Udacity Robotics Software Engineer

Project 2

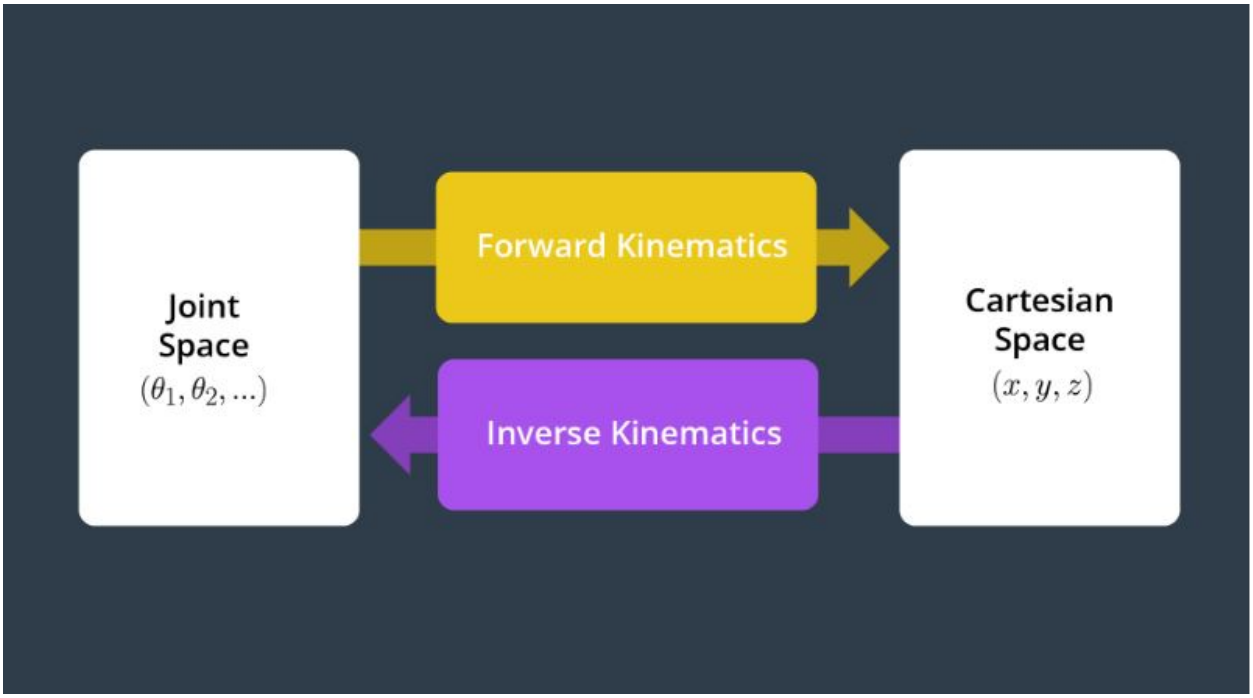
Robotic Arm – Pick and Place

By Leroy Friesenhahn

This was the most difficult project I have worked on in a long time. Understand deriving DH parameter table was my greatest challenge. I realize I have a long way to go to understand all there is to know about Forward Kinematics and Inverse Kinematics.

Kinematics

According to Wikipedia, Kinematics is a branch of classical mechanics that describes the motion of points, bodies and system of bodies without considering the mass of each of the forces that caused the motion. In this project, we will use Forward and Inverse Kinematics to solve the pick and place of the Kuba KR210 robot.



Forward Kinematics uses joint variable to determine the position and orientation of the KR210 gripper arm in 3D space (Cartesian Space).

The process uses composition of homogeneous transforms to solve this problem. Starting at the base link and moving link by link to the end effector. Denavit-Hartenberg (DH) convention is used in solving kinematic analysis for the KR210 robot. Using four parameter instead of twelve parameters help to simplify the analysis process. The DH parameters are used to build each transform

$$ {}^{0}_{N}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T \ldots {}^{N-1}_{N}T $$

.

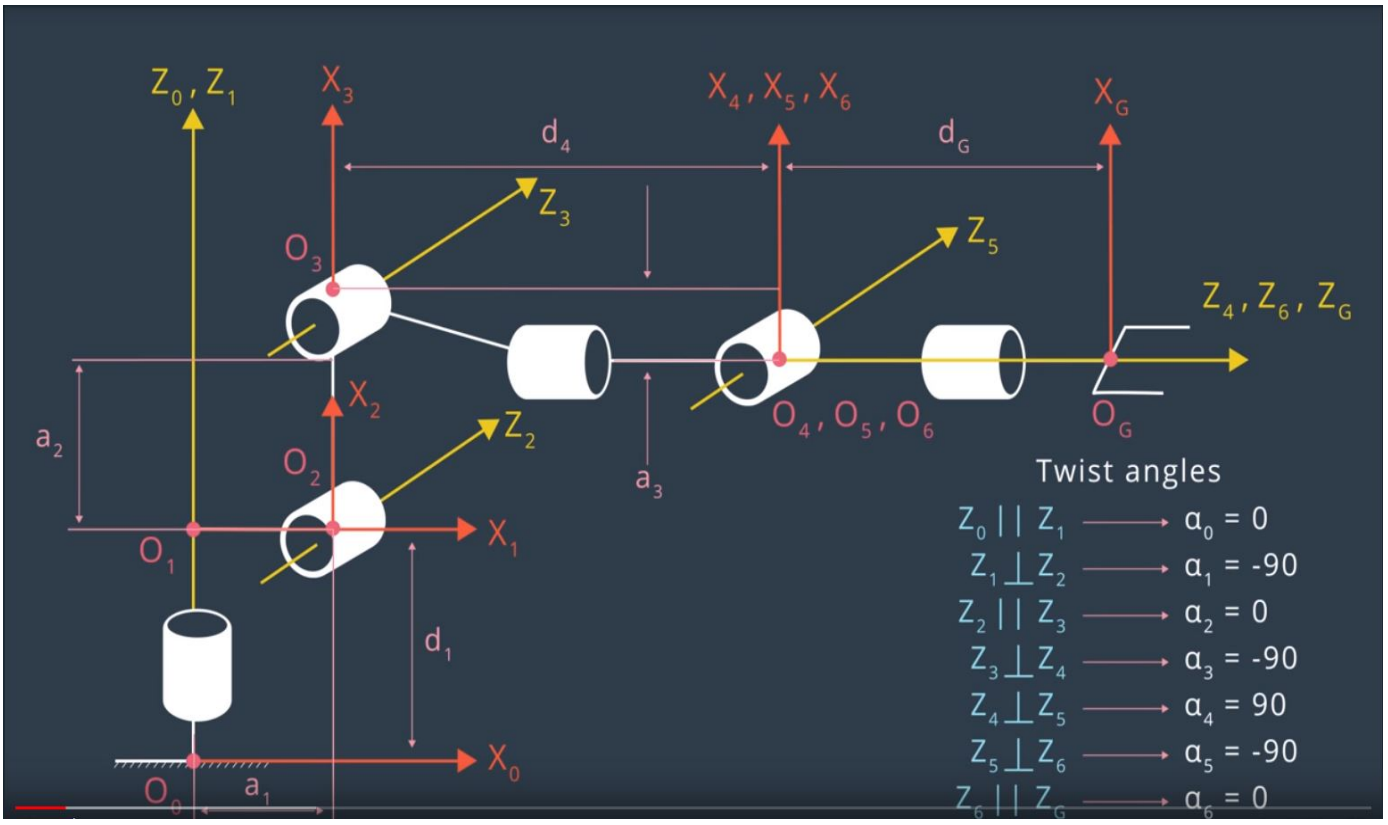
Total transform between adjacent links

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

Each link consist of four individual transforms: 2 rotations and 2 translation, performed in the order shown below.

$$ {}^{i-1}_{i}T = R_X(\alpha_{i-1})\, D_X(a_{i-1})\, R_Z(\theta_i)\, D_Z(d_i) $$

The analysis begins by drawing the diagram for the KR210 robot. Assigning the Z axis, X axis and gives us the necessary information to derive the values needed for the DH parameter table.



From the above diagram the following DH parameter table is derived. Where :

Alpha = the robot arm twist angle

a = arm link length

d – arm link offset

q – arm joint angle

Note: Do to lack of subscripts usage in the creation of this document, the following ledger applies

Alpha(i-1) – alpha value for link I -1

a(i-1) – a value for link i-1

d(i) – d value for link i

q(1) - theta value for designated link

DH Parameter Table

Link	Alpha (i-1)	a (i-1)	d(i)	Theta
0 to 1	0	0	d1	q(1)
1 to 2	-90	a1	0	q(2) – 90
2 to 3	0	a2	0	q(3)
3 to 4	-90	a3	d4	q(4)
4 to 5	90	0	0	q(5)
5 to 6	-90	0	0	q(6)
6 to G	0	0	d7	q(7)

Note: 7 is the Gripper

Using the Rviz Display for each of the links the following variables values can be obtained:

d1 = link2[z] = 0.75

a1 = link2[x] = 0.35

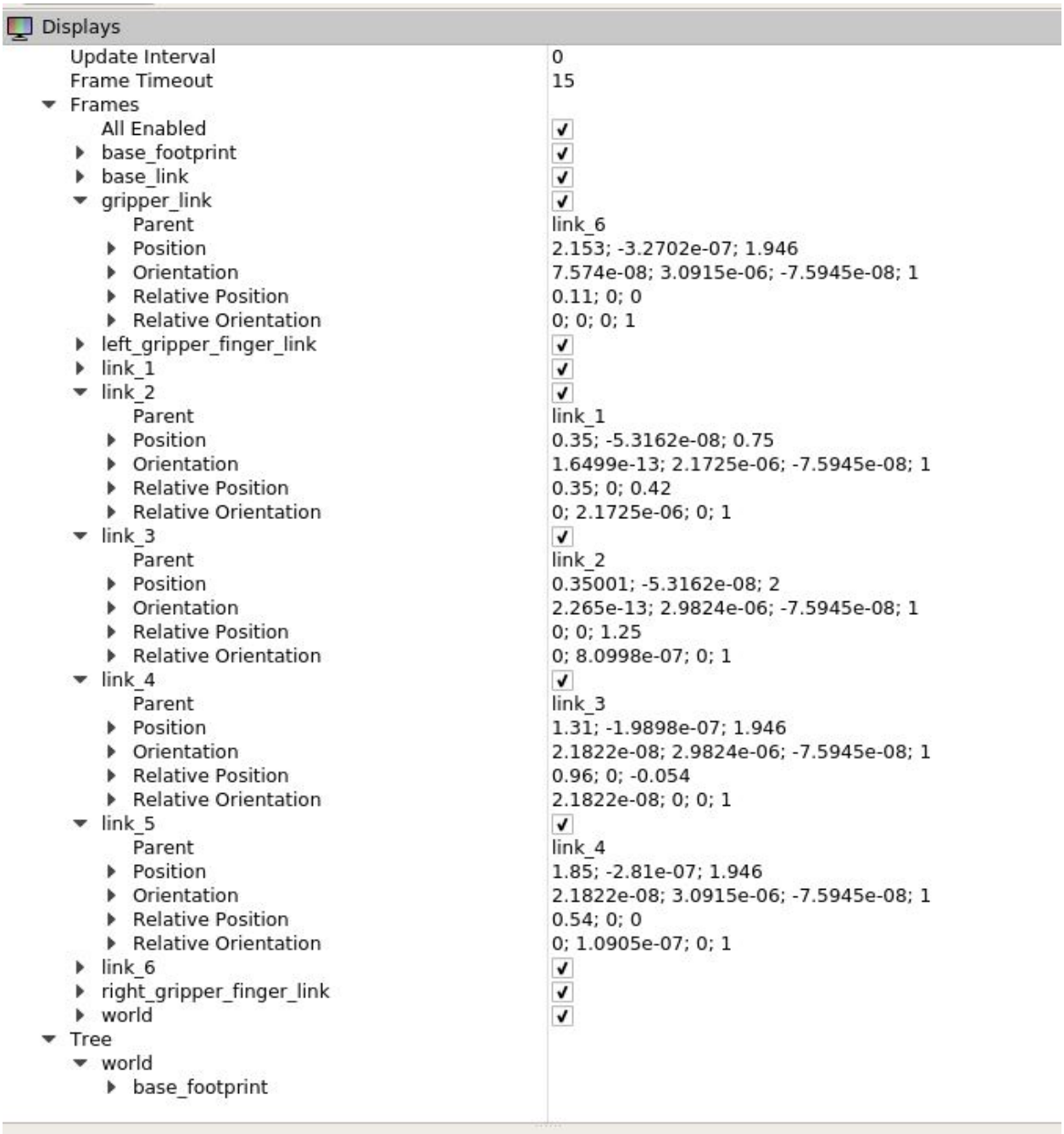
a2 = link3[z] = 1.25

a3 = link3[z] – link5[z] = 1.9464 - 2 =-- 0.054

d4 = link5[x] – link3[x] = 1.85 – 0.35001 = 1.5

Gripper: d7 = link_gripper[x] – link5[x] = 2.153 – 1.85 = 0.303

See view below:



Using Python code, create the transformation matrix:

```
# Creates Homogeneous Transform Matrix
#  from DH parameters
def TF_Matrix(q, d, a, alpha):
   T = 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 T
```

The homogeneous_transform is call for each individual joint to create each transform

```
# Generate each ndividual transformation matrices
# Transform from joint 0 to 1
T0_1 =TF_Matrix(alpha0,a0,d1,q1).subs(DH)
# Transform from joint 1 to 2
T1_2 = TF_Matrix(alpha1, a1, d2, q2).subs(DH)
# Transform from joint 2 to 3
T2_3 = TF_Matrix(alpha2, a2,d3,q3).subs(DH)
# Transform from joint 3 to 4
T3_4 = TF_Matrix(alpha3,a3,d4,q4).subs(DH)
```

Transform from joint 4 to 5

T4_5 = TF_Matrix(alpha4,a4, d5,q5).subs(DH)

Transform from Joint 5 to 6

T5_6 = TF_Matrix(alpha5,a5,d6,q6).subs(DH)

Transform from joint 6 to Gripper

T6_G = TF_Matrix(alpha6, a6,d7,q7).subs(DH)

The final generalized homogeneous transform is shown mathematically:

$$ {}^{0}_{N}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T \ldots {}^{N-1}_{N}T $$

Final Forward Kinematics is written as:

T0_2 = T0_1 * T1_2

T0_3 = T0_2 * T2_3

…

T0_EE = T0_6 * T0_EE

The Rotation Matrix can be found:

R0_1 = T0_1[0:3,0:3]

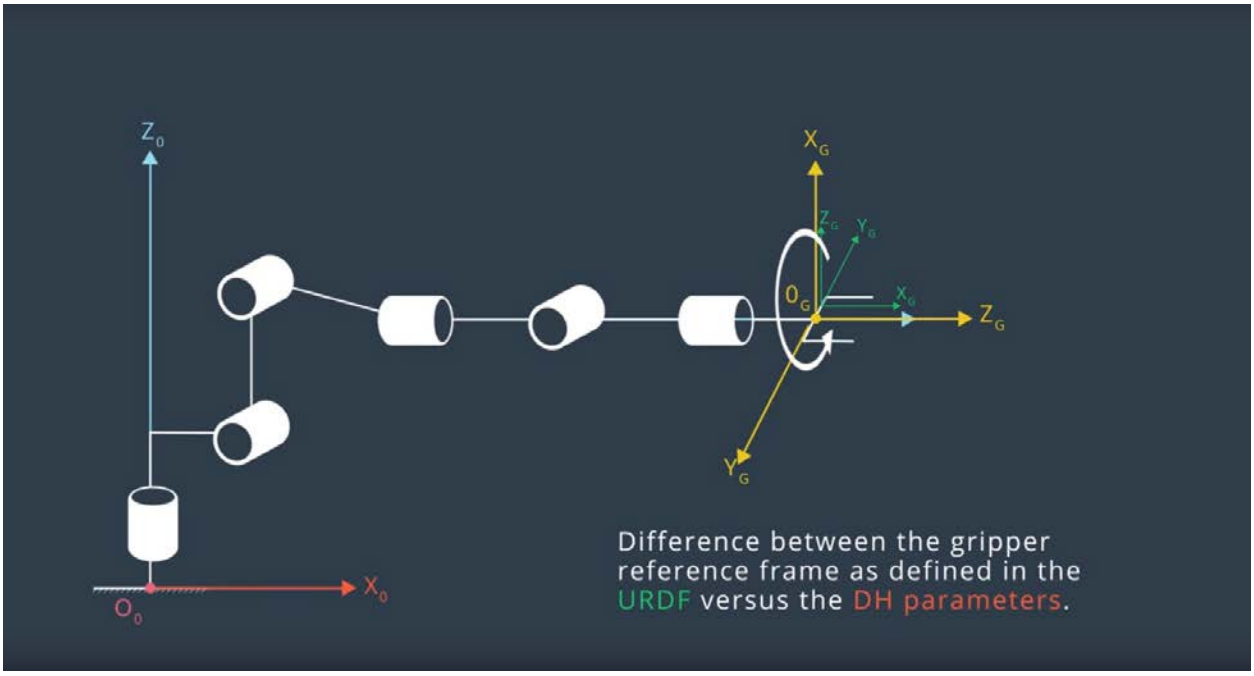
R0_2 = T0_2[0:3,0:3]

…

R0_EE = T0_EE[0:3,0:3]

Inverse Position Kinematics

In robotics, inverse kinematics uses kinematics equations to determine the joint parameters that provide the desired position of the robot's end-effectors. For this to occur with the KR210, one needs to account for the discrepancy between the DH parameters and the Gazebo URDF:



This can be corrected by rotating the Z axis by 180 degrees followed by a rotation of the Y axis of -90 degrees

Using the values r – rotate, p-pitch, y – yaw

R_rpy = Matrix([[cos(y)*cos(p), cos(y)*sin(p)*sin(r)-sin(y)*cos(r), cos(y)*sin(p)*cos(r)+sin(y)*sin(r)],

[sin(y)*cos(p), sin(y)*sin(p)*sin(r)+cos(y)*cos(r), sin(y)*sin(p)*cos(r)-cos(y)*sin(r)],

[-sin(p), cos(p)*sin(r), cos(p)*cos(r)]])

R_corr = Matrix([[0,0,1],

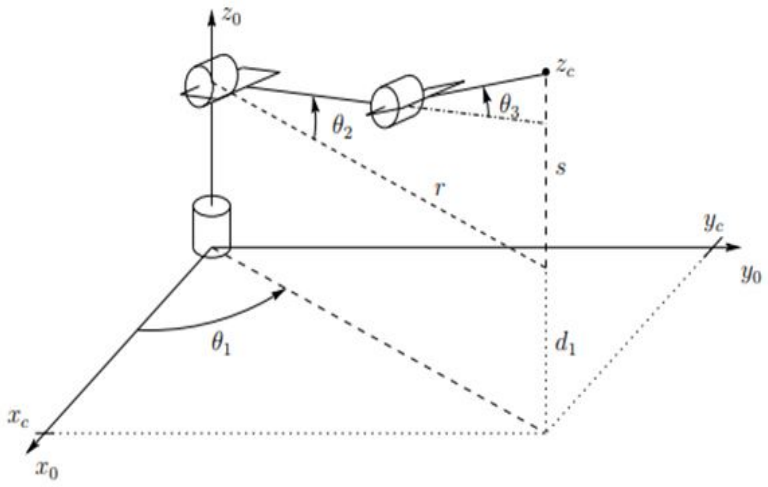
[0,-1,0],

[1,0,0]])

The Correction becomes: Rrpy = R_rpy * (R_corr.T)

Solving the inverse kinematics for the Kuka KR210 consists of dividing the robot into two parts:

Part 1 – joints 1 thru 3.



Geometrically, the angles and length can be obtained:

Part 2 – joint 4 thru 6 (wrist with gripper) : The previous three joint angles can be used to obtain the wrist center. Using the previous information, the following rotation matrix is used to solve for the remaining three angles

$$ {}^{3}_{6}R = \left({}^{0}_{3}R\right)^{-1} {}^{0}_{6}R = \left({}^{0}_{3}R\right)^{T} {}^{0}_{6}R $$

This results in:

$$ {}^{A}_{B}R_{XYZ} = R_Z(\alpha)R_Y(\beta)R_X(\gamma) $$
$$ = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix} $$

Solving this matrix for joint angles 4, 5 and 6 complete the inverse kinematics for KR210

Conclusion:

The code completed its task. It successfully grabbed the cylinder and placed into the barrel. I had to use an android phone camera to make the video since any video recording software on the vmplayer resulted in the simulator running very slow(unbearably slow). I used the next button to step through the process to allow the computer ample time

to perform the simulation. The continue button failed to allow sufficient time for the gripper fingers to close.

This was a very difficult project for me. I identified weakness in my matrix operations skills as also in my ability to identify and assign DH parameters. My slow workstation hindered my development, debugging and testing phases. I would like to thank the countless individual who posted videos, documents, and help hints on the internet including U-Tube, which helped me in understand DH Parameters determination, Matrix manipulations and KR210. The 40+ hours spend on these items only scratch the surface in letting me know area I need additional assistance. Thank you all.