PROJECT SPECIFICATION

Robotic Arm: Pick & Place

Writeup

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
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. Hereis a template writeup for this project you can use as a guide and a starting point.	The writeup / 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.	This document addresses all the rubric points with supporting documentation. It's named Report_RoboND_Pick_and_Place.pdf and will be in the root folder of the repo.

Kinematic Analysis

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
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 writeup template). It is strongly recommended that you use pen and paper to create this figure to get a better understanding of the robot kinematics.	DH Parameters are derived from lesson and is put in dh_parameters.pdf in the repo.
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	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	The DH convention uses four individual transforms as per: $^{i-1}_{i}T = R(x_{i-1}, \alpha_{i-1})\, T(x_{i-1}, a_{i-1})\, R(z_i, \theta_i)\, T(z_i, d_i)$ to describe the relative translation and orientation of link (i-1) to link(i). In matrix form

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
end-effector(gripper) pose.	created these matrices.	this transform is:

$$
{}^{i-1}_{\ i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

The code block for calculating the TF_Matrix is:

```
#
# Define Modified DH Transformation matrix
#
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


#
```

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS

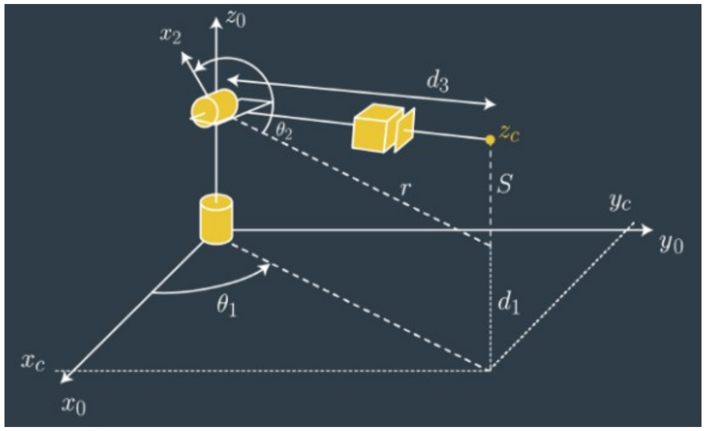
		# Create individual transformation matrices
		
		#
		
		T0_1 = TF_Matrix(alpha0, a0, d1, q1).subs(DH_Table)
		
		T1_2 = TF_Matrix(alpha1, a1, d2, q2).subs(DH_Table)
		
		T2_3 = TF_Matrix(alpha2, a2, d3, q3).subs(DH_Table)
		
		T3_4 = TF_Matrix(alpha3, a3, d4, q4).subs(DH_Table)
		
		T4_5 = TF_Matrix(alpha4, a4, d5, q5).subs(DH_Table)
		
		T5_6 = TF_Matrix(alpha5, a5, d6, q6).subs(DH_Table)
		
		T6_EE = TF_Matrix(alpha6, a6, d7, q7).subs(DH_Table)
		
		#
		
		# Extract rotation matrices from the transformation matrices
		
		#
		
		T0_EE = T0_1 * T1_2 * T2_3 * T3_4 * T4_5 * T5_6 * T6_EE

Decouple Inverse Kinematics problem into Inverse Position Kinematics and inverse Orientation Kinematics; doing	Based on the geometric Inverse Kinematics method described here, breakdown the IK problem into Position and Orientation problems. Derive the	First the location of the spherical wrist center is calculated in DH model using:

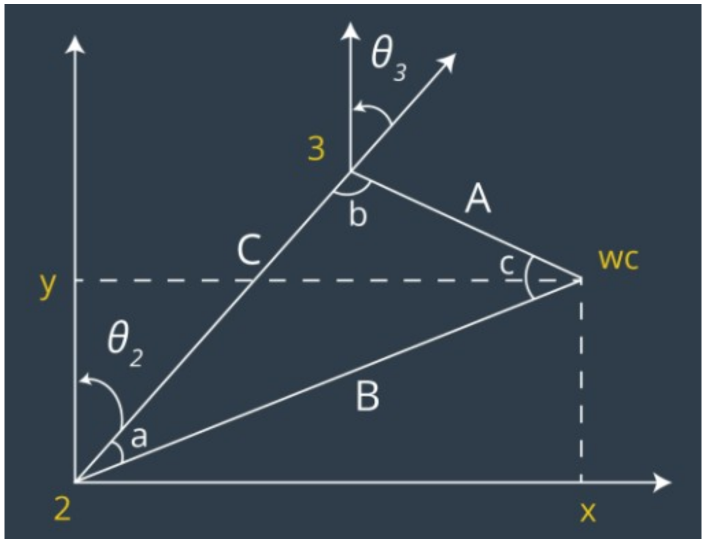
CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
so derive the equations to calculate all individual joint angles.	equations for individual joint angles. Your writeup must contain details about the steps you took to arrive at those equations. Add figures where necessary. If any given joint has multiple 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).	$^0r_{WC/0_0} = {}^0r_{0_G/0_0} - r_{0_5/0_G}$ $\begin{bmatrix} WC_x \\ WC_y \\ WC_z \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} - {}^0_6R \begin{bmatrix} 0 \\ 0 \\ d_G \end{bmatrix}$ The code to do this is:

```
r, p, y = symbols('r p y')
ROT_x = rot_x(r)
ROT_y = rot_y(p)
ROT_z = rot_z(y)

ROT_EE = simplify(ROT_z * ROT_y * ROT_x)
Rot_Error = ROT_z.subs(y, pi) * ROT_y.subs(p, -pi/2)
ROT_EE = simplify(ROT_EE * Rot_Error)
```

```
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])

### Your IK code here
# Compensate for rotation discrepancy between DH parameters and Gazebo
#


ROT_EE = ROT_EE.subs({'r': roll, 'p': pitch, 'y': yaw})

EE = Matrix([[px],
   [py],
   [pz]])

WC = EE - 0.303 * ROT_EE[:,2]
```

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
		Once the wrist center is known, we can calculate theta1:  z_0 x_2 d_3 z_c θ_2 r S y_c y_0 d_1 θ_1 x_c x_0 the code for this is: theta1 = atan2(WC[1], WC[0]) Using the figure below we can calculate theta2 and theta3

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS



The code for this is:

```
# SSS triangle for theta2, theta3
side_a = 1.501
side_b = sqrt(pow((sqrt(WC[0] * WC[0] + WC[1] * WC[1]) - 0.35), 2) + pow((WC[2] - 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(WC[2] - 0.75, sqrt(WC[0] * WC[0] + WC[1] * WC[1]) - 0.35)
theta3 = pi/2 - (angle_b + 0.036) # 0.036 accounts for sag in link4 of 0.054m
```

Using the formula, we can calculate the remaining three.

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
		${}^3_6R = \left({}^0_3R\right)^{-1}\,{}^0_6R$ The code for this is:

```
R0_3 = T0_1[0:3, 0:3] * T1_2[0:3, 0:3] * T2_3[0:3, 0:3]
R0_3 = R0_3.evalf(subs={q1: theta1, q2: theta2, q3: theta3})

R3_6 = R0_3.inv("LU") * ROT_EE

# Euler angles from rotation 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

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
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. A screenshot of the completed pick and place process is included.	`IK_server.py` must contain properly commented code. The robot must track the planned trajectory and successfully complete pick and place operation. Your writeup must include explanation for the code and a discussion on the results, and a screenshot of the completed pick and place process.	The project is discussed in the above table. The robotic arm picking and placing the objects as expected. I put a few videos in the videos folder. One of the videos(trajectory.mp4) show the trajectory. The robotic arm is able to meet the expected 8/10 pick and place cycles. The observation I made is, sometimes, the gazebo server is crashing or throwing the error “Failed to call service calculate_ik” at which stage we need to restart the environment. Then again it is successful. Sometimes, it gets annoying as it keep happens.