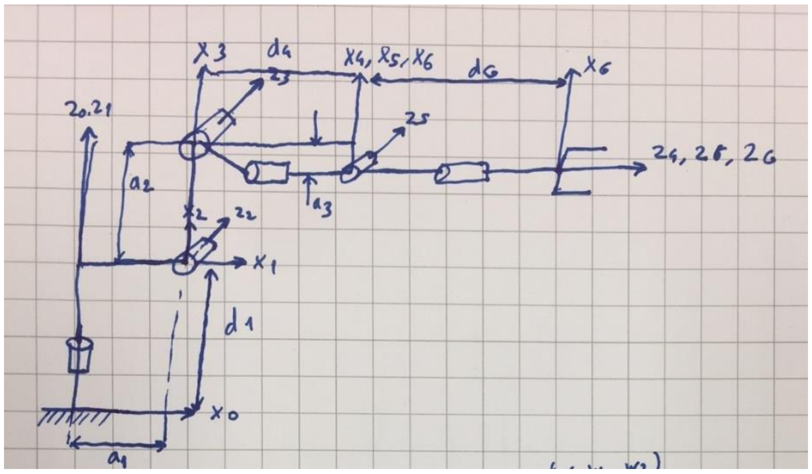
Project 2 : Robotic Arm Pick & Place, Udacity Robotics Nanodegree

Bahadır Özkan, October 2017

Modified DH-Parameters of the 6 DOF KUKA KR2110 can be found from the figure below with the help of the URDF file information provided with the project documents:



URDF file defines the pose of the robot when all joint variables are equal to zero and provides the numerical values for the DH parameter. Link twist alphas are measured according to the right hand rule.

i	alpha[i-1]	a[i-1]	d[i]	theta[i]
1	0	0	0.75	q1
2	-pi/2	0.35	0	q2-pi/2
3	0	1.25	0	q3
4	-pi/2	-0.54	1.50	q4
5	pi/2	0	0	q5
6	-pi/2	0	0	q6
7	0	0	0.303	0

Where,

- alpha[i-1] is the LINK TWIST, angle between axis z[i-1] and axis z[i] measured about axis x[i-1]
- a[i-1] is the LINK LENGTH, distance from axis z[i-1] to axis z[i] measured along axis x[i-1]
- d[i] is the LINK OFFSET, distance from axis x[i-1] to axis x[i] measured along axis z[i]
- theta[i] is the JOINT ANGLE, angle about previous z[i] from old x[i-1] to new x[i]

The pose of the joint frame given the DH parameters can be found with the following:

$$ {}^{i-1}_{i}T = R(x_{i-1}, \alpha_{i-1})\, T(x_{i-1}, a_{i-1})\, R(z_i, \theta_i)\, T(z_i, d_i) $$

Which results to:

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

With the transformation information above the following code is used to create the matrices which takes the DH table as input:

```
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)
```

And the final transformation can be found as:

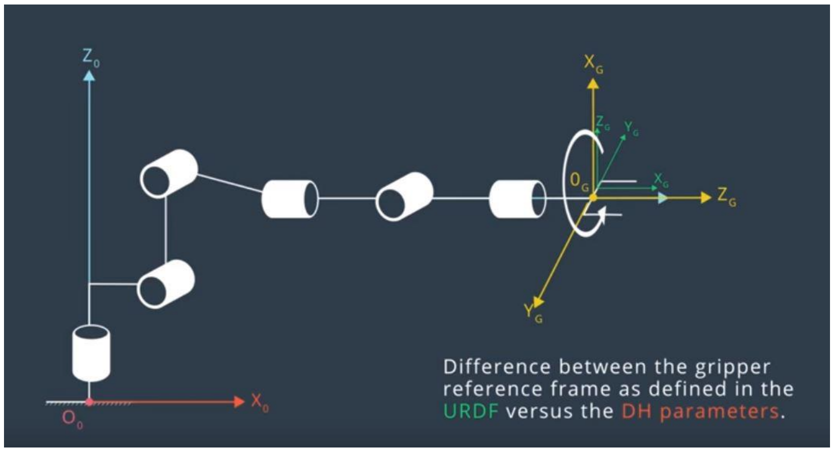
```
T0_EE = T0_1 * T1_2 * T2_3 * T3_4 * T4_5 * T5_6 * T6_EE
```

In addition to the individual transformation matrices total homogeneous transform from the base-link to the end effector can be shown as below:

$$T = \left[\begin{array}{ccc|c} & & & P_x \\ & R_T & & P_y \\ & & & P_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Where Px, Py and Pz are the position of the end effector with respect to the base-link and R_T is the rotation.

The rotation part of the above transformation is as follows:


Z0
X0
O0
XG
ZG
YG
OG
Difference between the gripper reference frame as defined in the URDF versus the DH parameters.

The code to get the rotation matrices is as follows:

```
# Extract rotation matrices from the transformation matrices
r, p, y = symbols('r p y')

ROT_x = Matrix([[1,      0,       0],
               [0, cos(r), -sin(r)],
               [0, sin(r),  cos(r)]])  # Roll
ROT_z = Matrix([[cos(y), -sin(y), 0],
              [sin(y),   cos(y), 0],
              [0,            0, 1]])  # Yaw
ROT_y = Matrix([[cos(p), 0, sin(p)],
              [      0, 1,      0],
              [-sin(p), 0, cos(p)]])  # Pitch

ROT_EE = ROT_z * ROT_y * ROT_x
```

By applying the angles on the ROT matrices end effector’s transformation can be found.

```
Rot_error = ROT_z.subs(y, radians(180)) * ROT_y.subs(p, radians(-90))
ROT_EE = ROT_EE * Rot_error
ROT_EE = ROT_EE.subs({'r': roll, 'p': pitch, 'y': yaw})
```

After getting the ROT_EE, wrist center can be found with the formula below

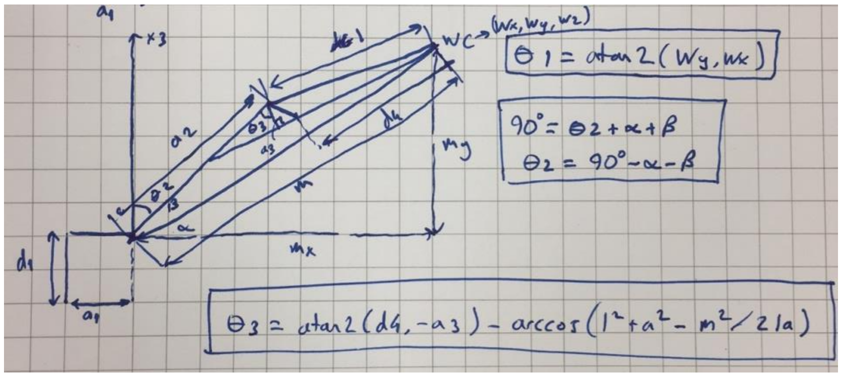
$$w_x = p_x - (d_6 + l) \cdot n_x$$
$$w_y = p_y - (d_6 + l) \cdot n_y$$
$$w_z = p_z - (d_6 + l) \cdot n_z$$

And with the code below the wrist center is achieved. 0.303 is the d6 parameter from the DH table, ROT_EE[:,2] gives the vales for nx, ny and nz.

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

# Calculate joint angles using Geometric IK method
WC = EE - 0.303 * ROT_EE[:,2]
```

After defining the Wrist Center (WC) coordinates, first three joint angles that are needed for the forward kinematics can be found using the trigonometric functions.

Below illustrations would be helpful to understand where the angles are located and how they can be calculated.

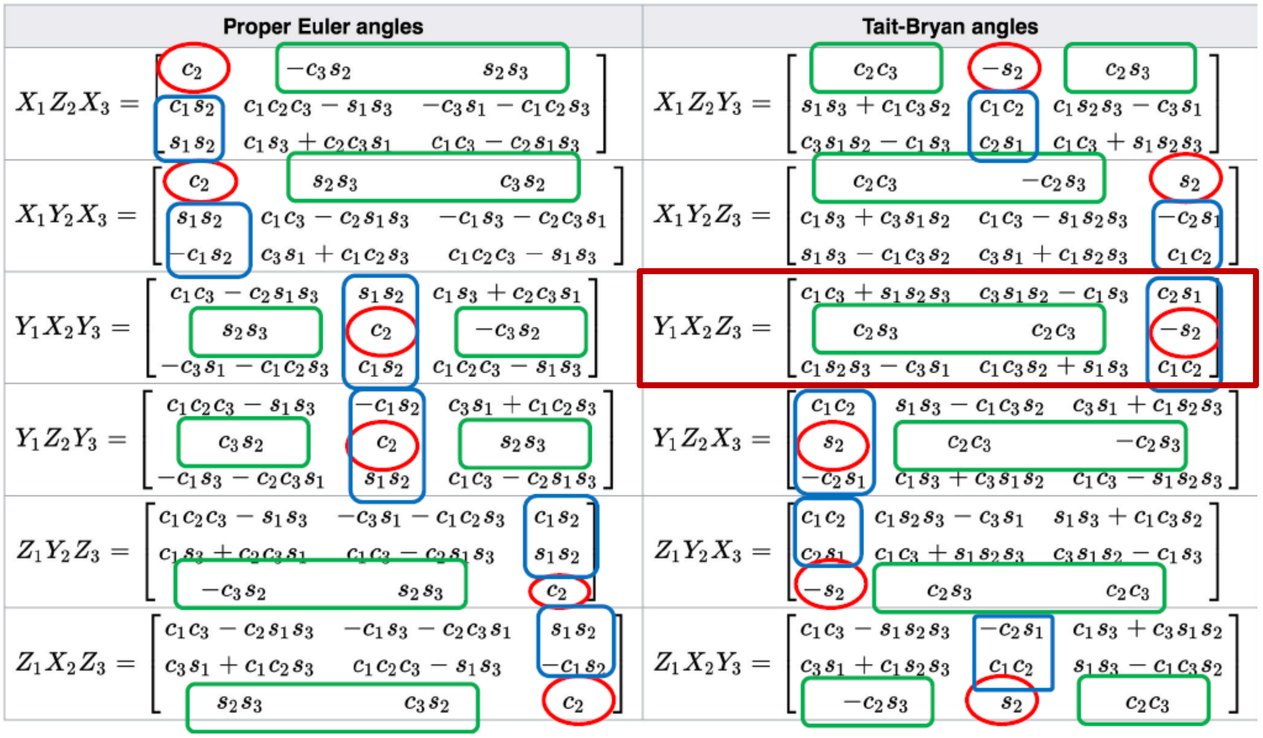


Code to find the first three joint angles is as follows:

```
theta1 = atan2(WC[1],WC[0])

l = sqrt(pow(s[d4],2) + pow(-s[a3],2))
m1 = sqrt(pow(WC[0],2) + pow(WC[1],2))
mx = m1 - s[a1]
my = WC[2]  - s[d1]
m = sqrt(pow(mx,2) + pow(my,2))
phi = atan2(s[d4],-s[a3])
alpha = atan2(my,mx)

side_a = m
side_b = s[a2]
side_c = l

   gamma = acos((side_b * side_b + side_c * side_c - side_a * side_a) / (2 * side_b * side_c))
beta = acos((side_b * side_b + side_a * side_a - side_c * side_c) / (2 * side_b * side_a))

theta2 = pi/2 - alpha - beta
theta3 = phi - gamma
```

There are 12 possible solutions to derive theta4,5 and 6 from the Euler angles from rotation matrices. YXZ Tait-Bryan notation shown with the red box is used in this case.

From Wikipedia:



Proper Euler angles	Tait-Bryan angles
$X_1Z_2X_3 = \begin{bmatrix} c_2 & -c_3 s_2 & s_2 s_3 \\ c_1 s_2 & c_1 c_2 c_3 - s_1 s_3 & -c_3 s_1 - c_1 c_2 s_3 \\ s_1 s_2 & c_1 s_3 + c_2 c_3 s_1 & c_1 c_3 - c_2 s_1 s_3 \end{bmatrix}$	$X_1Z_2Y_3 = \begin{bmatrix} c_2 c_3 & -s_2 & c_2 s_3 \\ s_1 s_3 + c_1 c_3 s_2 & c_1 c_2 & c_1 s_2 s_3 - c_3 s_1 \\ c_3 s_1 s_2 - c_1 s_3 & c_2 s_1 & c_1 c_3 + s_1 s_2 s_3 \end{bmatrix}$
$X_1Y_2X_3 = \begin{bmatrix} c_2 & s_2 s_3 & c_3 s_2 \\ s_1 s_2 & c_1 c_3 - c_2 s_1 s_3 & -c_1 s_3 - c_2 c_3 s_1 \\ -c_1 s_2 & c_3 s_1 + c_1 c_2 s_3 & c_1 c_2 c_3 - s_1 s_3 \end{bmatrix}$	$X_1Y_2Z_3 = \begin{bmatrix} c_2 c_3 & -c_2 s_3 & s_2 \\ c_1 s_3 + c_3 s_1 s_2 & c_1 c_3 - s_1 s_2 s_3 & -c_2 s_1 \\ s_1 s_3 - c_1 c_3 s_2 & c_3 s_1 + c_1 s_2 s_3 & c_1 c_2 \end{bmatrix}$
$Y_1X_2Y_3 = \begin{bmatrix} c_1 c_3 - c_2 s_1 s_3 & s_1 s_2 & c_1 s_3 + c_2 c_3 s_1 \\ s_2 s_3 & c_2 & -c_3 s_2 \\ -c_3 s_1 - c_1 c_2 s_3 & c_1 s_2 & c_1 c_2 c_3 - s_1 s_3 \end{bmatrix}$	$Y_1X_2Z_3 = \begin{bmatrix} c_1 c_3 + s_1 s_2 s_3 & c_3 s_1 s_2 - c_1 s_3 & c_2 s_1 \\ c_2 s_3 & c_2 c_3 & -s_2 \\ c_1 s_2 s_3 - c_3 s_1 & c_1 c_3 s_2 + s_1 s_3 & c_1 c_2 \end{bmatrix}$
$Y_1Z_2Y_3 = \begin{bmatrix} c_1 c_2 c_3 - s_1 s_3 & -c_1 s_2 & c_3 s_1 + c_1 c_2 s_3 \\ c_3 s_2 & c_2 & s_2 s_3 \\ -c_1 s_3 - c_2 c_3 s_1 & s_1 s_2 & c_1 c_3 - c_2 s_1 s_3 \end{bmatrix}$	$Y_1Z_2X_3 = \begin{bmatrix} c_1 c_2 & s_1 s_3 - c_1 c_3 s_2 & c_3 s_1 + c_1 s_2 s_3 \\ s_2 & c_2 c_3 & -c_2 s_3 \\ -c_2 s_1 & c_1 s_3 + c_3 s_1 s_2 & c_1 c_3 - s_1 s_2 s_3 \end{bmatrix}$
$Z_1Y_2Z_3 = \begin{bmatrix} c_1 c_2 c_3 - s_1 s_3 & -c_3 s_1 - c_1 c_2 s_3 & c_1 s_2 \\ c_1 s_3 + c_2 c_3 s_1 & c_1 c_3 - c_2 s_1 s_3 & s_1 s_2 \\ -c_3 s_2 & s_2 s_3 & c_2 \end{bmatrix}$	$Z_1Y_2X_3 = \begin{bmatrix} c_1 c_2 & c_1 s_2 s_3 - c_3 s_1 & s_1 s_3 + c_1 c_3 s_2 \\ c_2 s_1 & c_1 c_3 + s_1 s_2 s_3 & c_3 s_1 s_2 - c_1 s_3 \\ -s_2 & c_2 s_3 & c_2 c_3 \end{bmatrix}$
$Z_1X_2Z_3 = \begin{bmatrix} c_1 c_3 - c_2 s_1 s_3 & -c_1 s_3 - c_2 c_3 s_1 & s_1 s_2 \\ c_3 s_1 + c_1 c_2 s_3 & c_1 c_2 c_3 - s_1 s_3 & -c_1 s_2 \\ s_2 s_3 & c_3 s_2 & c_2 \end{bmatrix}$	$Z_1X_2Y_3 = \begin{bmatrix} c_1 c_3 - s_1 s_2 s_3 & -c_2 s_1 & c_1 s_3 + c_3 s_1 s_2 \\ c_3 s_1 + c_1 s_2 s_3 & c_1 c_2 & s_1 s_3 - c_1 c_3 s_2 \\ -c_2 s_3 & s_2 & c_2 c_3 \end{bmatrix}$

Joint angles theta4, theta5 and theta 6 are found with the code below:

```
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])
```

Finally with the 6 joints provided to KUKA KR2110 the robot arm can calculate the inverse kinematics and retrieve the object and drop to the bin.

More optimization can be done to avoid wrist flips. With the code available in the repository the IK_server.py can calculate the joint angles in about a second.