Project: Pick and place

1. Writeup:

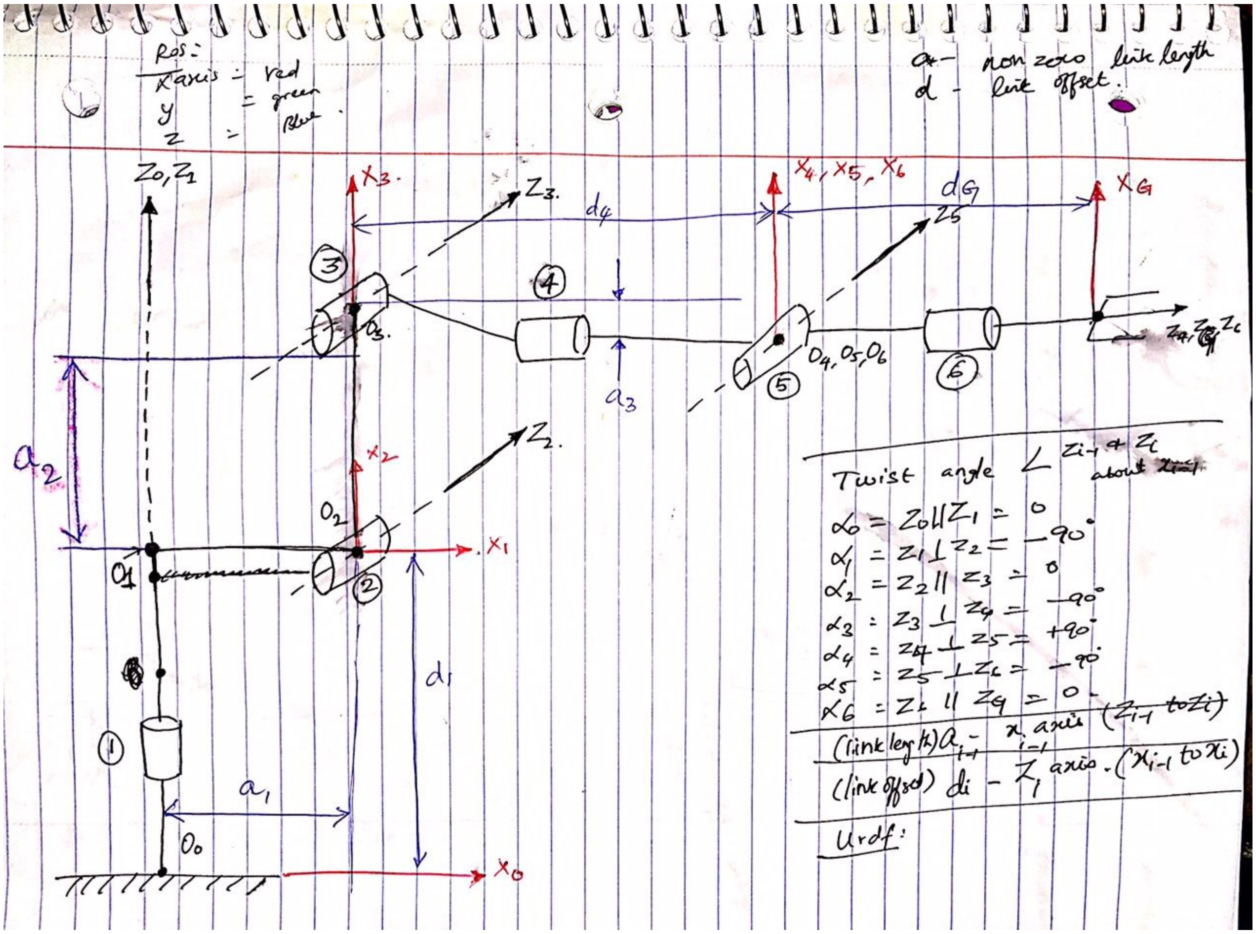
Criteria 1:

Criteria	Meets specifications	Result
Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your write up as markdown or pdf. Here is a template writeup for this project you can use as a guide and a starting point.	The write up / README should include a statement and supporting figures / images that explain how each rubric item was addressed, and specifically where in the code each step was handled.	Criteria met Writeup has statements describing how the rubric is addressed Figures, code extracts and calculations are provided to describe the code.

2. Kinematic analysis:

Criteria 1:

Criteria	Meets specifications	Result
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.	Your writeup should contain a DH parameter table with proper notations and description about how you obtained the table. Make sure to use the modified DH parameters discussed in [this lesson](). Please add an annotated figure of the robot with proper link assignments and joint rotations (Example figure provided in the write up template). It is strongly recommended that you use pen and paper to create this figure to get a better understanding of the robot kinematics.	Criteria met DH table and the method used to extract the values from the URDF file is described Annotated figure shown below



The above figure was drawn using the procedure described in lesson 3_10

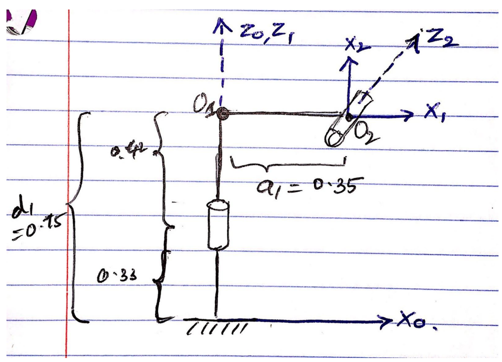
Extracting DH parameters from kr210.urdf.xacro file:

DH convention

- α_{i-1} (twist angle) = angle between $\hat{Z}_{i-1}$ and $\hat{Z}_i$ measured about $\hat{X}_{i-1}$ in a right-hand sense.
- a_{i-1} (link length) = distance from $\hat{Z}_{i-1}$ to $\hat{Z}_i$ measured along $\hat{X}_{i-1}$ where $\hat{X}_{i-1}$ is perpendicular to both $\hat{Z}_{i-1}$ to $\hat{Z}_i$
- d_i (link offset) = signed distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured along $\hat{Z}_i$. Note that this quantity will be a variable in the case of prismatic joints.
- θ_i (joint angle) = angle between $\hat{X}_{i-1}$ to $\hat{X}_i$ measured about $\hat{Z}_i$ in a right-hand sense. Note that this quantity will be a variable in the case of a revolute joint.

Example: extracting a1 and d1 from the urdf file

```
<joint name="joint_1" type="revolute">
  <origin xyz="0 0 0.33" rpy="0 0 0"/>
  <parent link="base_link"/>
  <child link="link_1"/>
  <axis xyz="0 0 1"/>
  <limit lower="${-185*deg}" upper="${185*deg}" effort="300" velocity="${123*deg}"/>
</joint>
<joint name="joint_2" type="revolute">
  <origin xyz="0.35 0 0.42" rpy="0 0 0"/>
  <parent link="link_1"/>
  <child link="link_2"/>
  <axis xyz="0 1 0"/>
  <limit lower="${-45*deg}" upper="${85*deg}" effort="300" velocity="${115*deg}"/>
</joint>
```



The relevant x,y,z values can be referred from the urdf file and mapped to equivalent links in the figure

DH Parameter table

alpha0: 0,	a0: 0,	d1: 0.75,	
alpha1: -pi/2,	a1: 0.35,	d2: 0,	q2: q2 - pi/2,
alpha2: 0,	a2: 1.25,	d3: 0,	
alpha3: -pi/2,	a3: -0.054,	d4: 1.5,	
alpha4: pi/2,	a4: 0,	d5: 0,	
alpha5: -pi/2,	a5: 0,	d6: 0,	
alpha6: 0,	a6: 0,	d7: 0.303,	q7: 0

Relevant Code extract:

```
        # Define DH param symbols
        q1, q2, q3, q4, q5, q6, q7 = symbols('q1:8') #Theta i
        d1, d2, d3, d4, d5, d6, d7 = symbols('d1:8')
        a0, a1, a2, a3, a4, a5, a6 = symbols('a0:7')
        alpha0, alpha1, alpha2, alpha3, alpha4, alpha5, alpha6 = symbols('alpha0:7')


        # Create Modified DH parameters

        s = {alpha0: 0, a0:   0,    d1: 0.75,
        alpha1: -np.pi/2,  a1: 0.35,   d2: 0,      q2: q2 - np.pi/2,
        alpha2: 0,      a2: 1.25,   d3: 0,
        alpha3: -np.pi/2,  a3: -0.054, d4: 1.5,
        alpha4: np.pi/2,   a4: 0,      d5: 0,
        alpha5: -np.pi/2,  a5: 0,      d6: 0,
        alpha6: 0,      a6: 0,      d7: 0.303,  q7: 0}
```

Criteria 2:

Criteria	Meets specifications	Result
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.	Your writeup should contain individual transform matrices about each joint using the DH table and a homogeneous transform matrix from base_link to gripper_link using only the position and orientation of the gripper_link. These matrices can be created using any software of your choice or hand written. Also include an explanation on how you created these matrices.	Criteria met Individual transform matrices about each joint is described below in a generic format Tn_n+1 matrix Homogenous transform matrix from base to gripper link is provided in T_total matrix Homogeneous transform matrix from base_link to gripper_link using only the position and orientation of the gripper_link is also described below in the Euler angle method

Individual transformation matrices:

Create individual transformation matrices

Tn_n+1 = Matrix			
cos(q_n)	-sin(q_n)	0	a_{n-1}
sin(q_n)*cos($alpha_{n-1}$)	os(q_n)*cos($alpha_{n-1}$)	sin($alpha_{n-1}$)	-sin($alpha_{n-1}$)*d_n
sin(q_n)*sin($alpha_{n-1}$)	cos(q_n)*sin($alpha_{n-1}$)	cos($alpha_{n-1}$)	cos($alpha_{n-1}$)*d_n
0	0	0	1

Individual transformation matrices for each link is calculated by using the above transformation matrix and substituting q, alpha, d and a from the DH parameter table for the specific ‘n’ index

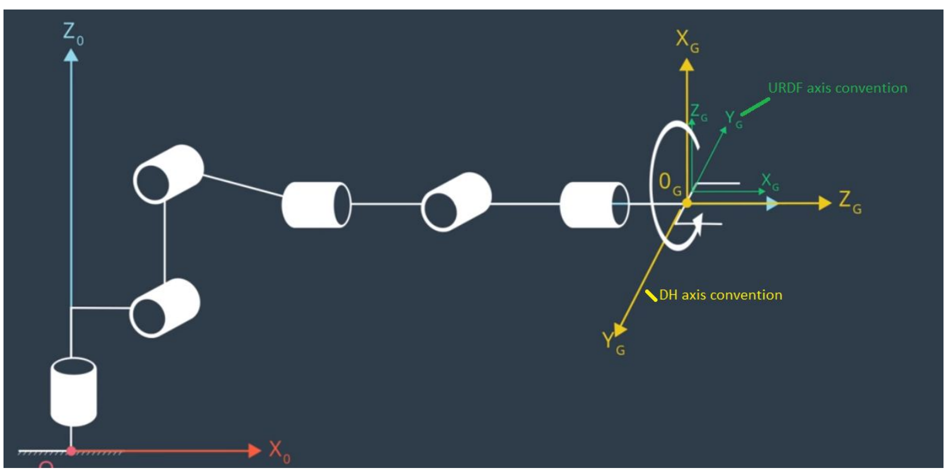
T0_n = T0_n.subs(s)

Homogenous transform matrix from base to gripper link:

Calculate the overall transformation matrix by incrementally calculating T0_n all the way to T0_G (transformation matrix from base link to the gripper link)

```
T0_2 = simplify(T0_1 * T1_2)
T0_3 = simplify(T0_2 * T2_3)
T0_4 = simplify(T0_3 * T3_4)
T0_5 = simplify(T0_4 * T4_5)
T0_6 = simplify(T0_5 * T5_6)
T0_G = simplify(T0_6 * T6_G)
```

This transformation matrix has to be corrected because it is following the DH convention. The figure below shows the difference between the DH and URDF axis convention. Hence we rotate by z axis by pi and y axis by -pi/2 to do the conversion. This is achieved by multiplying the appropriate rotation matrix with the transformation matrix T0_G




```
R_z =   Matrix
        ([[  cos(pi),    -sin(pi),    0,          0],
          [  sin(pi),     cos(pi),    0,          0],
          [  0,           0,          1,          0],
          [  0,           0,          0,          1]])

R_y =   Matrix
        ([[  cos(-pi/2),  0,          sin(-pi/2),  0],
```

```
[   0,              0,              1,              0],
[   -sin(-pi/2), 0,              cos(-pi/2),  0],
[   0,              0,              0,              1]])
```

R_corr = simplify(R_z * R_y)
T_total = simplify(T0_G * R_corr)

T_total matrix derived from the above calculation

T_total Matrix			
-(s(q1)*s(q4) + s(q2 + q3)*c(q1)*c(q4))*s(q5) + c(q1)*c(q5)*c(q2 + q3)	0	sqrt(2)*(s(q1)*s(q4)*s(q6 + np.pi/4)*c(q5) - s(q1)*c(q4)*c(q6 + np.pi/4) + s(q4)*s(q2 + q3)*c(q1)*c(q6 + np.pi/4) + s(q5)*s(q6 + np.pi/4)*c(q1)*c(q2 + q3) + s(q2 + q3)*s(q6 + np.pi/4)*c(q1)*c(q4)*c(q5))	-0.303*(s(q1)*s(q4) + s(q2 + q3)*c(q1)*c(q4))*s(q5) + (1.25*s(q2) - 0.054*s(q2 + q3) + 1.5*c(q2 + q3) + 0.35)*c(q1) + 0.303*c(q1)*c(q5)*c(q2 + q3)
-(s(q1)*s(q2 + q3)*c(q4) - s(q4)*c(q1))*s(q5) + s(q1)*c(q5)*c(q2 + q3)	0	sqrt(2)*(s(q1)*s(q4)*s(q2 + q3)*c(q6 + np.pi/4) + s(q1)*s(q5)*s(q6 + np.pi/4)*c(q2 + q3) + s(q1)*s(q2 + q3)*s(q6 + np.pi/4)*c(q4)*c(q5) - s(q4)*s(q6 + np.pi/4)*c(q1)*c(q5) + c(q1)*c(q4)*c(q6 + np.pi/4))	-0.303*(s(q1)*s(q2 + q3)*c(q4) - s(q4)*c(q1))*s(q5) + (1.25*s(q2) - 0.054*s(q2 + q3) + 1.5*c(q2 + q3) + 0.35)*s(q1) + 0.303*s(q1)*c(q5)*c(q2 + q3)
-s(q5)*c(q4)*c(q2 + q3) - s(q2 + q3)*c(q5)	0	sqrt(2)*(s(q4)*c(q2 + q3)*c(q6 + np.pi/4) - s(q5)*s(q2 + q3)*s(q6 + np.pi/4) + s(q6 + np.pi/4)*c(q4)*c(q5)*c(q2 + q3))	-0.303*s(q5)*c(q4)*c(q2 + q3) - 0.303*s(q2 + q3)*c(q5) - 1.5*s(q2 + q3) + 1.25*c(q2) - 0.054*c(q2 + q3) + 0.75
0	0	0	1

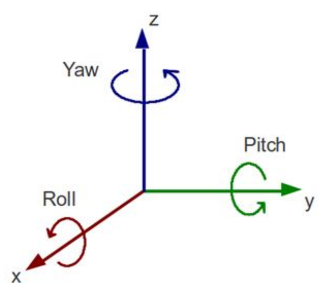
In the above matrix after substituting q1 to 6, we get the evaluated matrix in the format below where the rightmost column gives the coordinates of the gripper as P_x, P_y, P_z

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

Homogeneous transform matrix from base_link to gripper_link using only the position and orientation of the gripper_link

Euler angle method to determine the transformation matrix using the gripper pose in terms of roll, pitch and yaw

$$\begin{aligned} {}^A_BR_{ZYX} &= R_Z(\alpha)R_Y(\beta)R_X(\gamma) \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix} \end{aligned}$$



Using the above convention, alpha is yaw, beta is pitch, gamma is roll
The homogenous transformation matrix is calculated by substituting R_{ZYX} in R_T below. Px, Py, Pz are the gripper co-ordinates.

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

Criteria 3:

Criteria	Meets specifications	Result
Decouple Inverse Kinematics problem into Inverse Position Kinematics and inverse Orientation Kinematics; doing so derive the equations to calculate all individual joint angles.	Based on the geometric Inverse Kinematics method described here, breakdown the IK problem into Position and Orientation problems. Derive the equations for individual joint angles. Your write up must contain details about the steps you took to arrive at those equations. Add figures where necessary. If any given joint has multiple solutions, select the best solution and provide explanation about your choice (Hint: Observe the active robot workspace in this project and the fact that some joints have physical limits).	Criteria met Steps taken to calculate theta 1 to 6 is described below

Theta 1,2,3 calculation based on closed form solution

Determining the wrist position:

Extracting end effector position from the robot model

```
            # Extract end-effector position and orientation from request
            # px,py,pz = end-effector position
            # roll, pitch, yaw = end-effector orientation
            px = req.poses[x].position.x
            py = req.poses[x].position.y
            pz = req.poses[x].position.z

            (roll, pitch, yaw) = tf.transformations.euler_from_quaternion(
                [req.poses[x].orientation.x, req.poses[x].orientation.y,
                    req.poses[x].orientation.z, req.poses[x].orientation.w])
```

Calculating wrist position from the roll pitch yaw information from the robot model
Rotation matrix Rrpy is calculated from the following matrix, substituting alpha beta gamma for yaw, pitch, roll respectively.

$$
\begin{aligned}
{}^{A}_{B}R_{ZYX} &= R_Z(\alpha)R_Y(\beta)R_X(\gamma) \\
&= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}
\end{aligned}
$$

```python
alpha = yaw
beta = pitch
gamma = roll

# Calculate joint angles using Geometric IK method
eel = 0.303 #eel - end effector length obtained from d7 in the DH table


#Calculate the total rotation from base link to the end effector using the roll pitch yaw values
Rrpy_uncorr = Matrix([[    cos(alpha)*cos(beta),   cos(alpha)*sin(beta)*sin(gamma) - sin(alpha)*cos(gamma),    cos(alpha)*sin(beta)*cos(gam
                      [   sin(alpha)*cos(beta),   sin(alpha)*sin(beta)*sin(gamma) + cos(alpha)*cos(gamma),    sin(alpha)*sin(beta)*cos(gamma) - cos(
                      [   -sin(beta),             cos(beta)*sin(gamma),                                      cos(beta)*cos(gamma)]])


#No correction was applied. We continue to use the Rrpy based on RPY provided in URDF convention
Rrpy = Rrpy_uncorr

#Rrpy has three columns l,m,n which are the orthonormal vectors representing the end effector orientation.  Since we are following the URDF

lx = Rrpy[0,0]
ly = Rrpy[1,0]
lz = Rrpy[2,0]
```

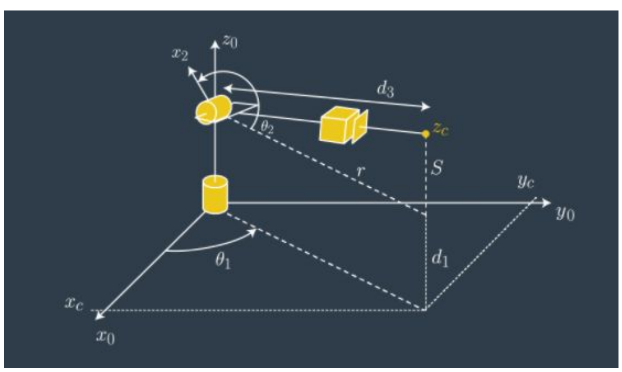
The resulting rotation matrix contain the orthonormal vectors representing the end effector orientation.

$$\begin{bmatrix} l_x & m_x & n_x \\ l_y & m_y & n_y \\ l_z & m_z & n_z \end{bmatrix}$$

The gripper is oriented in the x axis in the robot URDF convention. Hence we translate in the x axis thereby using the l column

```python
d6 = 0 #Obtained from the DH parameter table

#Calculate the wrist positions
wx = px - (d6 + eel) * lx
wy = py - (d6 + eel) * ly
wz = pz - (d6 + eel) * lz
```

Theta 1 calculation:



Note: Figure applies to theta 1 calculation only

$$\theta_1 = atan2(y_c, x_c)$$

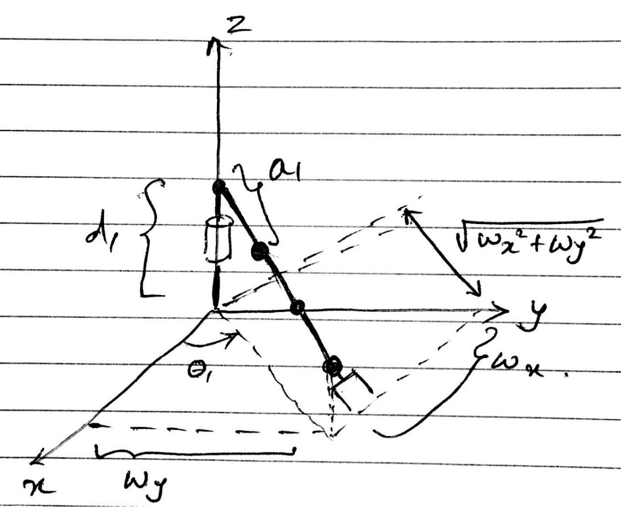
Theta 3 calculation:

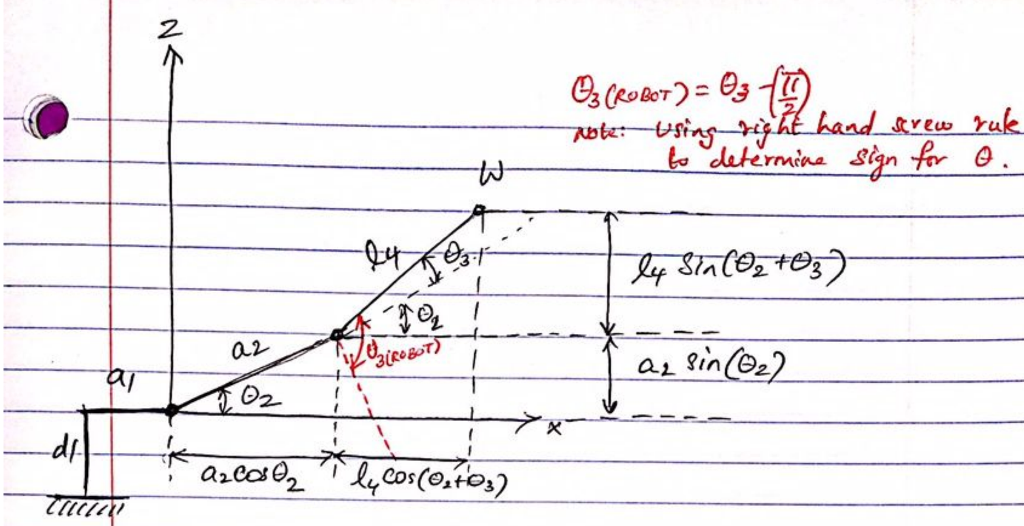
To calculate theta2,3 we assume theta1 as zero. So the robot will be aligned along the XZ plane

compensating for link 1.

$$w_{x\_adj} = \sqrt{w_x^2 + w_y^2} - a_1$$

$$w_{z\_adj} = w_z - d_1$$





$$w_x = a_2 c\theta_2 + l_4 c\theta_{12}.$$

$$w_z = a_2 s\theta_2 + l_4 s\theta_{12}$$

where, $c\theta_{12} = c\theta_2 c\theta_3 - s\theta_2 s\theta_3.$

$s\theta_{12} = s\theta_2 c\theta_3 + c\theta_2 s\theta_3.$

Calculating θ_3.

$$w_x^2 = a_2^2 c^2\theta_2 + l_4^2 c^2\theta_{12} + 2a_2 l_4 c\theta_2 c\theta_{12}$$

$$w_z^2 = a_2^2 s^2\theta_2 + l_4^2 s^2\theta_{12} + 2a_2 l_4 s\theta_2 s\theta_{12}$$

$$w_x^2 + w_z^2 = a_2^2(\underbrace{c^2\theta_2 + s^2\theta_2}_{1}) + l_4^2(\underbrace{c^2\theta_{12} + s^2\theta_{12}}_{1}) + 2a_2 l_4 (c\theta_2 c\theta_{12} + s\theta_2 s\theta_{12}).$$

(*) $c^2\theta_2 + s^2\theta_2 = 1$

$$= a_2^2 + l_4^2 + 2a_2 l_4 (c\theta_2 [c\theta_2 c\theta_3 - s\theta_2 s\theta_3] + s\theta_2 [s\theta_2 c\theta_3 + c\theta_2 s\theta_3])$$

$$= a_2^2 + l_4^2 + 2a_2 l_4 (c^2\theta_2 c\theta_3 - c\theta_2 s\theta_2 s\theta_3 + s^2\theta_2 c\theta_3 + c\theta_2 s\theta_2 \cdot s\theta_3)$$

$$= a_2^2 + l_4^2 + 2a_2 l_4 (c\theta_3 [\underbrace{c^2\theta_2 + s^2\theta_2}_{1}])$$

$$= a_2^2 + l_4^2 + 2a_2 l_4 c\theta_3.$$

$$c\theta_3 = \boxed{\frac{w_x^2 + w_z^2 - a_2^2 - l_4^2}{2 a_2 l_4}}$$ stuff 1

From (*) $c^2\theta + s^2\theta = 1$

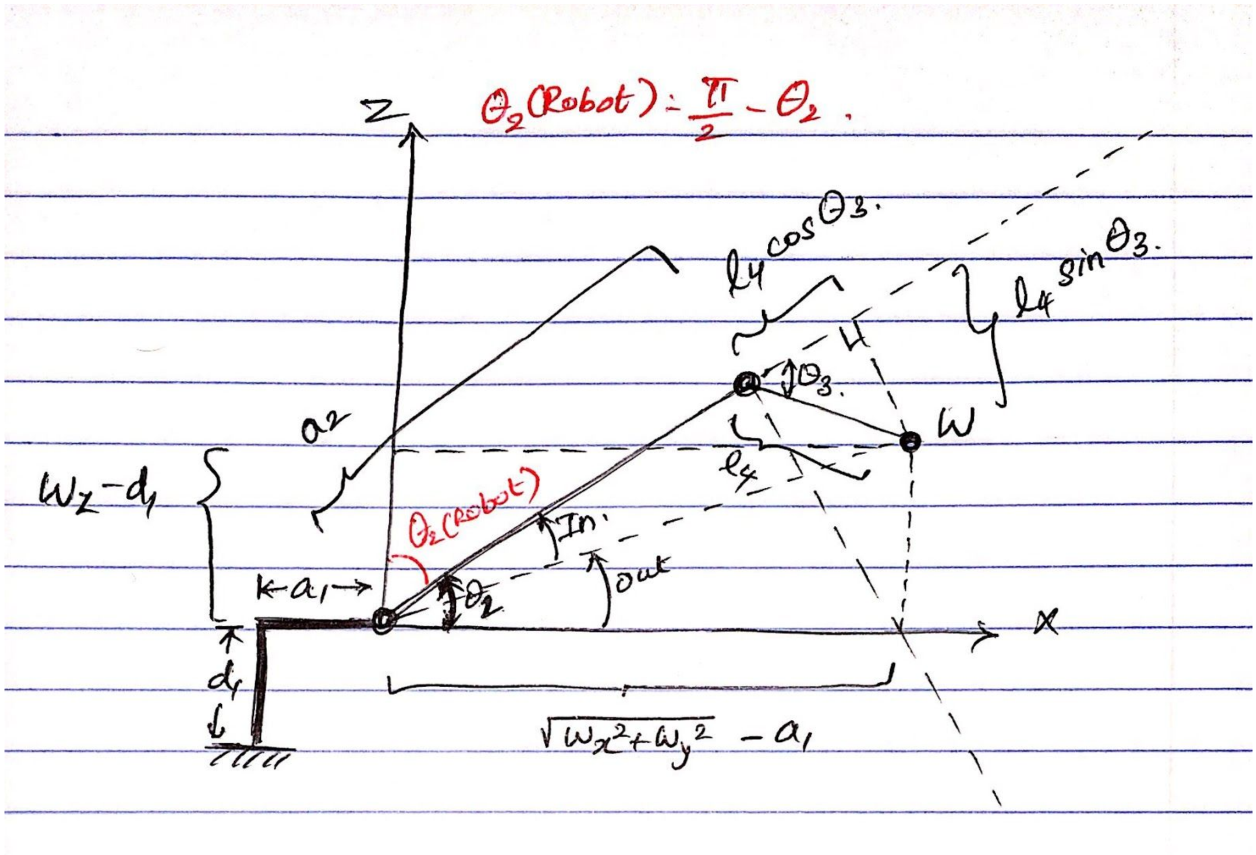
$$s\theta_3 = \pm \sqrt{1 - \text{stuff}_1^2}$$

$$\boxed{\theta_3 = \text{atan2}\left(\pm\sqrt{1 - \text{stuff}_1^2},\ \text{stuff 1}\right)}$$

Θ_3 in the picture refers to theta3_calc variable in the code.
Theta3 in the code is Θ_3(robot) in the figure above

Theta 2 calculation:

To simplify the calculation we orient joint 3 as shown in the picture below:



$$In = atan2(l_4 \sin(\theta_3),\ a_2 + l_4 \cos\theta_3)$$

$$Out = atan2(w_z - d_1,\ \sqrt{w_x^2 + w_y^2} - a_1)$$

Θ_2 = in + out

Θ_2 in the picture refers to theta2_calc variable in the code.

Theta2 in the code is Θ_2(robot) in the figure

Theta 4,5,6 calculation:

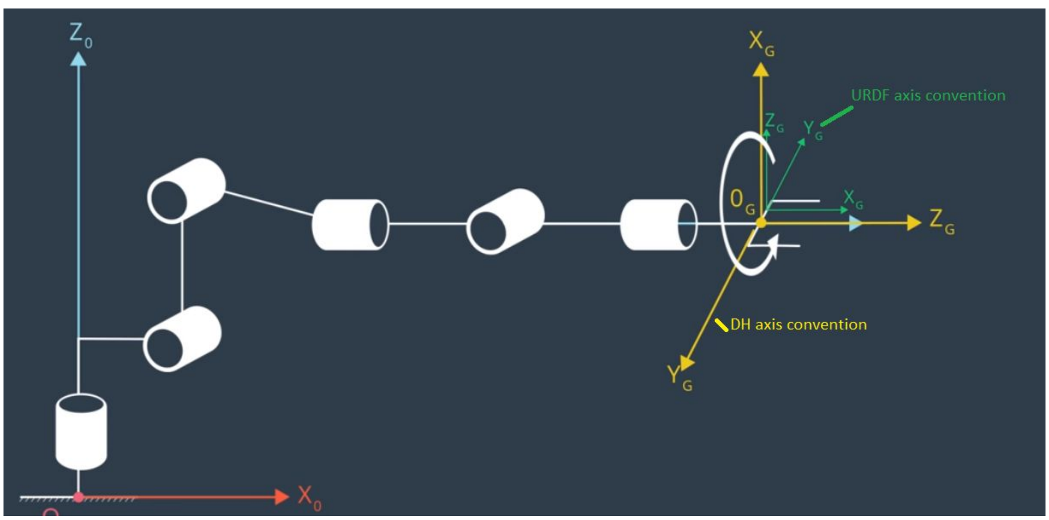
Using the end effector pose we can calculate the transformation matrix from the base link to the end effector using the following matrix

$$
{}^{A}_{B}R_{ZYX} = R_Z(\alpha)R_Y(\beta)R_X(\gamma)
$$

$$
= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}
$$

$$R_{RPY} = R_{ZYX} = R_{0\_G}$$

This matrix obtained from the end effector pose will follow the URDF convention. Hence, we need to convert to the DH convention to apply the values from the DH table. The difference in axis convention is shown in this figure below.



To convert to the DH convention we rotate by -pi/2 about the y axis and by pi about the x axis. The rotation matrices are listed below.

```
R_y =   Matrix([[ cos(-pi/2),   0,  sin(-pi/2)],
             [  0,            1,  0         ],
             [  -sin(-pi/2),  0,  cos(-pi/2)]])

R_x = Matrix([[ 1,             0,        0],
            [ 0,        cos(pi), -sin(pi)],
            [ 0,        sin(pi),  cos(pi)]])

R_corr = simplify(R_y * R_x)
```

From the transformation matrices calculated incrementally for each joint, we obtain R0_3. (Refer Tn_n+1 matrix in previous criteria 2). R0_3 matrix only has theta 2 and theta 3 which are known arguments. Hence a numerical value can be obtained.

```
R0_3_eval = Matrix([
        [sin(theta2 + theta3)*cos(theta1), cos(theta1)*cos(theta2 + theta3), -sin(theta1)],
        [sin(theta1)*sin(theta2 + theta3), sin(theta1)*cos(theta2 + theta3),  cos(theta1)],
        [       cos(theta2 + theta3),        -sin(theta2 + theta3),        0]])
```

The numerical value of R_{3_6} can be calculated from
$R_{3\_6} = inv(R_{0\_3}) * R_{RPY}$

From the transformation matrix Tn_n+1, we can incrementally obtain the symbolic matrix for R_{3_6} in terms of theta4,5,6. This calculation is performed separately in a different script because the calculation takes a lot of computation time and cannot be calculated at run time.

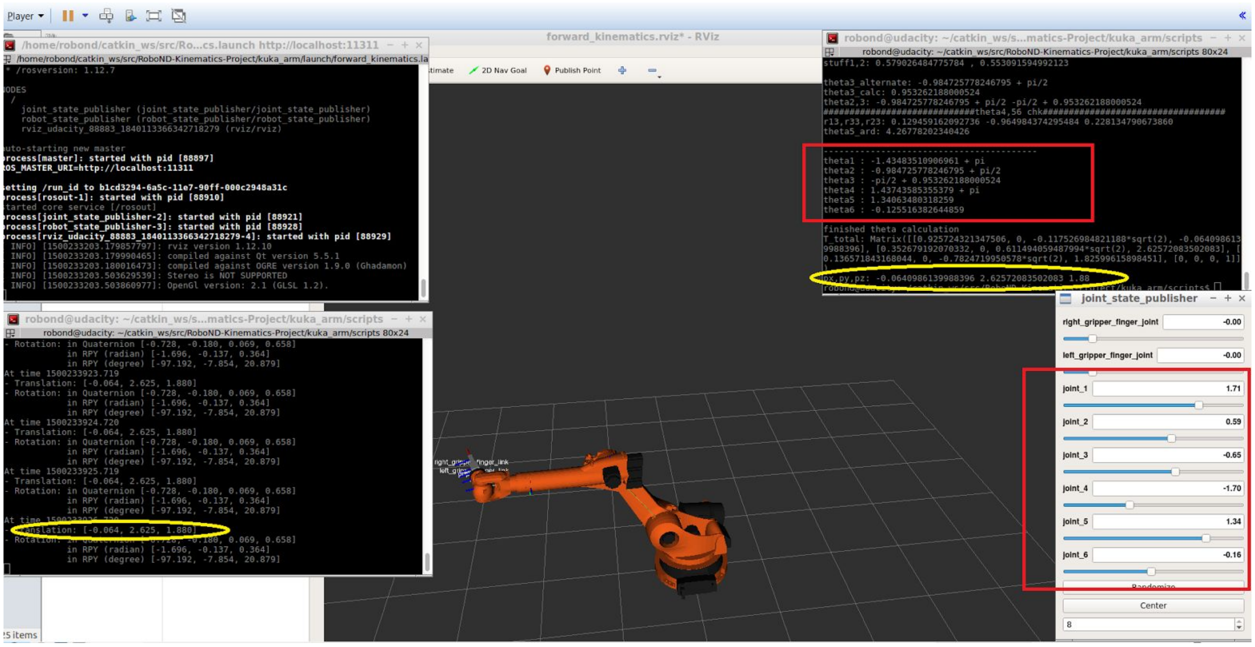
```
T3_6 = simplify(T3_4*T4_5*T5_6)

R3_6 = T3_6[0:3,0:3]
```

R3_6 Matrix		
-sin(q4)*sin(q6) + cos(q4)*cos(q5)*cos(q6)	-sin(q4)*cos(q6) - sin(q6)*cos(q4)*cos(q5)	-sin(q5)*cos(q4)
sin(q5)*cos(q6)	-sin(q5)*sin(q6)	cos(q5)
-sin(q4)*cos(q5)*cos(q6) - sin(q6)*cos(q4)	sin(q4)*sin(q6)*cos(q5) - cos(q4)*cos(q6)	sin(q4)*sin(q5)

From the above matrix we can calculate theta 4,5,6 by the following formula:

theta6 = atan2(-r22,r21)
theta5 = atan2(sqrt(r13**2 + r33**2),r23)
theta4 = atan2(-r33,r13) + pi

Using forward kinematics demo and tf_echo to check theta calculation



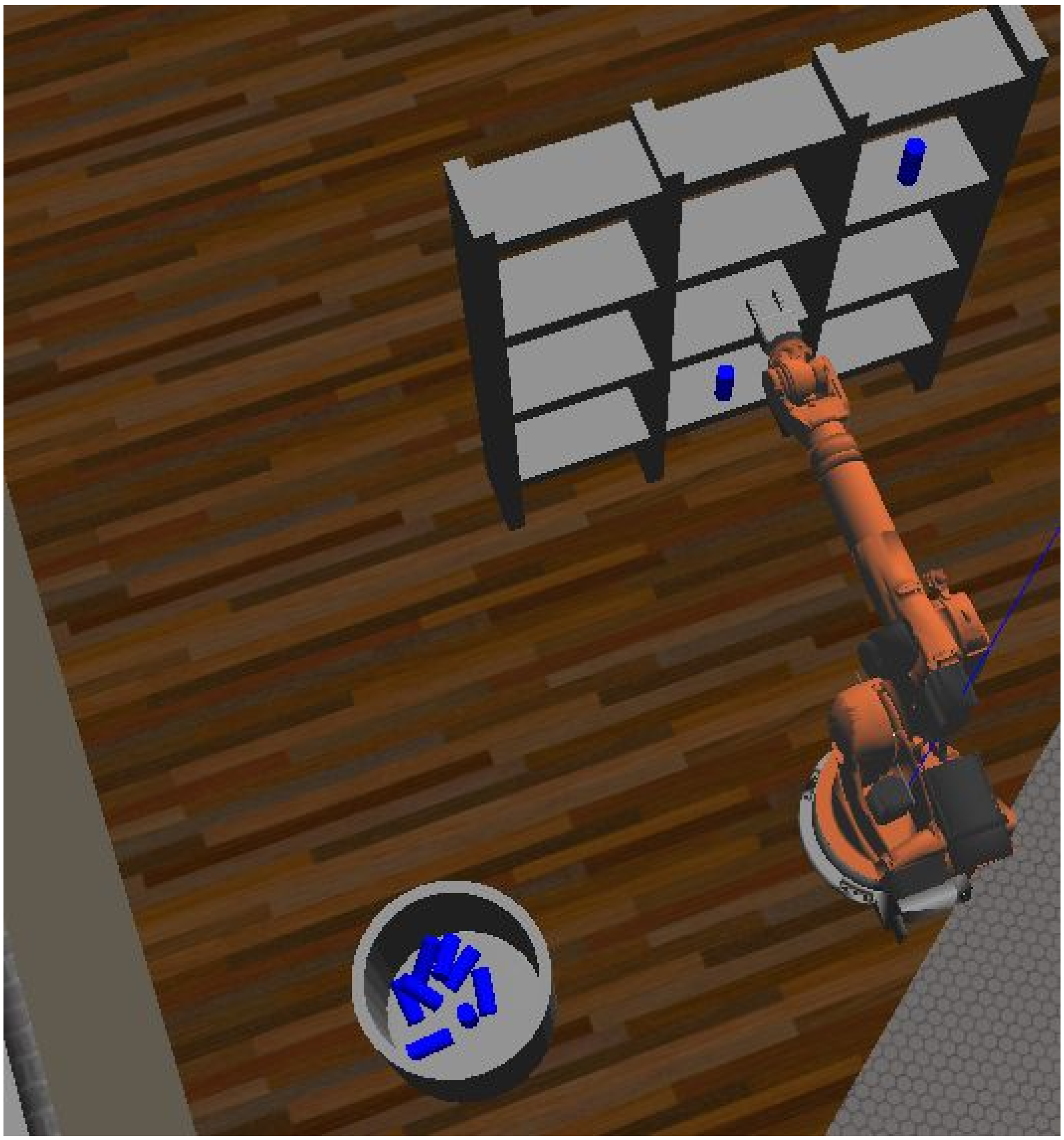
3. Project Implementation

Criteria 3:

Criteria	Meets specifications	Result
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.	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.	Criteria met IK_server.py file is attached Robot successfully completed 8/10 pick and place cycles Error plot from a subset of data is provided below

Complete 8/10 pick and place cycles:

- The robot completed 8 pick and place successfully.
- The robot failed to grip the cylinder in one cycle
- In the tenth cycle, the IK_server.py returned “no valid poses received”



Error calculated by substituting theta 1 to 6 in the forward kinematics transformation matrix:

Error in px,py,pz: -0.0990968335942661 0.0320861960505285 -0.252077473941376
Error in px,py,pz: -0.126829346779732 -0.239046730913604 0.00247607372870351
Error in px,py,pz: -0.0215235362331792 -0.0234540360889888 -0.0436301472772084

Error in px,py,pz: -0.0707976389930836 -0.189703355151841 -0.0323196528465646
Error in px,py,pz: -0.0679162532031579 0.140699752295997 -0.228007216438202
Error in px,py,pz: -0.00889473781877181 -0.0219972355508529 -0.0485175168318721
Error in px,py,pz: -0.0221364132592399 -0.0879660286849724 -0.0288576822941344
Error in px,py,pz: -0.00258642691226285 -0.0173508640047078 -0.0510803493174545
Error in px,py,pz: 0.000510623459902104 -0.0432716533009083 -0.0865339109675487
Error in px,py,pz: -0.301061877301010 0.0512154814259280 0.220219538832045
Error in px,py,pz: 0.00147778174815111 -0.00720088594713086 -0.0535061452074097
Error in px,py,pz: -0.00273709064940064 2.37433897911434e-7 -0.0539393409951956
Error in px,py,pz: -0.00372870785825263 -0.000191407574657250 -0.0230540670322896