Project: Kinematics Pick & Place

Brett Gleason

July 31, 2017

The rubric for this project can be found at the following URL:
https://review.udacity.com/#!/rubrics/972/view
I will consider the rubric points individually and describe how I addressed each point in my implementation.

Project Scope

The scope of this project is to calculate the inverse kinematics for the KUKA KR210 robotic arm, allowing the robot to perform a pick and place operation. Gazebo is used to simulate the robot and RViz is used for path planning. A passing project submission must successfully place the cylinder into the bucket at least 8/10 times.

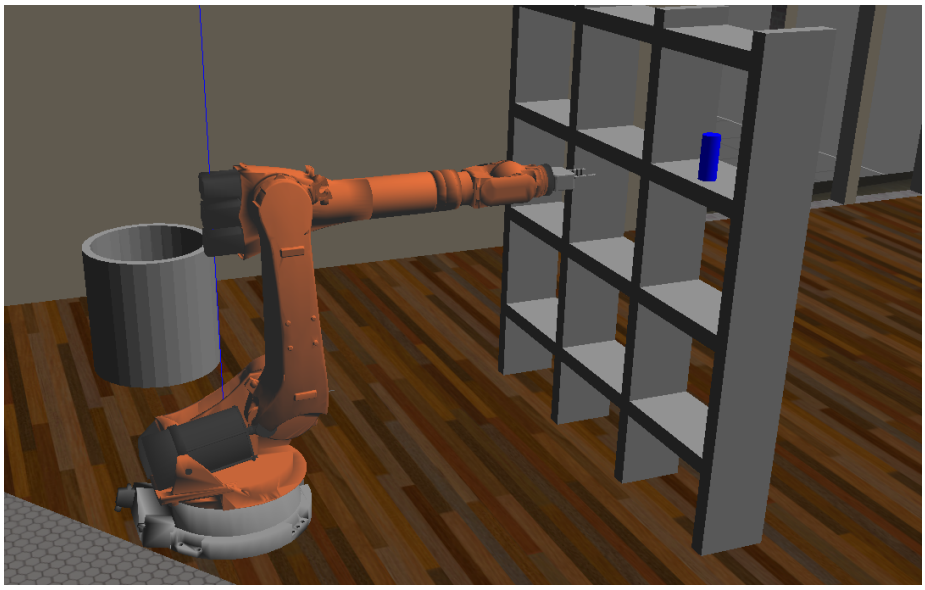

Figure 1: KUKA KR210 robotic arm in the simulated environment

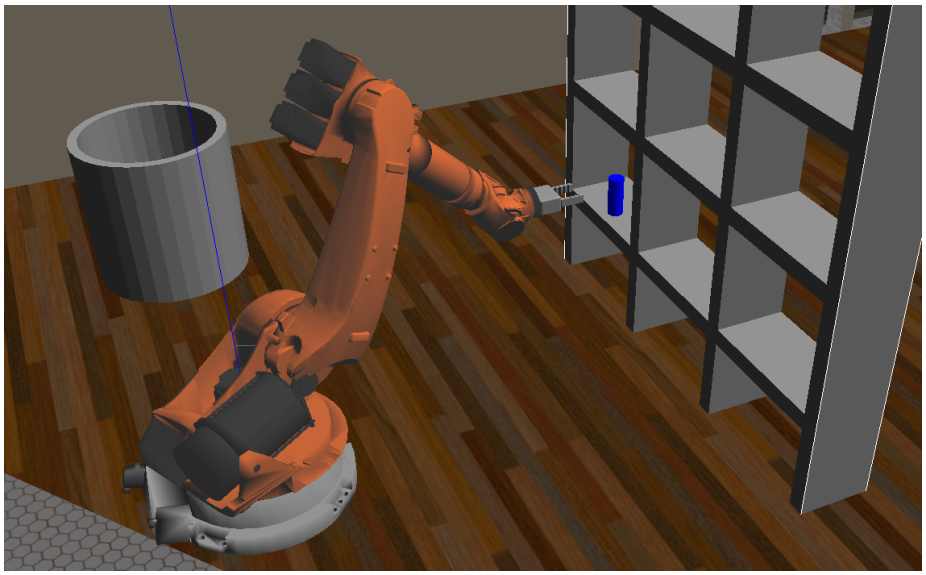

Figure 2: Picking up an object

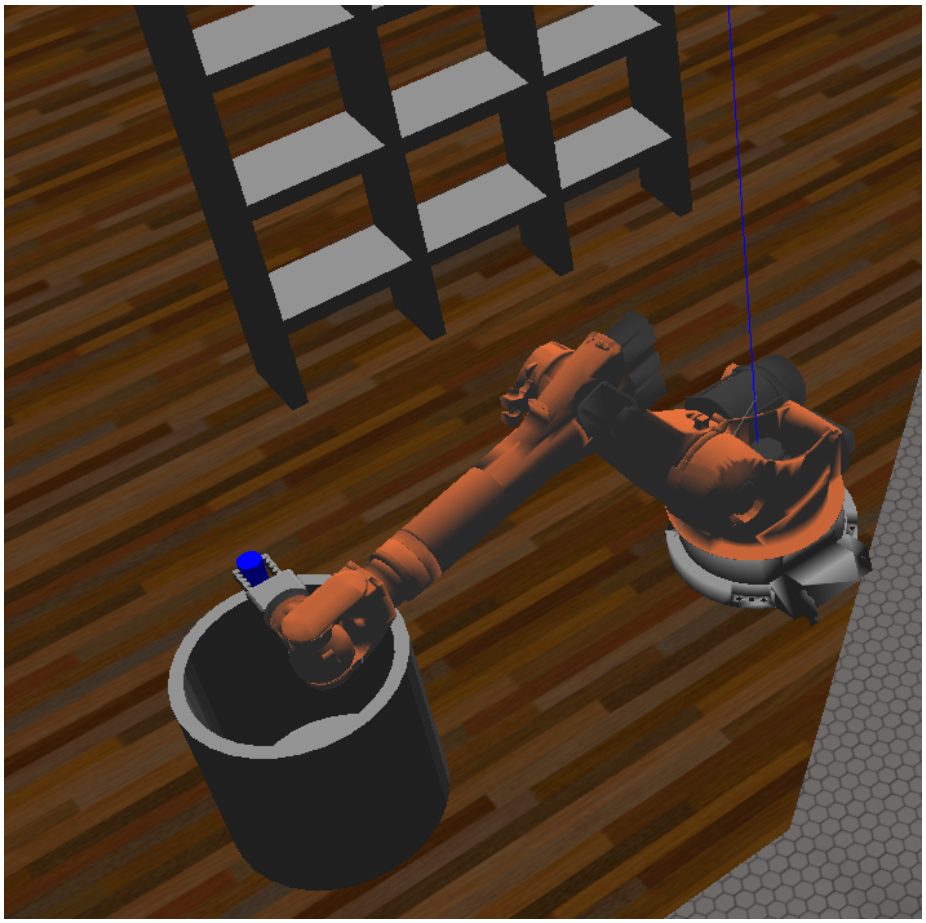

Figure 3: Placing the object into the bucket

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.

Using the model of the Kuka KR210 robotic arm in the forward kinematics demo as well as the description of the joints within the URDF file, a schematic diagram of the robot can be drawn.

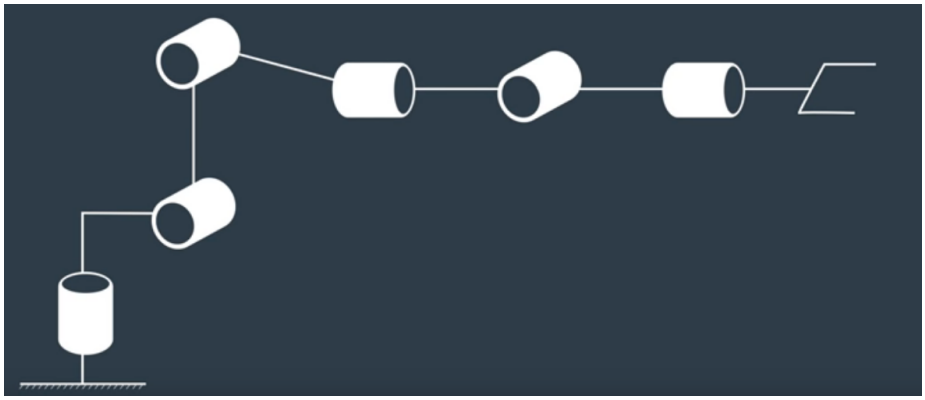

Figure 4: Basic schematic as shown in project lesson 10

Next the joints are labeled from 1 to n and the links are labeled from 0 to n.

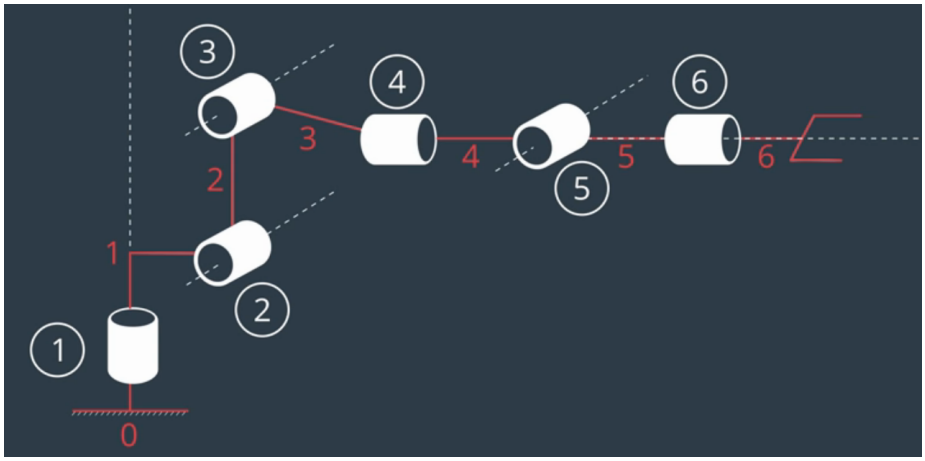

Figure 5: Schematic showing joint and link numbers as shown in project lesson 10

After the joints and links are labeled, reference frames can be defined for each joint.

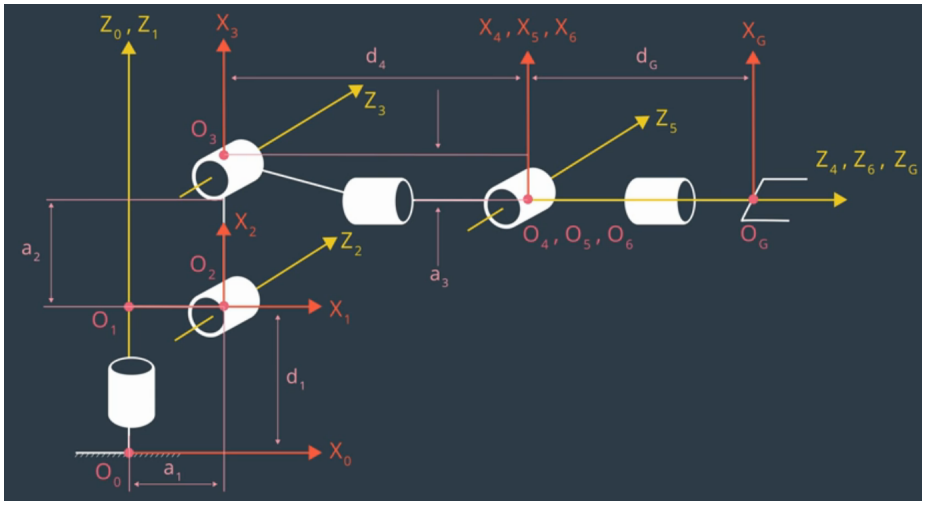

Figure 6: Schematic showing reference frames for each joint as shown in project lesson 10

Using the reference frames the Denavit-Hartenberg parameters can be defined. For this project the DH parameters are defined using the convention described by John J. Craig in his book Introduction to Robotics: Mechanics and Control. The definitions are as follows (from lesson 2 section 12):

- Twist angle (α_{i-1}): angle between $\hat{Z}_{i-1}$ and $\hat{Z}_i$ measured about $\hat{X}_{i-1}$ in a right hand sense.
- Link length (a_{i-1}): distance from $\hat{Z}_{i-1}$ to $\hat{Z}_i$ measured along $\hat{X}_{i-1}$.
- Link offset (d_i): signed distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured along $\hat{Z}_i$.
- Joint angle: angle between $\hat{X}_{i-1}$ and $\hat{X}_i$ measured about $\hat{Z}_i$ in a right hand sense.

i	Transform	α_{i-1}	a_{i-1}	d_i	θ_i
1	T_1^0	0	0	0.75	θ_1
2	T_2^1	$-\frac{\pi}{2}$	0.35	0	$\theta_2 - \frac{\pi}{2}$
3	T_3^2	0	1.25	0	θ_3
4	T_4^3	$-\frac{\pi}{2}$	-0.054	1.5	θ_4
5	T_5^4	$\frac{\pi}{2}$	0	0	θ_5
6	T_6^5	$-\frac{\pi}{2}$	0	0	θ_6
7	T_G^6	0	0	0.303	0

Table 1: Denavit-Hartenberg parameter table with values derived from the URDF file

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 general form of a homogeneous transform between two reference frames described using our convention can be written as follows:

$$T_{i-1}^{i} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

From this equation the individual transform matrices can be derived.

$$T_1^0 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & a_0 \\ \sin(\theta_1)\cos(\alpha_0) & \cos(\theta_1)\cos(\alpha_0) & -\sin(\alpha_0) & -\sin(\alpha_0)d_1 \\ \sin(\theta_1)\sin(\alpha_0) & \cos(\theta_1)\sin(\alpha_0) & \cos(\alpha_0) & \cos(\alpha_0)d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^1 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & a_1 \\ \sin(\theta_2)\cos(\alpha_1) & \cos(\theta_2)\cos(\alpha_1) & -\sin(\alpha_1) & -\sin(\alpha_1)d_2 \\ \sin(\theta_2)\sin(\alpha_1) & \cos(\theta_2)\sin(\alpha_1) & \cos(\alpha_1) & \cos(\alpha_1)d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & a_2 \\ \sin(\theta_3)\cos(\alpha_2) & \cos(\theta_3)\cos(\alpha_2) & -\sin(\alpha_2) & -\sin(\alpha_2)d_3 \\ \sin(\theta_3)\sin(\alpha_2) & \cos(\theta_3)\sin(\alpha_2) & \cos(\alpha_2) & \cos(\alpha_2)d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^3 = \begin{bmatrix} \cos(\theta_4) & -\sin(\theta_4) & 0 & a_3 \\ \sin(\theta_4)\cos(\alpha_3) & \cos(\theta_4)\cos(\alpha_3) & -\sin(\alpha_3) & -\sin(\alpha_3)d_4 \\ \sin(\theta_4)\sin(\alpha_3) & \cos(\theta_4)\sin(\alpha_3) & \cos(\alpha_3) & \cos(\alpha_3)d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5^4 = \begin{bmatrix} \cos(\theta_5) & -\sin(\theta_5) & 0 & a_4 \\ \sin(\theta_5)\cos(\alpha_4) & \cos(\theta_5)\cos(\alpha_4) & -\sin(\alpha_4) & -\sin(\alpha_4)d_5 \\ \sin(\theta_5)\sin(\alpha_4) & \cos(\theta_5)\sin(\alpha_4) & \cos(\alpha_4) & \cos(\alpha_4)d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_6^5 = \begin{bmatrix} \cos(\theta_6) & -\sin(\theta_6) & 0 & a_5 \\ \sin(\theta_6)\cos(\alpha_5) & \cos(\theta_6)\cos(\alpha_5) & -\sin(\alpha_5) & -\sin(\alpha_5)d_6 \\ \sin(\theta_6)\sin(\alpha_5) & \cos(\theta_6)\sin(\alpha_5) & \cos(\alpha_5) & \cos(\alpha_5)d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_G^6 = T_7^6 = \begin{bmatrix} \cos(\theta_7) & -\sin(\theta_7) & 0 & a_6 \\ \sin(\theta_7)\cos(\alpha_6) & \cos(\theta_7)\cos(\alpha_6) & -\sin(\alpha_6) & -\sin(\alpha_6)d_7 \\ \sin(\theta_7)\sin(\alpha_6) & \cos(\theta_7)\sin(\alpha_6) & \cos(\alpha_6) & \cos(\alpha_6)d_7 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The total homogeneous transform between the base link and the end link can be found by multiplying the individual transforms together

$$T_G^0 = T_1^0 * T_2^1 * T_3^2 * T_4^3 * T_5^4 * T_6^5 * T_G^6$$

Because of the difference in the way the orientation of the DH frames are defined versus the way the URDF file is written, two additional rotations must be applied to the gripper frame: a 180 degree rotation about the z-axis, followed by a -90 degree rotation about the y-axis.

$$R_z = \begin{bmatrix} \cos(\pi) & -\sin(\pi) & 0 & 0 \\ \sin(\pi) & \cos(\pi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y = \begin{bmatrix} \cos(-\frac{\pi}{2}) & 0 & \sin(-\frac{\pi}{2}) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-\frac{\pi}{2}) & 0 & \cos(-\frac{\pi}{2}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The total transform can then be described as:

$$T_{total} = T_G^0 * R_z * R_y$$

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

Because the axes of joints 4, 5, and 6 intersect at a single point, the position and orientation of the end effector are kinematically decoupled. Joints 4, 5, and 6 form a spherical wrist that determine the orientation of the end effector, while joints 1, 2 and 3 determine the location of the end effector.

Inverse Position

Wrist Center Position The first step for finding the first three joint angles is to use the end effector position that is passed to the inverse kinematics server to find the location of the wrist center. Because the wrist center is located at joint five, it can be found by starting at the end effector location and subtracting the link length from joint five to joint 6 and from joint 6 to the end effector along the Z_G axis as shown in figure 6. This can be written mathematically as follows:

$$\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} - \begin{bmatrix} d_6 + d_7 \\ d_6 + d_7 \\ d_6 + d_7 \end{bmatrix} * \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}$$

Where:

- w = wrist center location
- p = end effector location
- n = the vector along the Z_G axis

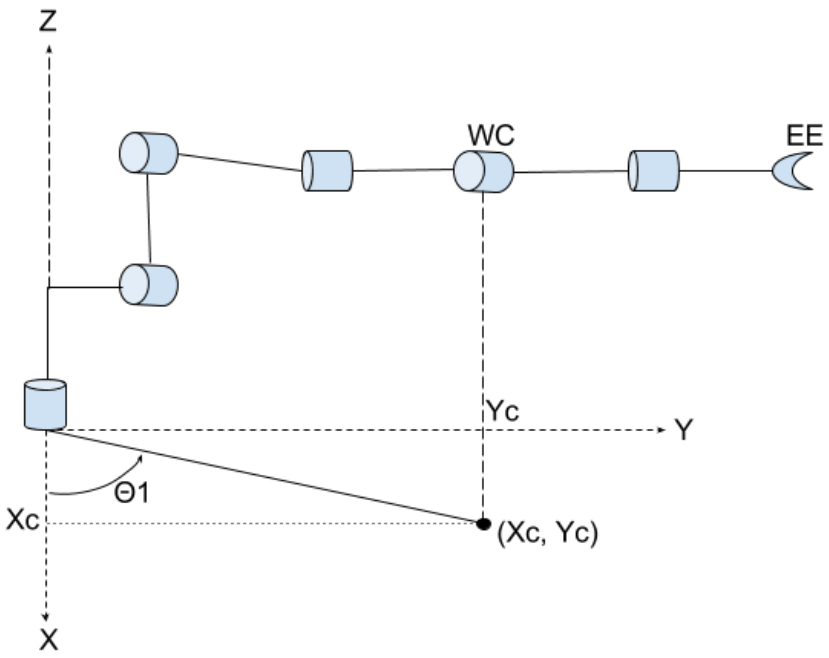

Figure 7: Diagram for calculating theta 1.

Joint 1 - θ_1

- EE = end effector
- WC = wrist center
- X_c = X component of wrist center position
- Y_c = Y component of wrist center position

Once the wrist center location is known, θ_1 can be found by using the projection of the wrist center onto the XY plane. θ_1 is the arctangent of the X and Y components of the wrist center position.

$$\theta_1 = atan2(Y_c, X_c)$$

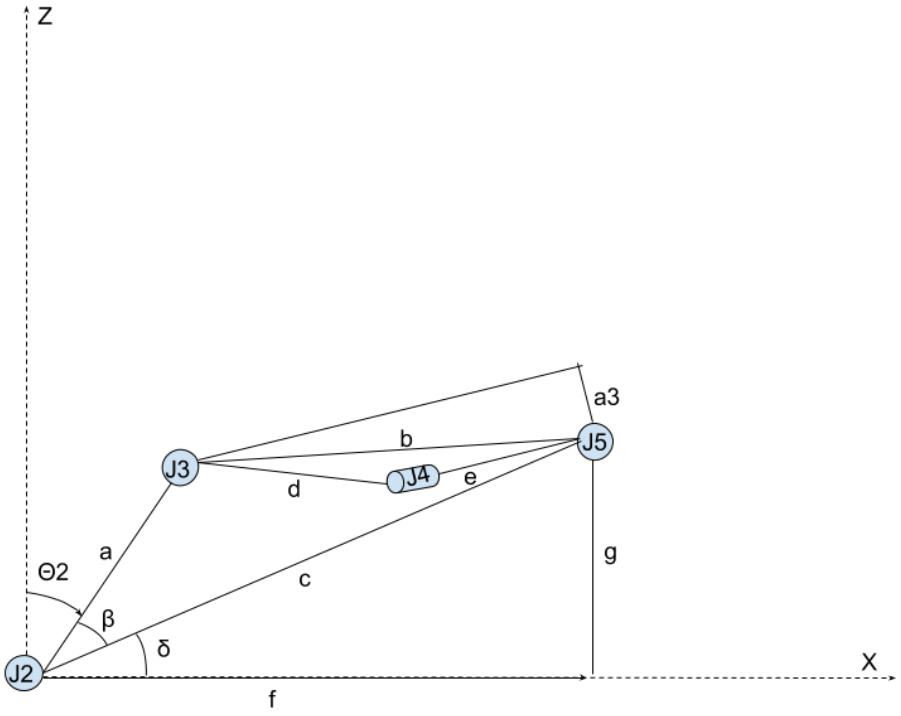

Figure 8: Diagram for calculating theta 2.

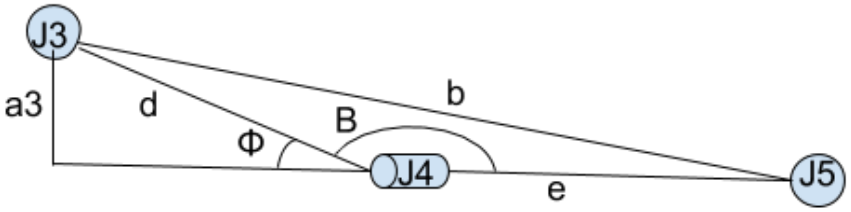

Figure 9: Diagram for calculating the length of b.

Joint 2 - θ_2

- a = link length from joint 2 to joint 3
- b = straight line distance between joint 3 and joint 5
- c = straight line distance between joint 2 and joint 5
- d = link length from joint 3 to joint 4
- e = link length from joint 4 to joint 5
- f = the projection of c onto the XY plane
- g = the distance between the Z position of joint 5 and the Z position of joint 2
- $a3$ = the vertical offset between joint 3 and joint 4
- β = the angle between a and c
- δ = the angle between c and f
- B = the angle between d and e
- ϕ = the offset angle caused by a3

a, d, e, and $a3$ are all from the DH parameter table.
Using figure 9, b can be calculated as follows:

$$\phi = asin\left(\frac{a3}{d}\right)$$

$$B = \pi - \phi$$

$$b = \sqrt{d^2 + e^2 - 2de\cos(B)}$$

Since the position of J5 is known after finding the wrist center and the position of J2 can be calculated using the DH parameter table and rotating by θ_1, c can be found by taking the distance between the two points:

$$c = \sqrt{(J5_x - J2_x)^2 + (J5_y - J2_y)^2 + (J5_z - J2_z)^2}$$

f can be found by removing the Z term from the c equation:

$$f = \sqrt{(J5_x - J2_x)^2 + (J5_y - J2_y)^2}$$

g is simply the distance between two points on the Z axis:

$$g = J5_z - J2_z$$

β can be found using the law of cosines because the lengths of all three sides of the triangle abc are known:

$$\beta = acos\left(\frac{b^2 - a^2 - c^2}{-2ac}\right)$$

since g and f are known, δ can be found using the arctangent:

$$\delta = atan2(g, f)$$

because the sum of θ_2, β and δ forms a right angle, θ_2 can be written as:

$$\theta_2 = \frac{\pi}{2} - \beta - \delta$$

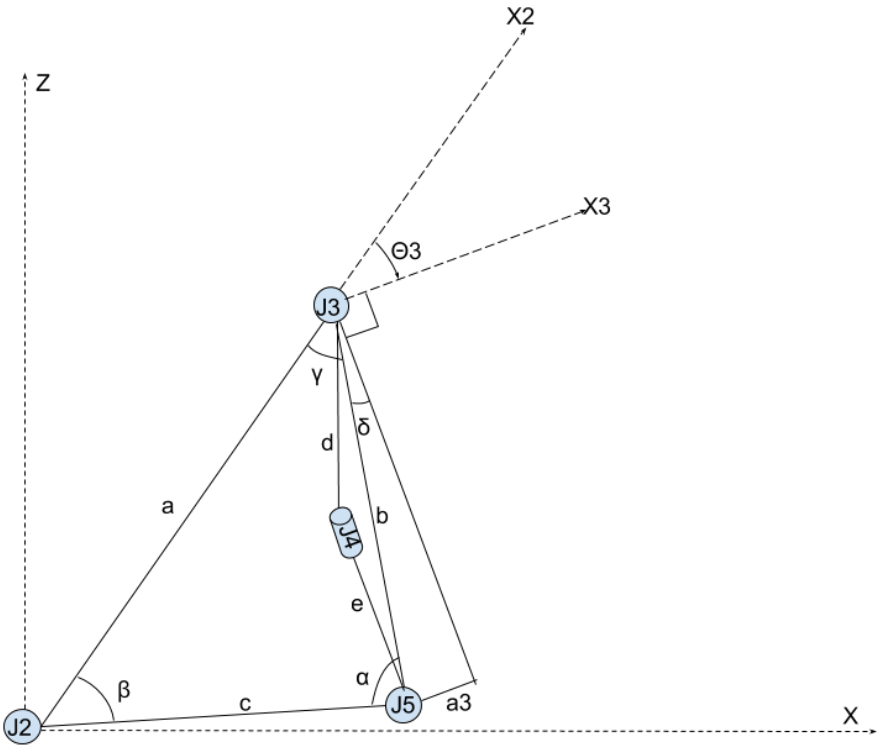

Figure 10: Diagram for calculating theta 3.

Joint 3 - θ_3

- a = link length from joint 2 to joint 3
- b = straight line distance between joint 3 and joint 5
- c = straight line distance between joint 2 and joint 5
- d = link length from joint 3 to joint 4
- e = link length from joint 4 to joint 5
- $a3$ = the vertical offset between joint 3 and joint 4
- α = the angle between b and c
- β = the angle between a and c
- γ = the angle between a and b
- δ = the angle between b and the Y axis of joint 3

a, d, e, and $a3$ are all from the DH parameter table.
b and c are calculated the same way as in the θ_2 calculation.
Because all side lengths of the triangle abc are known, γ can be found using the law of cosines:

$$\gamma = acos\left(\frac{c^2 - a^2 - b^2}{-2ab}\right)$$

$$\delta = asin\left(\frac{a3}{b}\right)$$

$$\theta_3 = \frac{\pi}{2} - \delta - \gamma$$

Inverse Orientation

Once the inverse position is solved the inverse orientation can be solved. The overall orientation is known based on the roll, pitch, and yaw Euler angles that are input to the inverse kinematics server. Since the first three joint angles are known, the last three joint angles must account for the difference in orientation between the first three joints and the total orientation. The total orientaion is found like so:

$$R_{total} = rot_z(yaw) * rot_y(pitch) * rot_x(roll)$$

Where rot_z, y, and x are the rotation matrices about the listed axis.

$$R_{total} = \begin{bmatrix} \cos(yaw) & -\sin(yaw) & 0 \\ \sin(yaw) & \cos(yaw) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos(pitch) & 0 & sin(pitch) \\ 0 & 1 & 0 \\ -\sin(pitch) & 0 & cos(pitch) \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(roll) & -\sin(roll) \\ 0 & \sin(roll) & \cos(roll) \end{bmatrix}$$

The orientation after the first three joints can be found by substituting the joint angles into the rotation portion of the Denavit-Hartenberg transforms for the first three frames:

$$R_3^0 = R_1^0(\theta_1) * R_2^1(\theta_2) * R_3^2(\theta_3)$$

$$R_3^0 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1)\cos(\alpha_0) & \cos(\theta_1)\cos(\alpha_0) & -\sin(\alpha_0) \\ \sin(\theta_1)\sin(\alpha_0) & \cos(\theta_1)\sin(\alpha_0) & \cos(\alpha_0) \end{bmatrix} * \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 \\ \sin(\theta_2)\cos(\alpha_1) & \cos(\theta_2)\cos(\alpha_1) & -\sin(\alpha_1) \\ \sin(\theta_2)\sin(\alpha_1) & \cos(\theta_2)\sin(\alpha_1) & \cos(\alpha_1) \end{bmatrix} * \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 \\ \sin(\theta_3)\cos(\alpha_2) & \cos(\theta_3)\cos(\alpha_2) & -\sin(\alpha_2) \\ \sin(\theta_3)\sin(\alpha_2) & \cos(\theta_3)\sin(\alpha_2) & \cos(\alpha_2) \end{bmatrix}$$

Because of the difference in orientation between the DH and URDF frames, a correction must be applied to R_3^0 similarly to the correction applied when calculating the total transform. The orientation of the last three joints can be found by taking the difference between the total orientation and the orientation of the first three joints:

$$R_6^3 = (R_3^0)^{-1} * R_{total}$$

Joints 4, 5, & 6 - $\theta_4, \theta_5, \theta_6$ Once the necessary orientation of the last three joints is known, θ_4, θ_5, and θ_6 can be found by transforming the orientation into Euler angles. These can be calculated by hand but for this project the euler_from_matrix() function from the TF package was used to find the Euler angles.

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'll talk about the code, what techniques I used, what worked and why, where the implementation might fail and how I might improve it if I were going to pursue this project further. All lines of code referenced refer to the IK_server.py file.

The majority of the transformation and rotation math was done symbolicly using the sympy package. Because the symbolic math takes a long time to calculate I moved as much of it as possible outside of the for loop that calculates the inverse kinematics. The Denavit-Hartenberg parameters are defined in lines 53-57. Using the URDF file and the reference frame schematic, the modified DH parameters are defined in lines 62-69. Transformation matrices between the DH reference frames are defined in lines 74-122 and the orientation difference between URDF and DH frames is corrected in lines 124-138. The total rotation

matrix is found on line 179. The position of the wrist center is calculated on lines 181-195. θ_1, θ_2, and θ_3 are calculated as described in the inverse position section on lines 195-227, many of the lengths used for these calculations are defined and calculated on lines 158-176. θ_4, θ_5, and θ_6 are calculated on lines 227-246. In lines 230-233 the R_3^0 rotation matrix is found by plugging the calculated values for the first three joint angles into the rotation portion of the homogeneous transform from frames 0-3. Line 236 applies the correction between the DH convention and URDF reference frames. Line 238 is where the R_6^3 rotation matrix is calculated. Because the inverse of a valid rotation matrix is equal to its transposition, the transposition of R_3^0 is used when calculating R_6^3. To find θ_4, θ_5, and θ_6 the euler_from_matrix() function is used with a yzy rotation. The reason a yzy rotation is used is because from the orientation at joint 3, the next joints 4, 5, and 6 rotate along the Y, Z, and Y axis respectively. This was one of the most difficult parts of the project to understand and calculate correctly, I was finally able to understand it thanks to helpful posts in the slack community.

One of the improvements I would make to this project would be to speed up the inverse kinematics calculations. Moving most of the sympy functions outside the for loop was helpful, but the inverse kinematics still take on the order of 30 seconds to a minute to calculate. One student in the slack community reported that they were able to increase the calculation speed by rewriting the calculations using the numpy package instead of sympy.

Another potential area of improvement is with the path the arm takes. Sometimes the wrist will rotate approximately a full revolution only to rotate right back. The debug script reports that the end effector position is found with near zero error for all the test cases, but for one of the test cases some of the angles are off by approximately 2π radians. My hunch is that modularly dividing the calculated joint angles by 2π might improve the problem, but I have not tested this hypothesis.