Project: Kinematics Pick & Place

Dong-Won Shin

Writeup / README

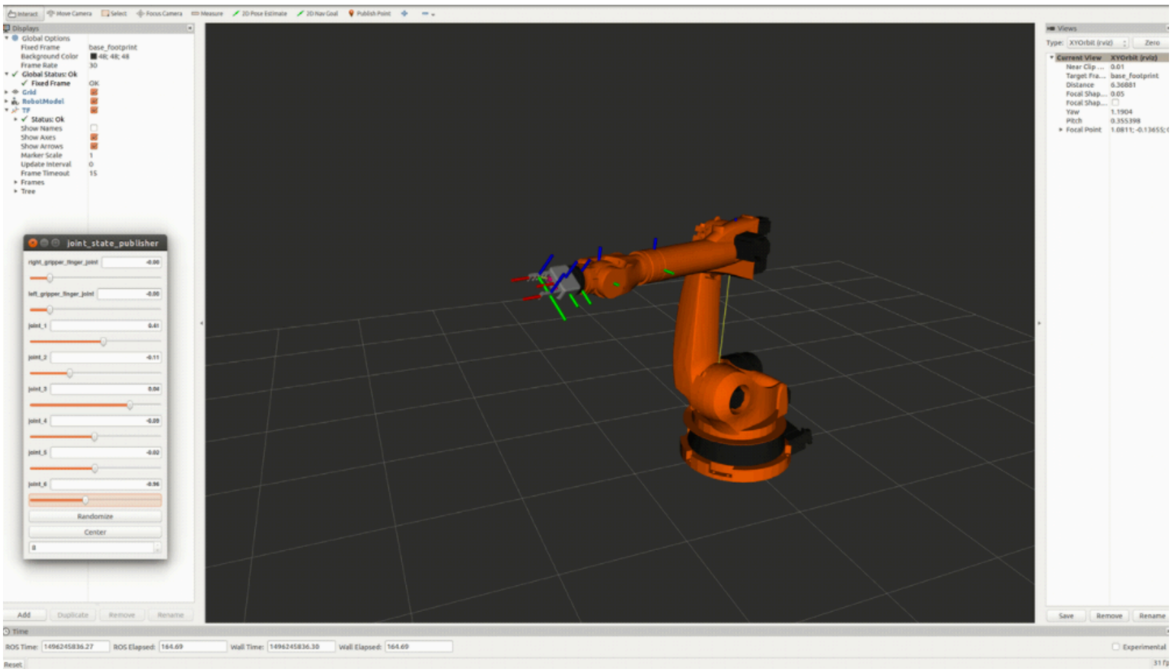
1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf.

You're reading it!

Kinematic Analysis

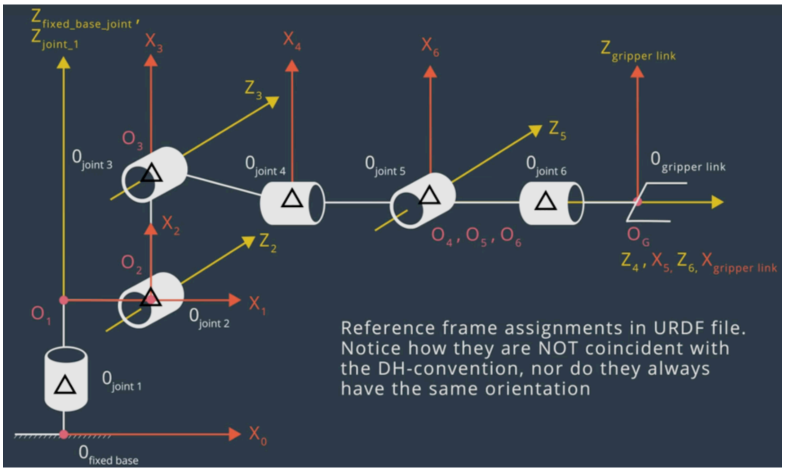
1. Run the forward_kinematics demo and evaluate the kr210.urdf.xacro file to perform kinematic analysis of Kuka KR210 robot and derive its DH parameters.

- Forward_kinematics demo



- Deriving DH parameters

The following figure shows the schematic diagram for kuka-kr210. We can derive the DH parameters by the provided instructions and evaluating the kr210.urdf.xacro file.



Links	alpha(i-1)	a(i-1)	d(i-1)	theta(i)
0->1	0	0	0.75	qi
1->2	-pi/2	0.35	0	-pi/2+q2
2->3	0	1.25	0	q3
3->4	-pi/2	-0.054	1.5	q4
4->5	pi/2	0	0	q5
5->6	-pi/2	0	0	q6
6->EE	0	0	0.303	0

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.

In order to create individual transformation matrices about each joint, I defined the Transformation_Matrix function. By substitute the symbols with the DH parameters, we can calculate the transformation matrices numerically. And by multiplying all of them in the order from origin to the end-effector, we can obtain the generalized homogeneous transform.

```
# Define Modified DH Transformation matrix
def Transformation_Matrix(alpha, a, d, q):
    MAT = 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 MAT

# Create individual transformation matrices
T0_1 = Transformation_Matrix(alpha0, a0, d1, q1).subs(s)
T1_2 = Transformation_Matrix(alpha1, a1, d2, q2).subs(s)
T2_3 = Transformation_Matrix(alpha2, a2, d3, q3).subs(s)
T3_4 = Transformation_Matrix(alpha3, a3, d4, q4).subs(s)
T4_5 = Transformation_Matrix(alpha4, a4, d5, q5).subs(s)
T5_6 = Transformation_Matrix(alpha5, a5, d6, q6).subs(s)
T6_G = Transformation_Matrix(alpha6, a6, d7, q7).subs(s)

# Transform from base_link to gripper
T0_G = T0_1 * T1_2 * T2_3 * T3_4 * T4_5 * T5_6 * T6_G
```

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

3.0 Getting the position and orientation of the end-effector

From the parameter(req) of the handle_calculate_IK function, we can calculate the position and orientation of the end-effector. (px, py, pz) are the x,y,z position of the end effector, respectively. In order to get the (roll, pitch, yaw) value from the parameter(req), I used the euler_from_quaternion function from tf package as instructed.

```
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])
```

3.1 inverse position kinematics

Since we have the case of a spherical wrist involving joints 4,5,6, the position of the wrist center is governed by the first three joints. We can obtain the position of the wrist center by using the complete transformation matrix. Let us symbolically define our homogeneous transform as following.

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

where l, m and n are orthonormal vectors representing the end-effector orientation along X,Y,Z axes of the local coordinate frame. Since n is the vector along the z-axis of the end-effector, we can write the following:

$$\begin{aligned} w_x &= p_x - (d_6 + l) \cdot n_x \\ w_y &= p_y - (d_6 + l) \cdot n_y \\ w_z &= p_z - (d_6 + l) \cdot n_z \end{aligned}$$

Where,

Px, Py, Pz = end-effector positions

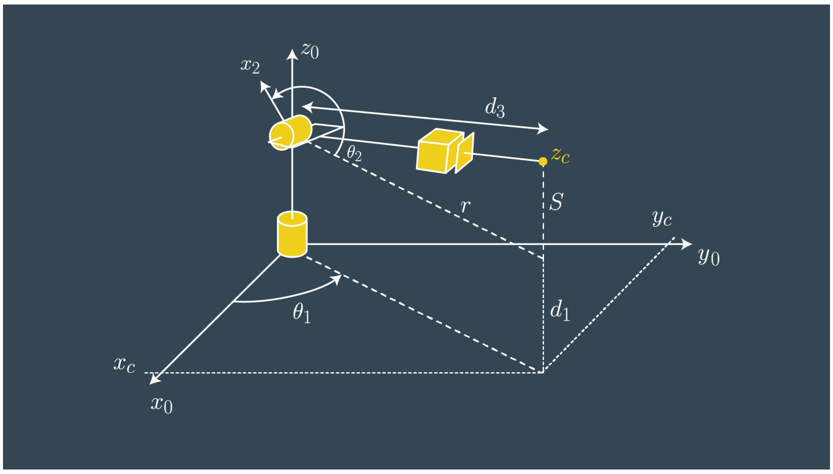
Wx, Wy, Wz = wrist positions

d6 = from DH table (=0)

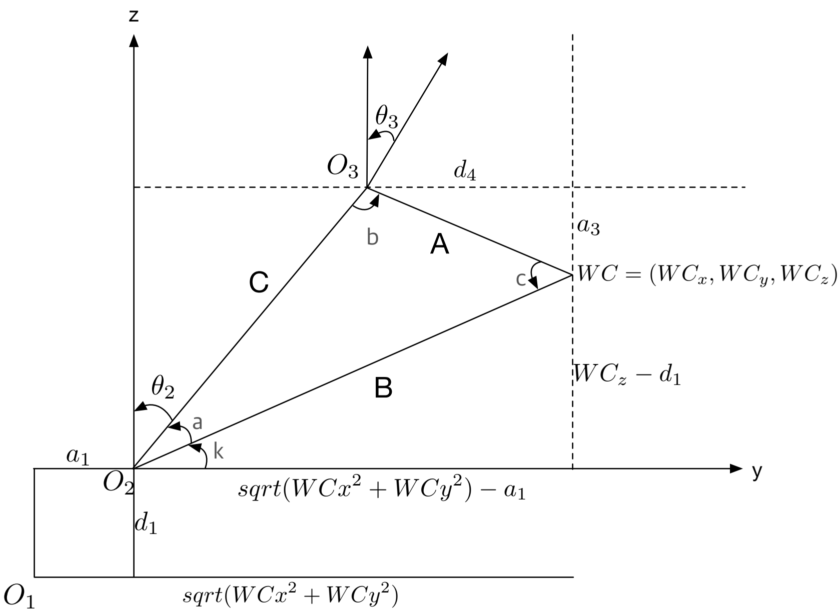
l = end-effector length (=0.303)

Now that we have the wrist center position, we can now calculate the theta 1,2,3.

Calculating theta_1 will be relatively easy. We just need to project the position of the wrist center onto the ground plane, then we just know the theta_1 is related to the x position and y position of the wrist center. We can get the theta_1 by using the atan2(WCy, WCx).



theta_2 and theta_3 are trickier to calculate.



$$A = sqrt({a_3}^2 + {d_4}^2)$$
$$B = sqrt((sqrt(WCx^2 + WCy^2) - a_1))^2 + (WCz - d_1)^2)$$
$$C = a_2 = 1.25 \text{ (from DH parameters)}$$

$$a = acos(\frac{B^2 + C^2 - A^2}{2BC})$$

$$b = acos(\frac{A^2 + C^2 - B^2}{2AC})$$

$$c = acos(\frac{A^2 + B^2 - C^2}{2AB})$$

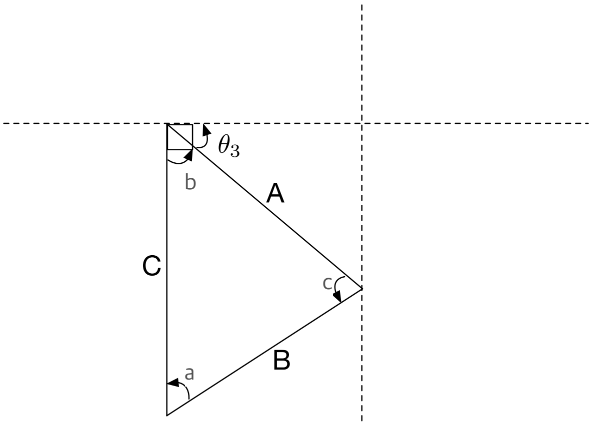
$$k = atan(WC_z - d_1, sqrt(WC_x^2 + WC_y^2) - a_1)$$

finally, theta_2 and theta_3 are

```
theta2 = pi/2 - a_1 - atan2(WCz - d1, sqrt(WCx**2 + WCy**2) - a1).subs(s)
theta3 = pi/2 - (b_1 + asin(0.054/A)) # using the angle generated by the sag in link4
```

in case of theta2, it is (theta2 + a + k) = pi/2. Therefore, it is obvious that theta2 = (pi/2 – a – k).

in case of theta3, we can consider the below illustration.



theta_3 can be worked out by solving the angle b and subtracting that from pi/2 and adjusting for the slight deviation in Z of -0.054 from our DH parameter table.

3.2 inverse orientation kinematics

For the inverse orientation problem, we need to find values of the final three joint variables. Using the individual DH transforms we can obtain the resultant transform and hence resultant rotation by

T0_G = T0_1 * T1_2 * T2_3 * T3_4 * T4_5 * T5_6 * T6_G

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 = R0_3.inv() * R_G

The calculation of the theta_4, theta_5, theta_6,
I used the formulation from project walkthrough to get the value for theta_4, theta_5, and theta_6.

```
# Reference to the project walkthrough was required for this section
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.

Here, I displayed several screenshots from the simulation from picking to the dropping items.