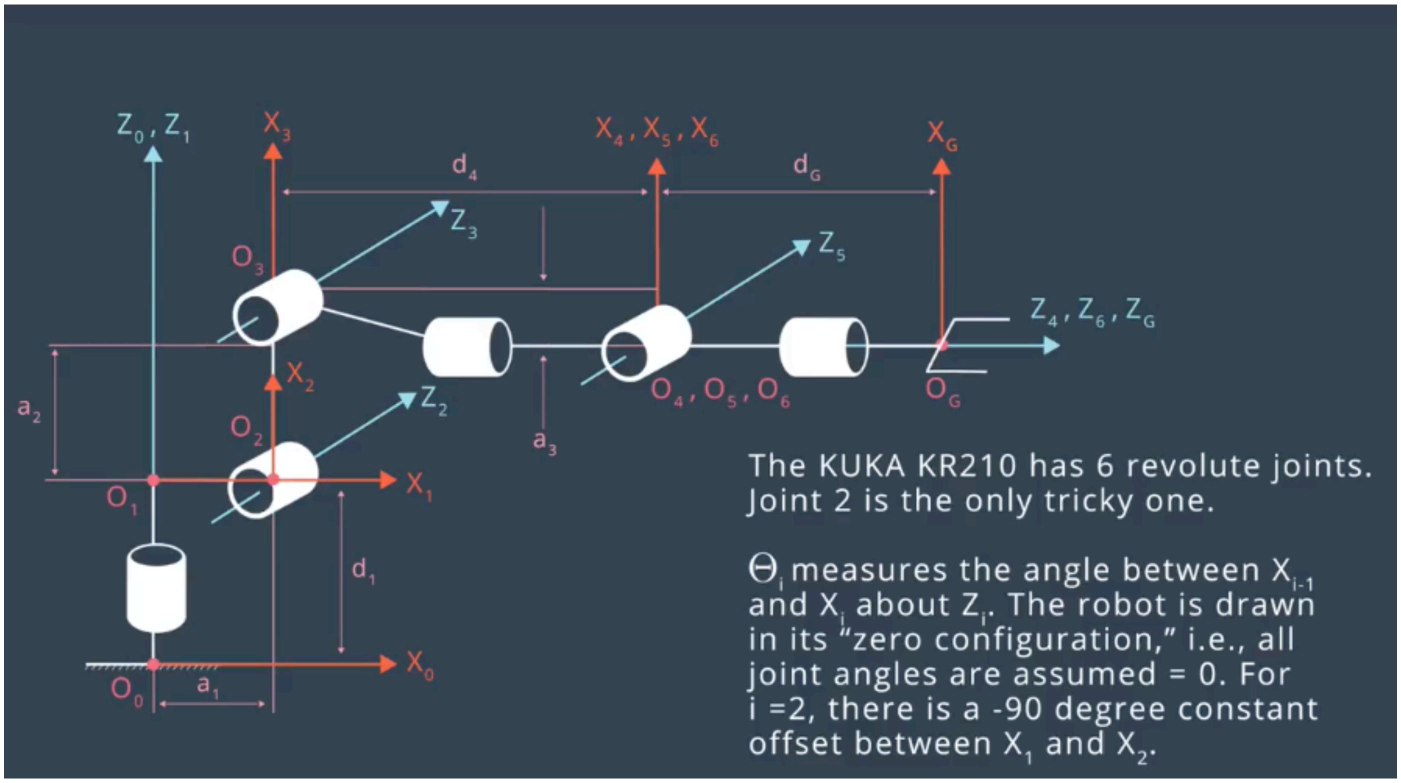
Project: Pick and Place

Writeup / README

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 joint coordinates and their translation relations between adjacent coordinates are defined according to the course materials and the best described via a the picture from it below.



The KUKA KR210 has 6 revolute joints. Joint 2 is the only tricky one.

Θ_i measures the angle between X_{i-1} and X_i about Z_i. The robot is drawn in its "zero configuration," i.e., all joint angles are assumed = 0. For i =2, there is a -90 degree constant offset between X_1 and X_2.

where:

1. alpha$_{i-1}$ - rotation offset about X_{i-1}.
2. a_{i-1} - translation offset between Z_{i-1} and Z_i.
3. d_i - translation offset between X_{i-1} and X_i.
4. theta$_i$ - rotation offset about Z_i axis.

The robot DH parameters are shown below:

Links	alpha(i-1)	a(i-1)	d(i)	theta(i)

0->1	0	0	0.75	q_1
1->2	- pi/2	0.35	0	-pi/2 + q_2
2->3	0	1.25	0	q_3
3->4	- pi/2	-0.054	1.50	q_4
4->5	pi/2	0	0	q_5
5->6	- pi/2	0	0	q_6
6->EE	0	0	0.303	0

These values were derived from the robot urdf.xaro file. During the calculations, it is required to remember that in the joints section, the origin location of each joint is described with respect to the previous joint. It is desired to combine offsets on two adjacent joints but along the same axis into one offset to reduce the none zero entity. For example, for joint 1 (z=0.33) and joint 2 (z=0.42), the offsets are combined and 0.75 placed on just joint 2

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 individual transformation matrix in each joint can be expressed as python function below, and the transformation matrix from base to gripper is just multiplication of all the individual matrics together.

```python
def TF_Matrix(alpha, a, d, q):
  TF = 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 TF

T0_1 = TF_Matrix(alpha0, a0, d1, q1).subs(s)
T1_2 = TF_Matrix(alpha1, a1, d2, q2).subs(s)
T2_3 = TF_Matrix(alpha2, a2, d3, q3).subs(s)
T3_4 = TF_Matrix(alpha3, a3, d4, q4).subs(s)
T4_5 = TF_Matrix(alpha4, a4, d5, q5).subs(s)
T5_6 = TF_Matrix(alpha5, a5, d6, q6).subs(s)
T6_EE = TF_Matrix(alpha6, a6, d7, q7).subs(s)

T0_EE = T0_1 * T1_2 * T2_3 * T3_4 * T4_5 * T5_6 * T6_EE
```

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

Thanks to the mechancial structure design, the inverse orientation kinematic problem can be achieved by the 3 joints at the wrist unit (J4 to J6) , and the inverse position kinematic problem with 6 joint variables for the desired end effector (EE) can be simplified as finding the wrist center (WC) position with 3 joint variables (J1 to J3). The following formula is used to compute the desired WC position based on the desired EE position and orientation.

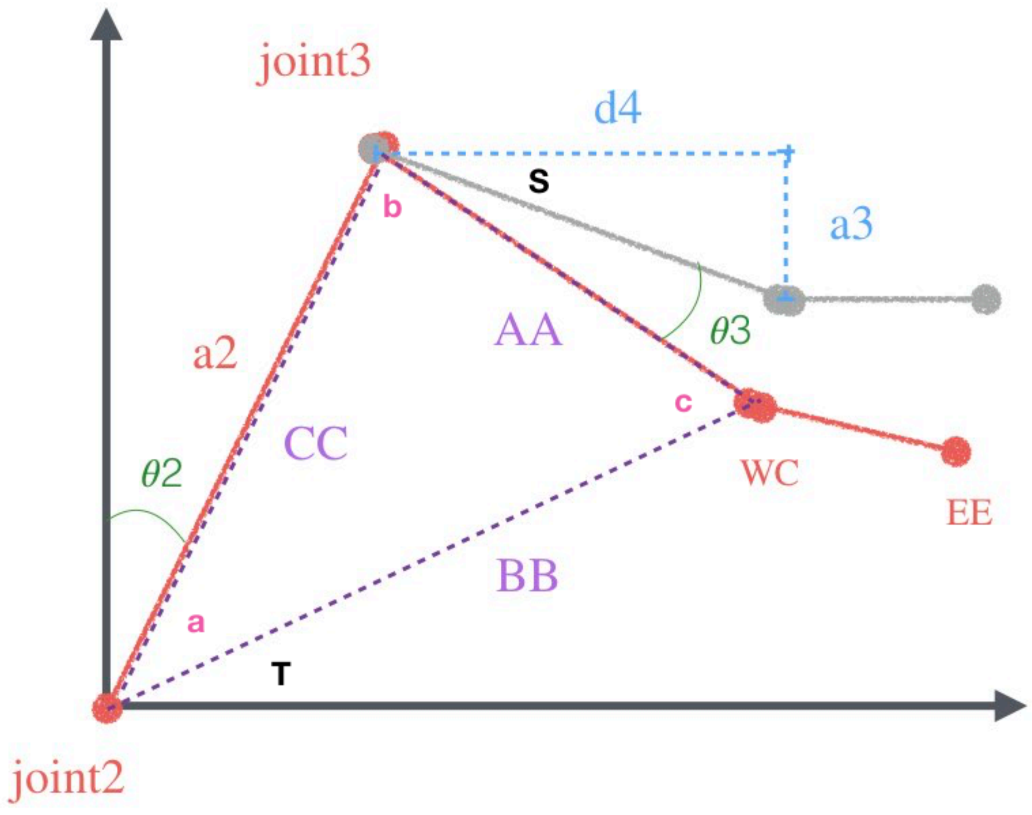
```
POS_WC = POS_EE - ROT_EE * (joint_6 length)
```

Inverse position

The inverse position question is to find J1, J2, and J3 values for the desired WC position. Joint 1 value can be found by X and Y values of WC.

```
theta1 = atan2(WC[1], WC[0]) # WC[0] = WC_x, WC[1] = WC_y, WC[2] = WC_z
```

For joint 2 and joint 3, the cos rule *cosC = (a² + b² - c²) / 2ab* is used. Refer to the picture below.



Note that angle between a_2 and d_4 is a right angle.

In the triangle AA_BB_CC, all the three length are known values and the three inner angles can be calcated. Then joint 2 and joint 3 can be found.

```
theta2 = pi/2 - angle_a - angle_T
theta3 = pi/2 - (angle_b + angle_S)
```

Inverser orientation

With J1 to J3, ROT0_3 and ROT3_6 can be calculated. Then J4 to J6 can be found in the matrix.

```
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.

The discussed solution was implemented in `IK_server.py` where it calculates Inverse Kinematics based on previously performed Kinematic Analysis. Robot successfully fulfills the project requirements and completes 9/10 pick&place cycles. In the pictures presented below, the robot was unable to do a pick in one of the attempts. There are 2 objects on the shelf as the pictures are taken when the system was ready to do the 11th attempt.