q3) - 0.303*sin(q2 +

```

$q_3) \cos(q_5) - 1.5 \sin(q_2 + q_3) + 1.25 \cos(q_2) - 0.054 \cos(q_2 + q_3) + 0.75],$   
 $[0, 0, 0, 1])$

Generalized homogenous vector using end-effector pose

$$T = \left[ \begin{array}{ccc|c} & & & P_x \\ & R_T & & P_y \\ & & & P_z \\ \hline 0 & 0 & 0 & 1 \end{array} \right]$$

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

The last three joints of Kuka\_arm are revolute joints, such a design is called a spherical wrist and the common point of intersection is called the wrist center. The advantage of such a design is that it kinematically decouples the position and orientation of the end effector. Mathematically, this means that instead of solving *twelve* nonlinear equations simultaneously (one equation for each term in the first three rows of the overall homogeneous transform matrix), it is now possible to independently solve two simpler problems: first, the Cartesian coordinates of the wrist center, and then the composition of rotations to orient the end effector. Physically speaking, a six degree of freedom serial manipulator with a spherical wrist would use the first three joints to control the *position* of the wrist center while the last three joints would orient the end effector as needed.

The end effector matrix looks like follows

$$\begin{bmatrix} l_x & m_x & n_x & p_x \\ l_y & m_y & n_y & p_y \\ l_z & m_z & n_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

**Step1: We can calculate wrist centre position using following equation**

$$w_x = p_x - (d_6 + l) \cdot n_x$$

$$w_y = p_y - (d_6 + l) \cdot n_y$$

$$w_z = p_z - (d_6 + l) \cdot n_z$$

Where,

Px, Py, Pz = end-effector positions

Wx, Wy, Wz = wrist positions

d6 = from DH table

l = end-effector length

We have to calculate  $n_x$ ,  $n_y$ , and  $n_z$  to substitute in above equation to get wrist centre.

We know roll, pitch and yaw of end effector from ROS by converting quaternions

Those can be substituted in following equation to get R<sub>rpv</sub>

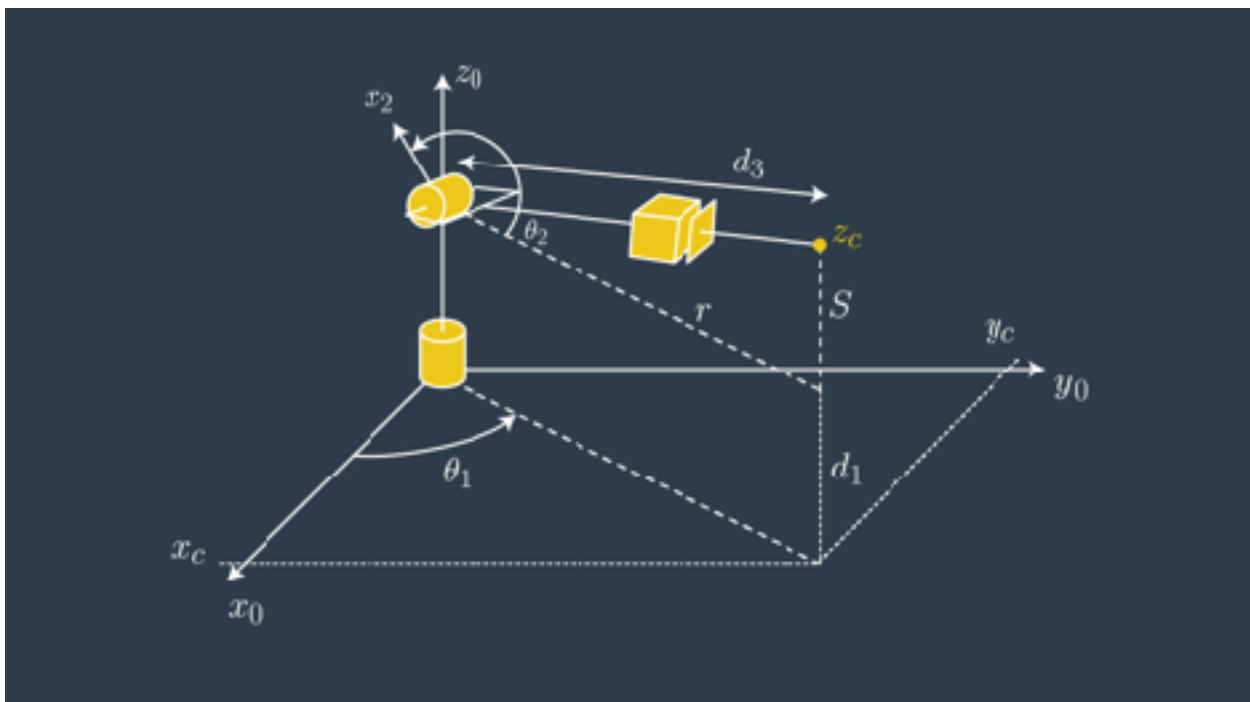
$$R_{rpv} = \text{Rot}(Z, \text{yaw}) * \text{Rot}(Y, \text{pitch}) * \text{Rot}(X, \text{roll}) * R_{\text{corr}}$$

We can extract  $n_x$ ,  $n_y$ ,  $n_z$  from R<sub>rpv</sub> matrix

This will give us  $w_x$ ,  $w_y$ ,  $w_z$  which represents wrist centre

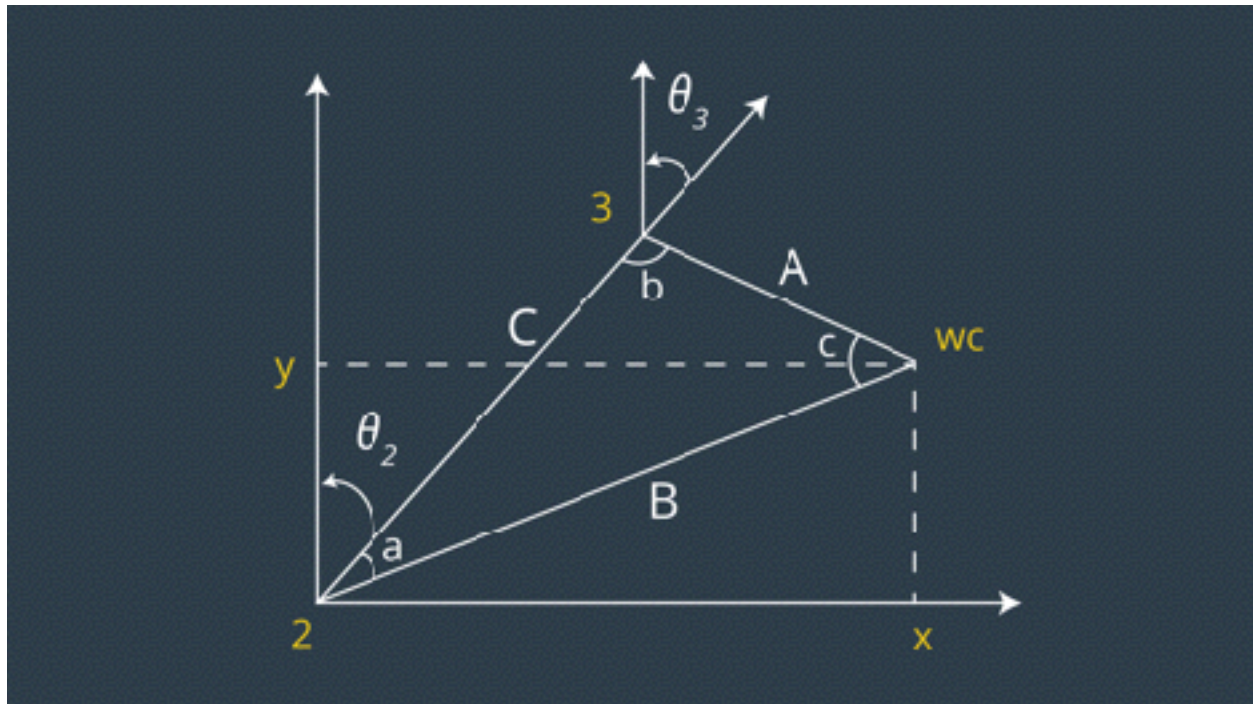
Step2: Calculate first three angles from wrist centre

$\theta_1 = \text{atan2}(w_y, w_x)$  as shown in following picture





We will use below picture to get theta2 and theta3



```

side_a = 1.50
r = sqrt(wx*wx+wy*wy) - 0.35 # a1: 0.35
side_b = sqrt((r*r) + pow(wz - 0.75, 2))
side_c = 1.25

angle_a = acos((side_b * side_b + side_c * side_c - side_a * side_a) /
(2 * side_b * side_c))

angle_b = acos((side_a * side_a + side_c * side_c - side_b * side_b) /
(2 * side_a * side_c))

angle_c = acos((side_a * side_a + side_b * side_b - side_c * side_c) /
(2 * side_a * side_b))

theta2 = pi/2 - angle_a - atan2(wz - 0.75, r)
theta3 = pi/2 - (angle_b + 0.036) # 0.036 is for sag in link4

```

### Step3: Calculate last three angles which is Inverse orientation

The resultant rotation is of the following form f

$$R_{0\_6} = R_{0\_1} * R_{1\_2} * R_{2\_3} * R_{3\_4} * R_{4\_5} * R_{5\_6}$$

Since the overall RPY (**R**oll **P**itch **Y**aw) rotation between `base_link` and `gripper_link` must be equal to the product of individual rotations between respective links, following holds true:

$$R_{0\_6} = R_{rpy}$$

where,

$R_{rpy}$  = Homogeneous RPY rotation between `base_link` and `gripper_link` as calculated above.

We can substitute the values we calculated for joints 1 to 3 in their respective individual rotation matrices and pre-multiply both sides of the above equation by  $\text{inv}(R_{0\_3})$  which leads to:

$$R_{3\_6} = \text{inv}(R_{0\_3}) * R_{rpy}$$

$$\theta_4 = \text{atan2}(R_{3\_6}[2,2], -R_{3\_6}[0,2])$$

$$\theta_5 = \text{atan2}(\sqrt{R_{3\_6}[0,2]^2 + R_{3\_6}[2,2]^2}, R_{3\_6}[1,2])$$

$$\theta_6 = \text{atan2}(-R_{3\_6}[1,1], R_{3\_6}[1,0])$$

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

### Step1: DH parameters

```
37         # Create Modified DH parameters
38         s = {alp0: 0, a0: 0, d1: 0.75, q1: q1,
39              alp1: -pi/2, a1: 0.35, d2: 0, q2: q2-pi/2,
40              alp2: 0, a2: 1.25, d3: 0, q3: q3,
41              }
42         ##Not needed## alp3: -pi/2, a3: -0.054, d4: 1.50,
q4: q4,
43         ##Not needed## alp4: pi/2, a4: 0, d5: 0,
q5: q5,
44         ##Not needed## alp5: -pi/2, a5: 0, d6: 0,
q6: q6,
45         ##Not needed## alp6: 0, a6: 0, d7: 0.303,
q7: 0}
46
```

### Step2: Individual Transform matrix

```
48         # Define Modified DH Transformation matrix
49         def TF_Matrix(alp, a, d, q):
50             TF = Matrix([[cos(q), -sin(q),
0, a,
51                          [sin(q)*cos(alp), cos(q)*cos(alp),
-sin(alp), -sin(alp)*d],
52                          [sin(q)*sin(alp), cos(q)*sin(alp),
cos(alp), cos(alp)*d],
53                          [0, 0,
0, 1]])
54             return TF
55
56
57         # Create individual transformation matrices
58         T0_1 = TF_Matrix(alp0, a0, d1, q1).subs(s)
59         T1_2 = TF_Matrix(alp1, a1, d2, q2).subs(s)
60         T2_3 = TF_Matrix(alp2, a2, d3, q3).subs(s)
61         ##Not needed ##T3_4 = TF_Matrix(alp3, a3, d4,
q4).subs(s)
62         ##Not needed ##T4_5 = TF_Matrix(alp4, a4, d5,
q5).subs(s)
63         ##Not needed ##T5_6 = TF_Matrix(alp5, a5, d6,
q6).subs(s)
```

```

64      ##Not needed ##T6_G = TF_Matrix(alp6, a6, d7,
q7).subs(s)
65

```

Step3: Rotation matrix for end effector with correction between DH and URDF

```

72      r, p, y = symbols('r p y')
73      ROT_x = Matrix([[1,      0,      0],
74                      [0, cos(r), -sin(r)],
75                      [0, sin(r),  cos(r)]])
76
77      ROT_y = Matrix([[cos(p),  0,  sin(p)],
78                      [0,      1,      0],
79                      [-sin(p), 0,  cos(p)]])
80
81      ROT_z = Matrix([[cos(y), -sin(y),  0],
82                      [sin(y),  cos(y),  0],
83                      [      0,      0,  1]])
84
85
86      R0_3 = simplify(T0_1[0:3,0:3] * T1_2[0:3,0:3] *
T2_3[0:3,0:3])
87      ROT_G = simplify(ROT_z * ROT_y * ROT_x)
88      ROT_error = ROT_z.subs(y, radians(180)) * ROT_y.subs(p,
radians(-90))
89      ROT_G = simplify(ROT_G * ROT_error)

```

Step4: Wrist centre from end effector pose

```

117      wx = px - 0.303 * ROT_G_eval[0,2]
118      wy = py - 0.303 * ROT_G_eval[1,2]
119      wz = pz - 0.303 * ROT_G_eval[2,2]

```

Step5: Calculate first three angles

```

122      # Calculate joint angles using Geometric IK method
123      theta1 = atan2(wy, wx)
124      side_a = 1.50
125      r = sqrt(wx*wx+wy*wy) - 0.35 # a1: 0.35
126      side_b = sqrt((r*r) + pow(wz - 0.75, 2))
127      side_c = 1.25
128
129      angle_a = acos((side_b * side_b + side_c * side_c -
side_a * side_a) / (2 * side_b * side_c))

```

```

130         angle_b = acos((side_a * side_a + side_c * side_c -
side_b * side_b) / (2 * side_a * side_c))
131         angle_c = acos((side_a * side_a + side_b * side_b -
side_c * side_c) / (2 * side_a * side_b))
132
133
134         theta2 = pi/2 - angle_a - atan2(wz - 0.75, r)
135         theta3 = pi/2 - (angle_b + 0.036) # 0.036 is for sag
in link4
136

```

Step6: Calculate last three angles

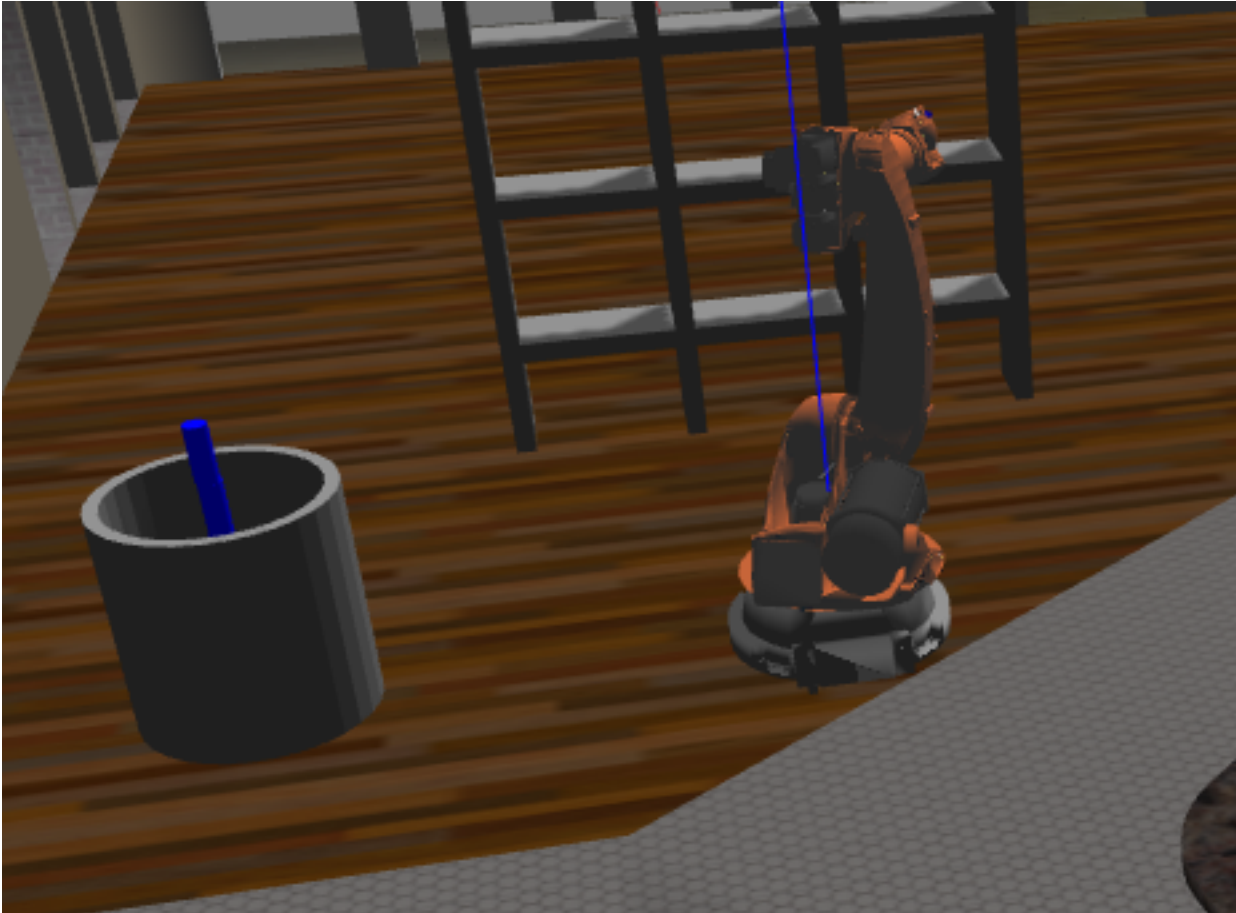
```

137         R0_3_eval = R0_3.evalf(subs={'q1':theta1, 'q2':theta2,
'q3':theta3})
138
139         R3_6 = R0_3_eval.inv("LU") * ROT_G_eval
140
141         theta5 = atan2(sqrt(R3_6[0,2]*R3_6[0,2] +
R3_6[2,2]*R3_6[2,2]), R3_6[1,2])
142
143         if (theta5 > pi):
144             theta4 = atan2(-R3_6[2,2], R3_6[0,2])
145             theta6 = atan2(R3_6[1,1], -R3_6[1,0])
146         else:
147             theta4 = atan2(R3_6[2,2], -R3_6[0,2])
148             theta6 = atan2(-R3_6[1,1], R3_6[1,0])

```

theta 5 can have multiple solutions and based on that we decide values of theta4 and theta6

## Pick and Place in action



## Observations:

- Overall the runtime for the code is very slow. Need to investigate if pre-calculation of some items help. I have removed most of the things out of loop so they don't get executed again. Also reduced R0\_6 to R0\_3 as the other calculation was redundant.
- Removed simplification of matrix as that was taking long time.

- There are too many rotations of the joint 4. It keeps rotating back and forth so many time which is slowing down the overall execution of the code
- There was one case where the gripper didn't grip the can. Most of the cases got fixed by adding delay of 2.0s in trajectory cpp code. One case still shows up which needs further investigation.
- Gazebo and Rviz sometime crashes need to understand why it happens; softwares or libraries out of date.