Robotics Nanodegree

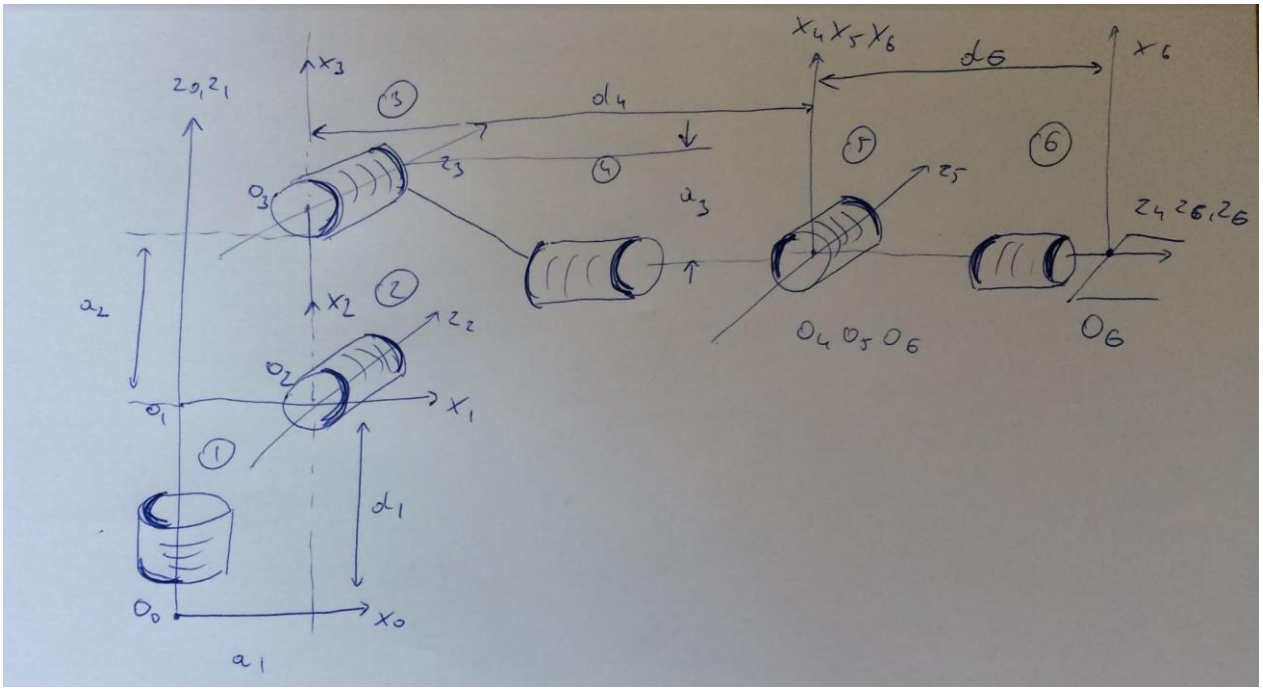
P2: Robotic Arm: Pick and Place

Markus Buchholz

Introduction

The goal for this project was to created forward and inverse kinematic model for KUKA KR210 robot arm. The model development, debugging and testing was performed by use of ROS and Rviz. Developed kinematic model allows to pick objects from shelf and place them to refuse heap (8/10). Debug process was performed in the same development environment where several robot position and joint angles(orientations) were verified.

Kinematic Analysis



1. **DH Parameters**

KUKA robot kinematic model can be presented as follows. Following sketch allowed to defined the DH parameters.

i	alpha(i-1)	a (i-1)	d (i)	q (i)
1	0	0	0.75	
2	-pi/2	0.35	0	q2: q2-pi/2
3	0	1.25	0	
4	-pi/2	-0.054	1.50	
5	pi/2	0	0	
6	-pi/2	0	0	
7	0	0	0.303	q7: 0

2. Transformation Matrices

Homogenous transform matrix TF_Matrix is a skeleton in order to compute individual transform matrices about each joint (using the DH_table).

```
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
```

```
 DH_table = {
     alpha0:       0,  a0:      0,  d1:  0.75, q1:  q1,
     alpha1: -pi/2.0,  a1:   0.35,  d2:     0, q2:  -pi/2.0 + q2,
     alpha2:       0,  a2:   1.25,  d3:     0, q3:  q3,
     alpha3: -pi/2.0,  a3: -0.054,  d4:   1.5, q4:  q4,
     alpha4:  pi/2.0,  a4:      0,  d5:     0, q5:  q5,
     alpha5: -pi/2.0,  a5:      0,  d6:     0, q6:  q6,
     alpha6: 0,        a6:      0,  d7: 0.303, q7:  0}
```

Homogeneous transform matrix T0_EE from base_link to gripper_link can be given by following matrix equations:

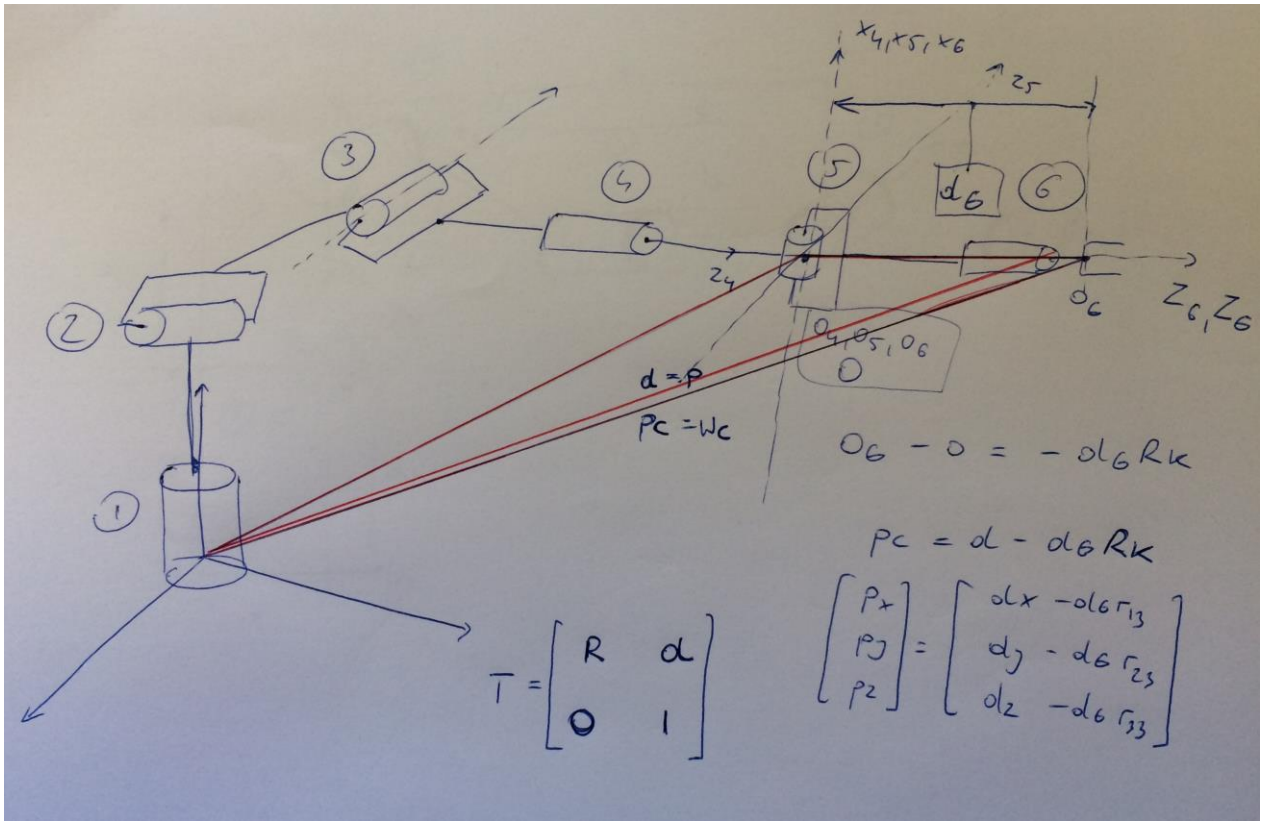
```
T0_1 = TF_Matrix(alpha0, a0, d1, q1).subs(DH_Table)
T1_2 = TF_Matrix(alpha1, a1, d2, q2).subs(DH_Table)
T2_3 = TF_Matrix(alpha2, a2, d3, q3).subs(DH_Table)
T3_4 = TF_Matrix(alpha3, a3, d4, q4).subs(DH_Table)
T4_5 = TF_Matrix(alpha4, a4, d5, q5).subs(DH_Table)
T5_6 = TF_Matrix(alpha5, a5, d6, q6).subs(DH_Table)
T6_EE = TF_Matrix(alpha6, a6, d7, q7).subs(DH_Table)


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

3. Decouple Inverse Kinematics

Inverse kinematic for 6 axis robot will be performed by kinematical decoupling, where in order to solve the whole problem can be divided by two subtasks: 1. Position inverse kinematic problem and 2. Orientation inverse kinematic problem.

a. Position inverse kinematic problem

Analyzing following figure is can be estimated that:



R0_6 = R_EE (Euler Rotation matrix: roll, pitch, yaw).

wc = p - dG*R*k

wc = p - dG*R_EE*k, where k = [0,0,1].T

p = [Px, Py, Pz] = end-effector positions

Wc = [Wx, Wy, Wz] = wrist positions

In spherical wrist, axes z4, z5 and z6 intersects in the same O point. Therefore O4, O5 and O6 are placed in the same place in the middle of wrist. This affects that movement of axis 4, 5 or 6 does not influence on position of point O, therefore position of O is the function of theta1, theta2 and theta3.

```
#roll
ROT_x = Matrix([[1, 0,  0          ],
                [0, cos(r), -sin(r)],
                [0, sin(r), cos(r) ]])
#pitch
ROT_y = Matrix([[cos(p),0   , sin(p)],
                [0, 1, 0            ],
                [-sin(p),   0, cos(p)]])
#pitch
ROT_z = Matrix([[cos(y), -sin(y), 0   ],
                [sin(y),    cos(y), 0],
                [0,  0, 1             ]])


ROT_EE =  ROT_z * ROT_y * ROT_x
```

Applying the correction (URDF frame is different):

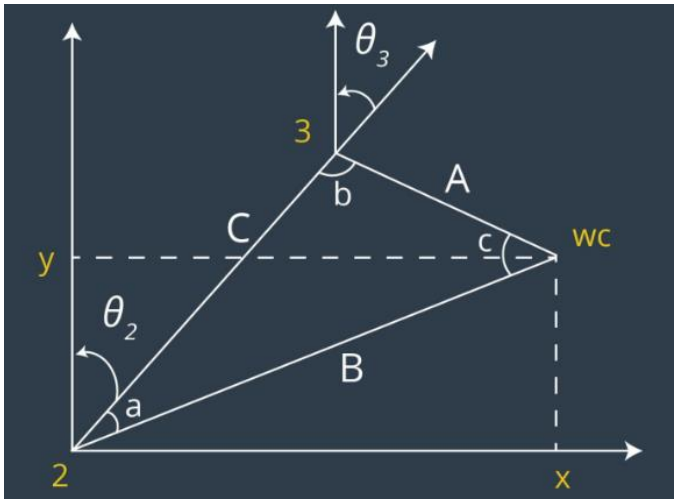
The ROT_EE can be given as follows:

```
Rot_Error = ROT_z.subs(y, radians(180)) * ROT_y.subs(p, radians(-90))

ROT_EE = ROT_EE * Rot_Error
```

Matrix ROT_EE is multiplied by vector **k** therefore only last columns is further applied. Finally the wrist positions is given by (in this project dG = 0.303), px,py and pz is given by user (robot trajectory) :

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

EE = Matrix([[px],
             [py],
             [pz]]) # end-effector positions
```

In order to compute theta 1, theta2 and theta3 below triangle should be evaluated.



Analyzing the below triangle with the position of wrist WC, calculating theta1 can be done as follows:

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

Applying trigonometric transformations (below) the theta2 and theta3 can be computed as follows:

```
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
theta2 = pi / 2 - angle_a - atan2(WC[2] - 0.75, sqrt(WC[0] * WC[0] + WC[1] * WC[1]) - 0.35)


#theta3
theta3 = pi / 2 - (angle_b + 0.036)
```

b. Orientation inverse kinematic problem

Since the overall RPY (Roll Pitch Yaw) rotation between base_link and gripper_link must be equal to the product of individual rotations between respective links, following holds true:

R0_6 = R (Euler Rotation matrix: roll, pitch, yaw).

R = R0_3 * R3_6

R3_6 = inv(R0_3) * R = (R0_3).T * R

R3_6 = (R0_3).T * R := U, where U is a Euler Matrix

```
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
```

Solving above equation (taking into account rotation matrix R3_6 and Euler Matrix) theta4, theta5 and theta6 are given by following equations:

```
#theta4
theta4 = atan2(R3_6[2,2], -R3_6[0,2])


#theta5
theta5 = atan2(sqrt(R3_6[0,2]*R3_6[0,2] + R3_6[2,2]*R3_6[2,2]),R3_6[1,2])


#theta6
theta6 = atan2(-R3_6[1,1], R3_6[1,0])
```

Project Implementation

In IK_Server.py the Python code with implemented solution was included. Solution calculates Inverse Kinematics based on previously performed Kinematic Analysis. Robot successfully fulfill project requirements and complete 8/10 pick and place cycles.