Project: Kinematics Pick & Place

Rubric Points

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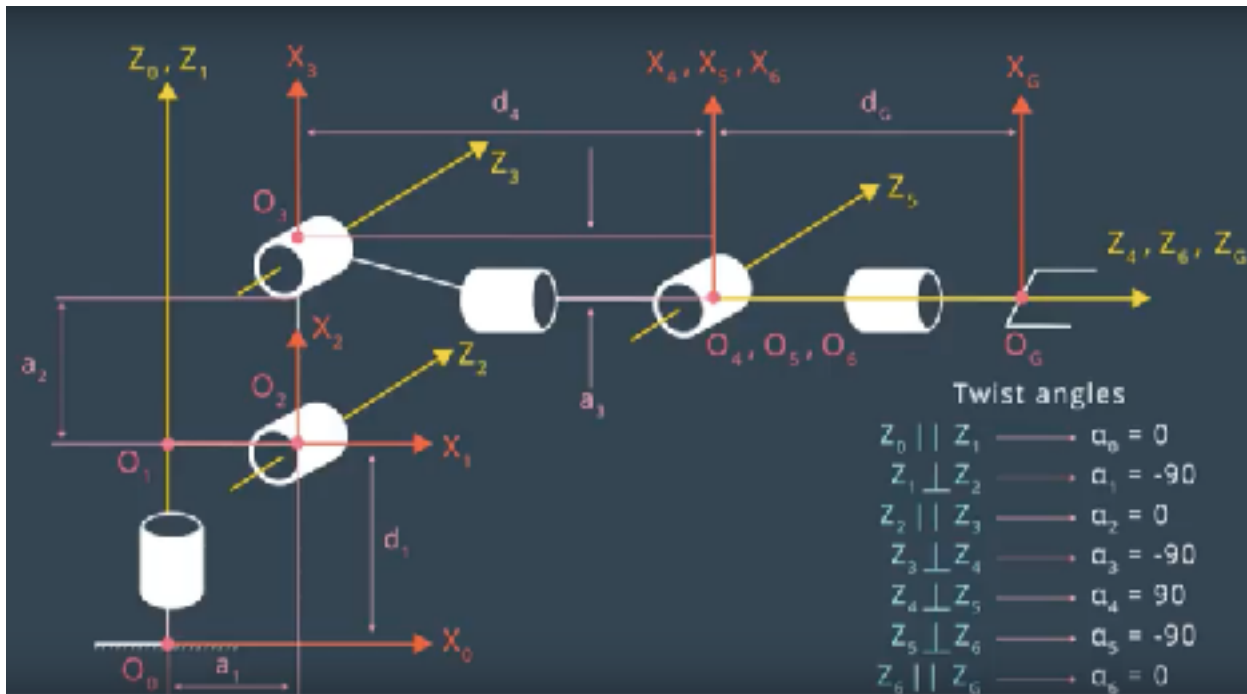
Writeup / README

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.
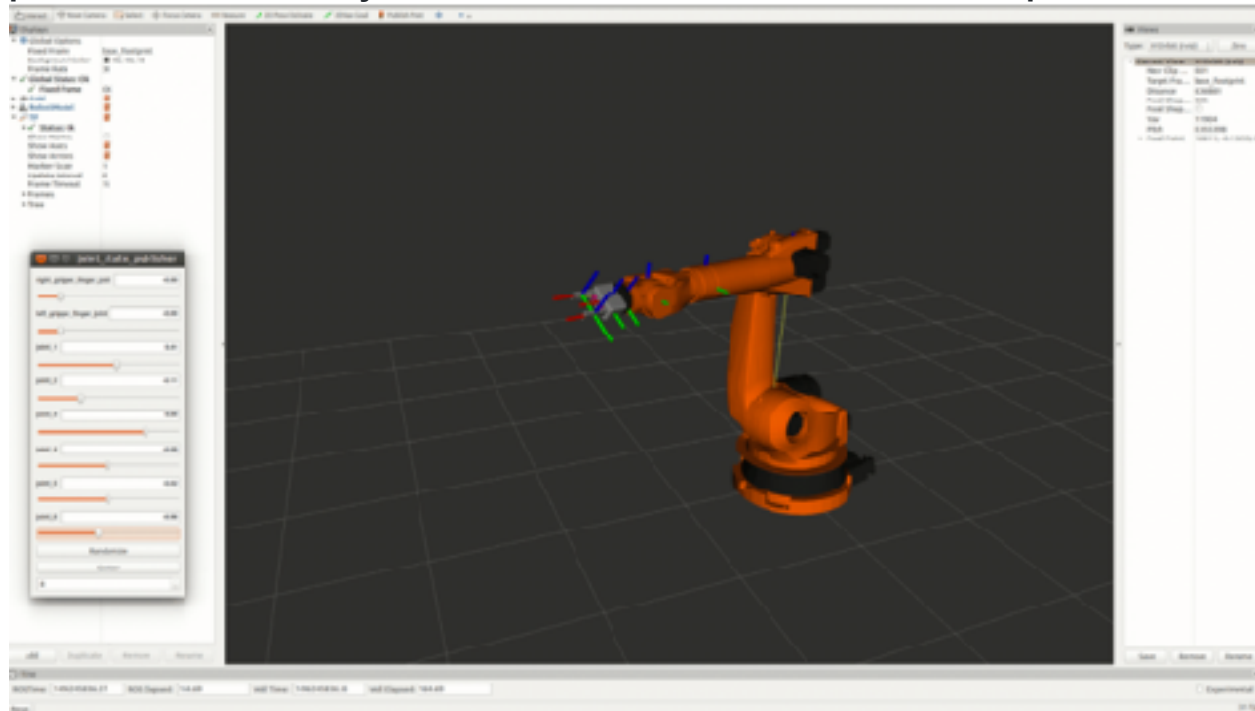
DH Parameters

Joint	alp(i-1)	a(i-1)	Di	Qi
1	0	0	0.75	Q1
2	-pi/2	0.35	0	Q2-pi/2
3	0	1.25	0	Q3
4	-pi/2	-0.054	1.5	Q4
5	pi/2	0	0	Q5
6	-pi/2	0	0	Q6
G	0	0	0.303	Q7

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 =  [            cos(q),            -sin(q),        0,            a]
      [sin(q)*cos(alp), cos(q)*cos(alp), -sin(alp), -sin(alp)*d]
      [sin(q)*sin(alp), cos(q)*sin(alp),  cos(alp),  cos(alp)*d]
      [              0,               0,        0,            1]


T0_1:         [cos(q1), -sin(q1), 0,    0]
              [sin(q1),  cos(q1), 0,    0]
              [      0,        0, 1, 0.75]
              [      0,        0, 0,    1]


T1_2:         [sin(q2),  cos(q2), 0, 0.35]
              [      0,        0, 1,    0]
              [cos(q2), -sin(q2), 0,    0]
              [      0,        0, 0,    1]


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]
```

Generalized homogenous vector using end-effector pose

$$T = \begin{bmatrix} & & & \vdots & P_x \\ & R_T & & \vdots & P_y \\ & & & \vdots & P_z \\ \hdashline 0 & 0 & 0 & \vdots & 1 \end{bmatrix}$$

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 nx, ny, and nz 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 Rrpy

```
Rrpy = Rot(Z, yaw) * Rot(Y, pitch) * Rot(X, roll) * R_corr
```
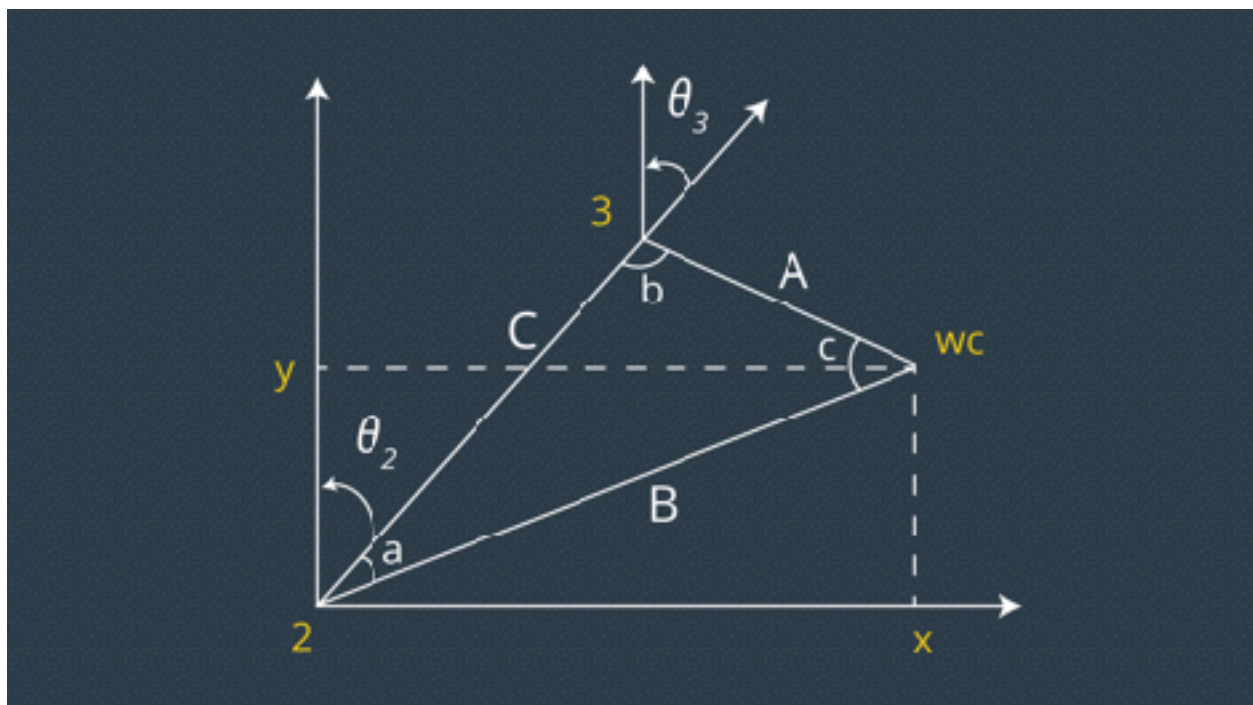
We can extract nx, ny, nz from Rrpy matrix

This will give us wx, wy, wz which represents wrist centre

Step2: Calculate first three angles from wrist centre

```
theta1 = atan2(wy, wx)
```

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

```
R0_6 = R0_1*R1_2*R2_3*R3_4*R4_5*R5_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:

```
R0_6 = Rrpy
```
where,
Rrpy = 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 **inv(R0_3)** which leads to:
```
R3_6 = inv(R0_3) * Rrpy
```

```
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. 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                 theta4 = atan2(R3_6[2,2], −R3_6[0,2])
142                 theta5 = atan2(sqrt(R3_6[0,2]*R3_6[0,2] +
R3_6[2,2]*R3_6[2,2]), R3_6[1,2])
143                 theta6 = atan2(−R3_6[1,1], R3_6[1,0])
144
```