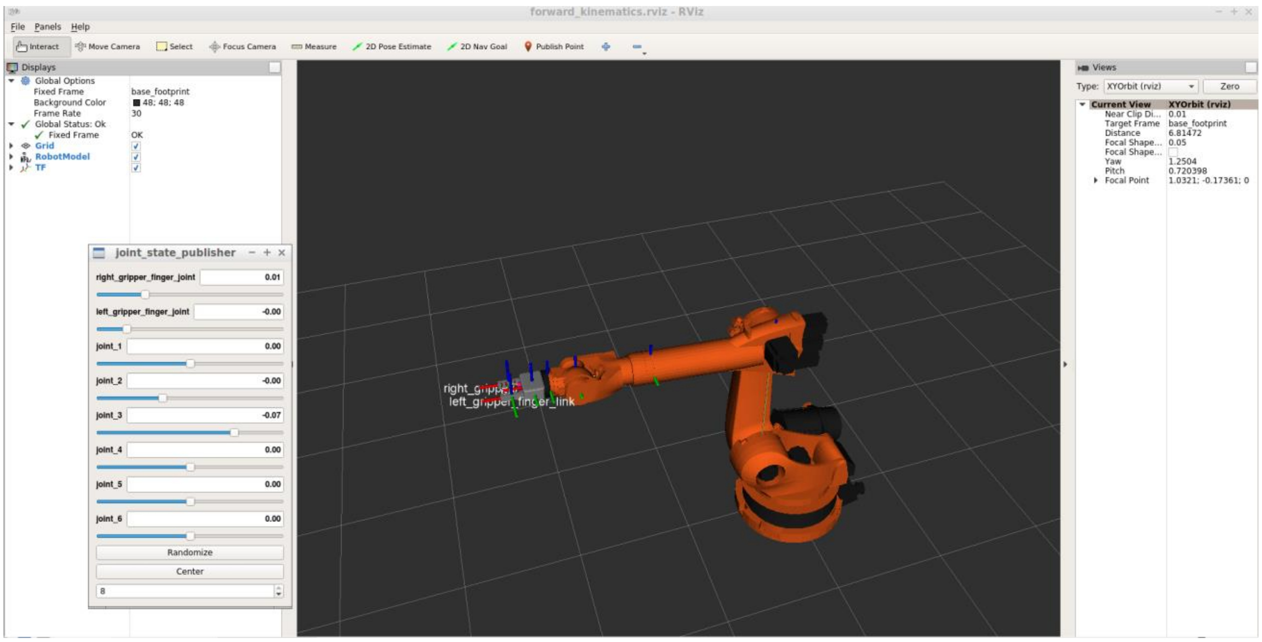
ISRAILOV SARDOR

README (Pick and Place)

Implementation of the pick&place Project Udacity Robotics Nanodegree

Kinematic Analysis



The project consists of implementing a mathematical model for 6-axis kuka-arm, then the python code. At first we find a DH-table, then transformation matrix, and complete FK part i.e find the position and orientation of a gripper knowing each angle of each joint(motor).

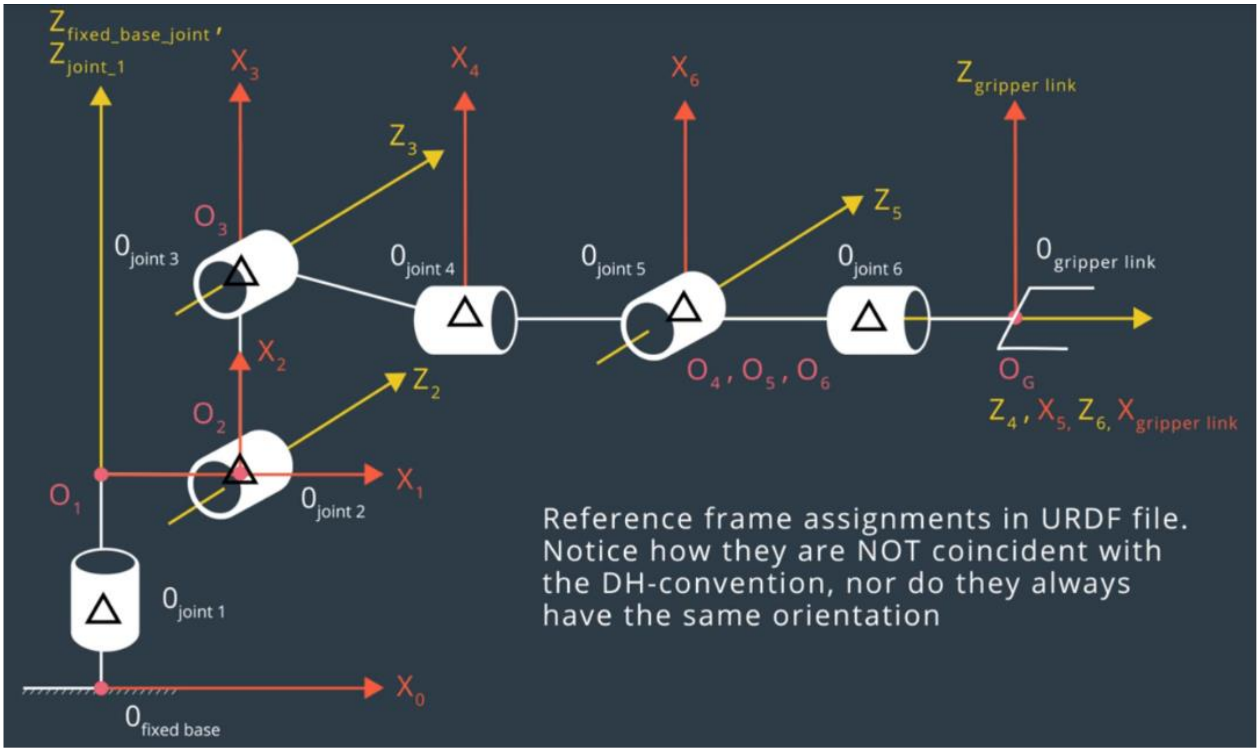
The first observations show that the robot lies essentially in the x,z plane.

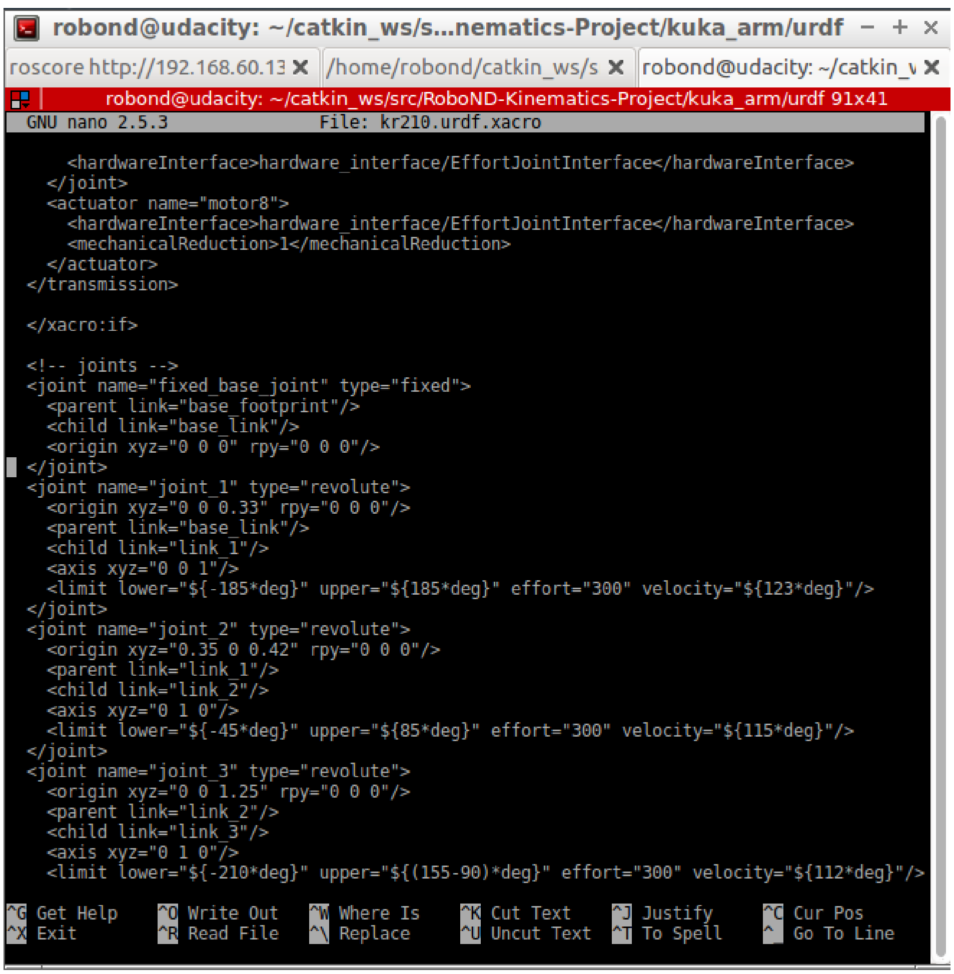
With commands:

~/catkin_ws/src/RoboND-Kinematics-Project/kuka_arm/urdf

nano kr210.urdf.xacro,

we obtain information about joints,liks,actuators etc. From this we derive DH table. The second method to construct the table is to use rviz, by using “measure tool” and joint_state_publisher.





First of all, we need to identify our joints and links. That's why Z2//Z3//Z5, moreover Z4 and Z6 are coincident. We have a special point called the wrist center, which is the center of reference frames 4,5,6, because they all intersect at the same point. We have 7 coordinate reference frames, that we try to construct as simple as possible. In the initial position, many angles are at +/-90 degrees. We see that the sum

i	Links	alpha(i-1)	a(i-1)	d(i)	theta(i)
1	0->1	0	0	0.75	q1
2	1->2	- pi/2	0.35	0	-pi/2 + q2
3	2->3	0	1.25	0	q3
4	3->4	- pi/2	-0.054	1.5	q4
5	4->5	pi/2	0	0	q5
6	5->6	- pi/2	0	0	q6
7	6->EE	0	0	0.303	q7

of 0.33+0.42=0.75, which is the projection from the distance from O_0 to O_1. From O_3 to O_4, we have the offset of d=0.96+0.54=1.5. Note that O_4 O_5 O_6 are at the same point so they have no link length or offset. They differ though by twist angles. The end effector differs from link_6 only by offset of 0.193+0.11=0.303.

Theta2=angle between x(i-1) and x(i) measured in the zi axis in the right hand sense. That's why there is offset of -pi/2.

In order to align the frame EE and the base_frame, we need to rotate the gripper intrinsically around z-axis about 180degrees and y axis by the angle of –pi/2.

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.

ISRAILOV SARDOR

The general transform matrix is:

$\cos(theta_i)$	$-\sin(theta_i)$	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

Note: the first three columns and rows are called rotation matrix; they are responsible for rotation; the fourth row is responsible for translation from one system of coordinates to another.

That's why when we substitute the values from the DH table into the matrix we obtain:

Base Link –> Joint1(some terms are zero due to the fact that **$\sin(alpha_{i-1})$=0**)

cos(q1)	-sin(q1)	0	0
sin(q1)	cos(q1)	0	0
0	0	1	0.75
0	0	0	1

ISRAILOV SARDOR

Joint1–> Joint2

cos(-pi/2 + q2)	-sin(-pi/2 + q2)	0	0.35
sin(-pi/2 + q2) * cos(-pi/2)	cos(-pi/2 + q2) * cos(-pi/2)	-sin(-pi/2)	0
sin(-pi/2 + q2) * sin(-pi/2)	cos(-pi/2 + q2) * sin(-pi/2)	cos(-pi/2)	0
0	0	0	1

Which leads to:

sin(q2)	cos(q2)	0	0.35
0	0	1	0
cos(q2)	- sin(q2)	0	0
0	0	0	1

ISRAILOV SARDOR

Joint2–> Joint3

cos(q3)	-sin(q3)	0	1.25
sin(q3)	cos(q3)	0	0
0	0	1	0
0	0	0	1

Joint3–> Joint4

cos(q4)	-sin(q4)	0	-0.054
sin(q4) * cos(-pi/2)	cos(q4) * cos(-pi/2)	-sin(-pi/2)	-sin(-pi/2) * 1.50
sin(q4) * sin(-pi/2)	cos(q4) * sin(-pi/2)	cos(-pi/2)	cos(-pi/2) * 1.50
0	0	0	1

Which leads to:

cos(q4)	-sin(q4)	0	-0.054
0	0	1	1.50
-sin(q4)	-cos(q4)	0	0
0	0	0	1

Joint4–> Joint5

cos(q5)	-sin(q5)	0	0
0	0	-1	0
sin(q5)	cos(q5)	0	0
0	0	0	1

ISRAILOV SARDOR

Joint5–> Joint6

cos(q6)	-sin(q6)	0	0
0	0	1	0
-sin(q6)	-cos(q6)	0	0
0	0	0	1

Joint6–> end Gripper Link

1	0	0	0
0	1	0	0
0	0	1	0.303
0	0	0	1

All the matrix multiplication of above gives the transformation of base link to gripper link:

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

The general form of this matrix is :

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

which consist of a rotation and translation part(Px(x direction),Py(y direction),Pz(z direction)) as mentioned above.

The rotation matrix can be obtained using individual rotation matrices along x,y,z axis of an end gripper.

So Roll,pitch,yaw: r,p,y;

Rot_x is

1	0	0
0	cos(r)	-sin(r)
0	sin(r)	Cos(r)

ISRAILOV SARDOR

Rot_y is

cos(p)	0	-sin(p)
0	1	0
-sin(p)	0	cos(p)

Rot_z is

cos(y)	-sin(y)	0
sin(y)	cos(y)	0
0	0	1

The final matrix becomes R0_6

cos(p)·cos(y)	sin(p)·sin(r)·cos(y) - sin(y)·cos(r)	sin(p)·cos(r)·cos(y) + sin(r)·sin(y)
sin(y)·cos(p)	sin(p)·sin(r)·sin(y) + cos(r)·cos(y)	sin(p)·sin(y)·cos(r) - sin(r)·cos(y)
-sin(p)	sin(r)·cos(p)	cos(p)·cos(r)

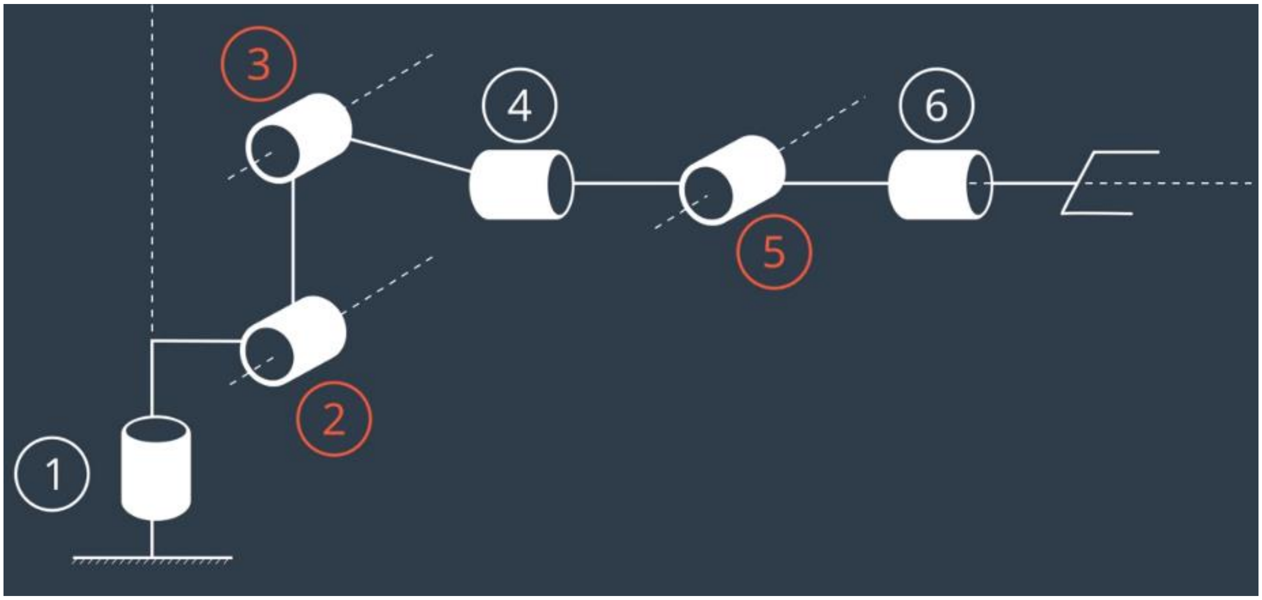
There is also rotation error due to the fact that the the frames in URDF file doesn't coincident with DH-convention. We apply intrisinsic rotations along z axis 180 degrees and the y axis by angle of -90 degrees. The error matrix of end gripper (Rot_z(180)*Rot_y(-90)):

0	0	1
0	-1	0
1	0	0

The R0_6 becomes R0_6 * Rot_Error:

sin(p)·cos(r)·cos(y) + sin(r)·sin(y)	-sin(p)·sin(r)·cos(y) + sin(y)·cos(r)	cos(p)·cos(y)
sin(p)·sin(y)·cos(r) - sin(r)·cos(y)	-sin(p)·sin(r)·sin(y) - cos(r)·cos(y)	sin(y)·cos(p)
cos(p)·cos(r)	-sin(r)·cos(p)	-sin(p)

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



First, cylindrical joints 4,5,6 have the intersection of their axis of rotation at the point O_5, so we replace these three cylindrical joints with a spherical one that can have any orientation and has the wrist center at O_5. We also observe that the rotation of a 1 joint doesn't interfere with the rotations of joints 2 and 3. That's why knowing the wrist center we can easily calculate theta1=atan(wcy/wcx). Then we end up with a triangle where we know 3 sides and we are looking for its angles. We find them using cosine law, and afterward we find theta2 and theta3. We know the wrist center coordinates with respect to the second joint, so the third side is calculated as: $\sqrt{(WCx)^2 + (WCy)^2}$

side_b=sqrt(pow((sqrt(WC[0]*WC[0]+WC[1]*WC[1])-0.35),2)+pow((WC[2]-0.75),2))

As the gripper is only translated along z-axis, we find the wrict center position as follows:

$$ {}^{0}r_{WC/0} = {}^{0}r_{EE/0} - d \cdot {}^{0}_{6}\mathrm{R} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} - d \cdot {}^{0}_{6}\mathrm{R} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} $$

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

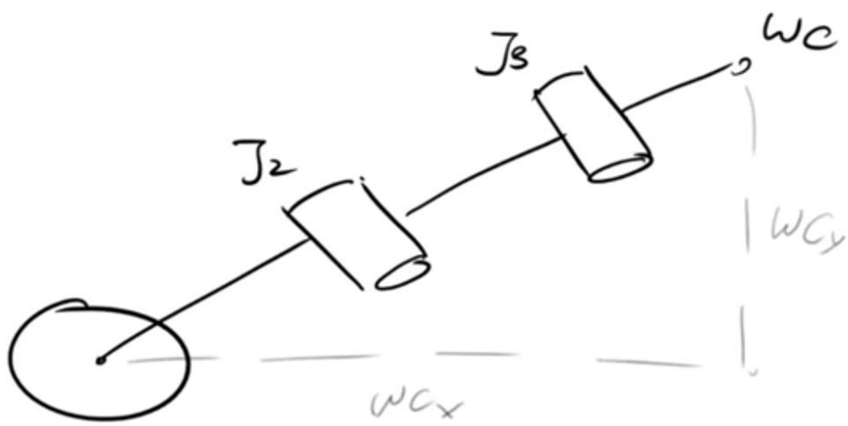


Figure 1(calculating theta1)

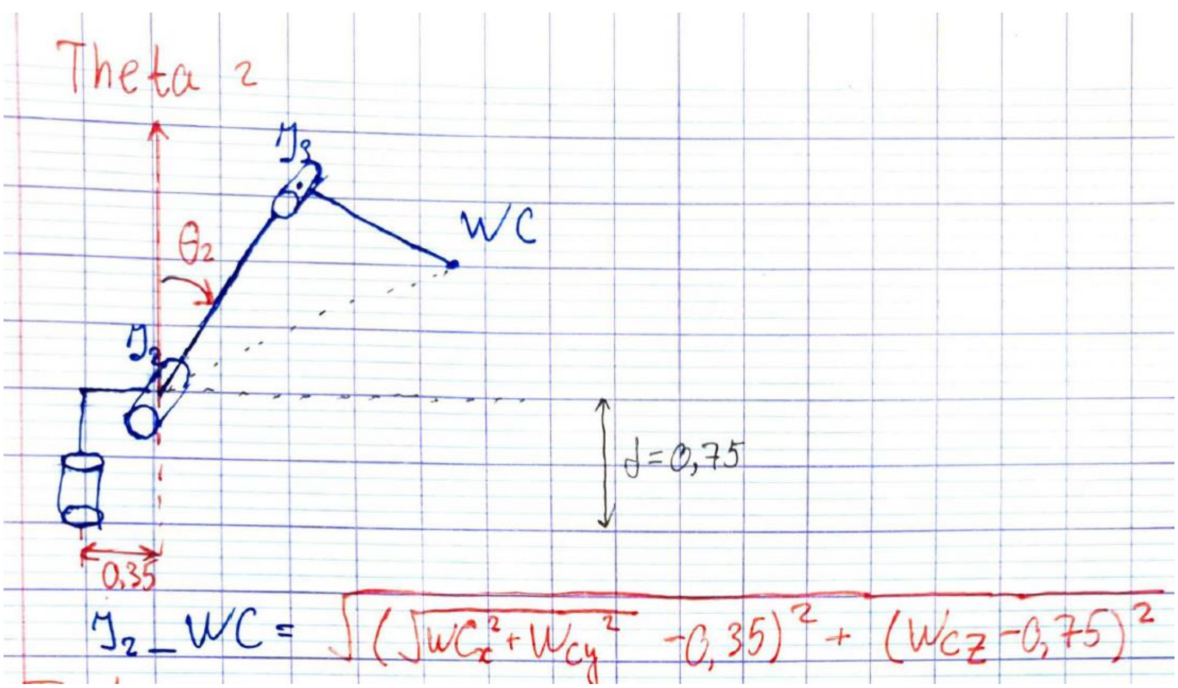


Figure 2(calculation du theta2)

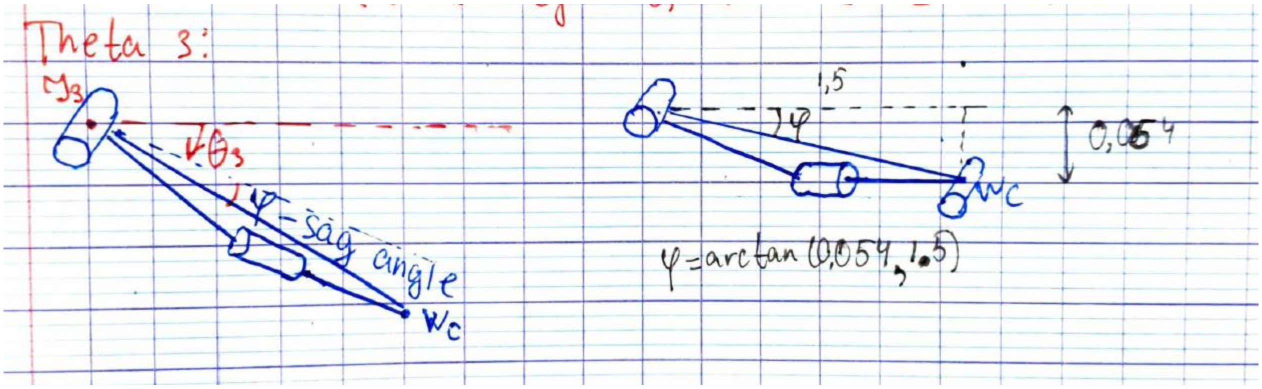


Figure 3(calculation of theta3)

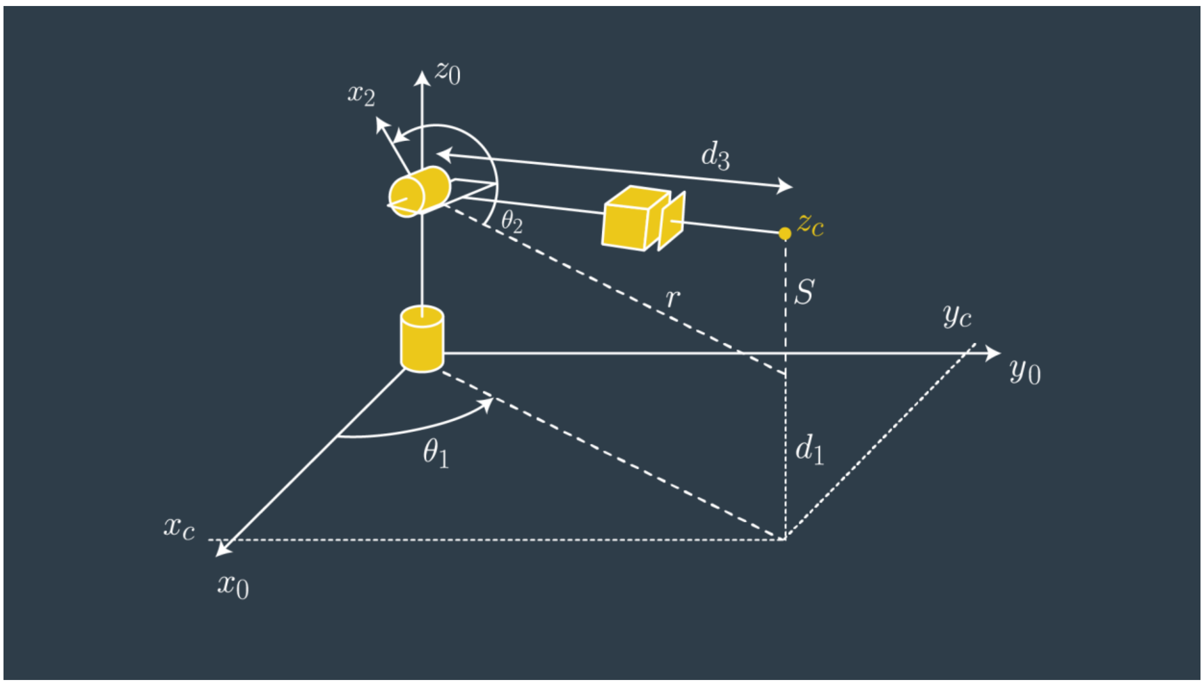


Figure 4(similar configuration for joint 1)

excerpt from the code:

```
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+a+arctan(y/x)=180^0, so theta2=180^0 -a+arctan(y/x)   //y,x are taken from figure 1

theta3 + (b + sag_angle) = 900, so theta3=pi/2-b-sag_angle, where sag_angle =arctan(-0,054/A)

ISRAILOV SARDOR

sag_angle ~= 0.036

After obtaining the rotation matrix from joint 3 to joint 6, by R3_6 = inv(R0_3) * Rrpy,

We obtain Euler angles(using tricks) from rotation matrix: R3_6

$-\sin(q_4)\cdot\sin(q_6) + \cos(q_4)\cdot\cos(q_5)\cdot\cos(q_6)$	$-\sin(q_4)\cdot\cos(q_6) - \sin(q_6)\cdot\cos(q_4)\cdot\cos(q_5)$	$-\sin(q_5)\cdot\cos(q_4)$
$\sin(q_5)\cdot\cos(q_6)$	$-\sin(q_5)\cdot\sin(q_6)$	$\cos(q_5)$
$-\sin(q_4)\cdot\cos(q_5)\cdot\cos(q_6) - \sin(q_6)\cdot\cos(q_4)$	$\sin(q_4)\cdot\sin(q_6)\cdot\cos(q_5) - \cos(q_4)\cdot\cos(q_6)$	$\sin(q_4)\cdot\sin(q_5)$

theta4=arctan(R3_6[2,2], -R3_6[0,2])

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

theta6= arctan(-R3_6[1,1],R3_6[1,0]).

For theta5 there are 2 possible solutions, because square root can take 2 possible values i.e

+/- arctan(sqrt(R3_6[0, 2]*R3_6[0,2]+R3_6[2,2]*R3_6[2,2]),R3_6[1,2]). I treat only + solution, that's why in the simulation, I have some unnecessary movements. Though, theta5 doesn't directly affect theta4 and theta6 since the axis of rotations are perpendicular (the projection is zero), in order for theta5 with the negative sign has the same result for theta5 with "+", we need to inverse both theta 4 and theta6(+ or – pi), that's why it's easier to take theta5 with the positive sign.

Project Implementation

After completing IK_debug.py and implementing it, I obtained the following results:

```
robond@udacity:~/catkin_ws/src/RoboND-Kinematics-Project$ python IK_debug.py

Total run time to calculate joint angles from pose is 1.6527 seconds

Wrist error for x position is: 0.00000046
Wrist error for y position is: 0.00000032
Wrist error for z position is: 0.00000545
Overall wrist offset is: 0.00000548 units

Theta 1 error is: 0.00093770
Theta 2 error is: 0.00181024
Theta 3 error is: 0.00205031
Theta 4 error is: 0.00172067
Theta 5 error is: 0.00197873
Theta 6 error is: 0.00251871

**These theta errors may not be a correct representation of your code, due to the fact
that the arm can have muliple positions. It is best to add your forward kinmeatics to
confirm whether your code is working or not**


End effector error for x position is: 0.00002010
End effector error for y position is: 0.00001531
End effector error for z position is: 0.00002660
Overall end effector offset is: 0.00003668 units

robond@udacity:~/catkin_ws/src/RoboND-Kinematics-Project$ █
```

In the IK_SERVER, the ROS node that we use to calculate angles of each joints and send them as a message:

I implement all transformation and rotation matrices outside of a for loop and evaluate for each case inside the for loop.

I declared symbols, DH table and all the transformation matrices; actually for our purposes we need only T0_3.

DESCRIPTION:

The objective is that we are given the position of and orientation (is extracted from an object req) of an end gripper and we need to return the 6 angles of a robotic arm joint. We initialize the dh-table, and symbolic representation. WE create the function of general transformation from kinematics section. Then, we create individual transformation matrices for T0_1, T1_2,T2_3. For debug script we do the transformation matrices till the end gripper, to check our result with the forward kinematics, but for

IK_server we only need the first three rows and columns of each matrix till the wrist center. FromWE have the DH-table that we use to construct o

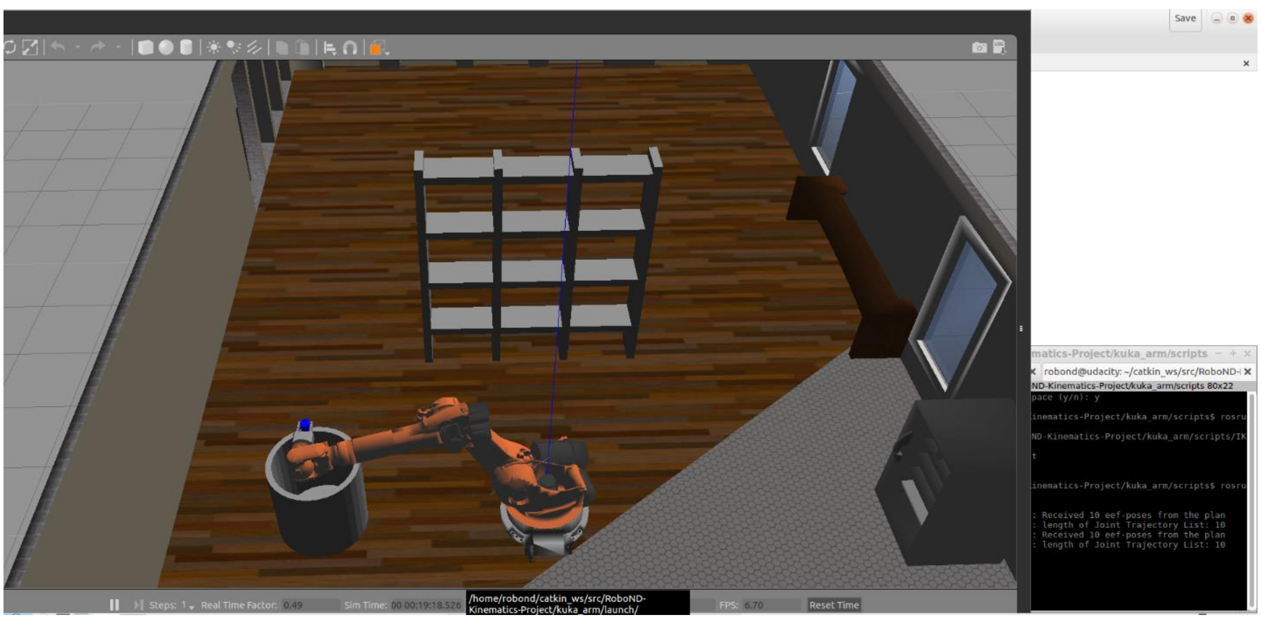
There is a rotation error from transforming URDF coordinates to DH parameters; and we correct it with rotating the end gripper matrix by pi along the z-axis and -pi/2 along the y-axis. The rotation of the end gripper ROT_EE we obtain from the angles extracted from the request using the rotation matrices that we defined for x, y and z. The wrist center position is displaced from the end effector position by z=0.303.

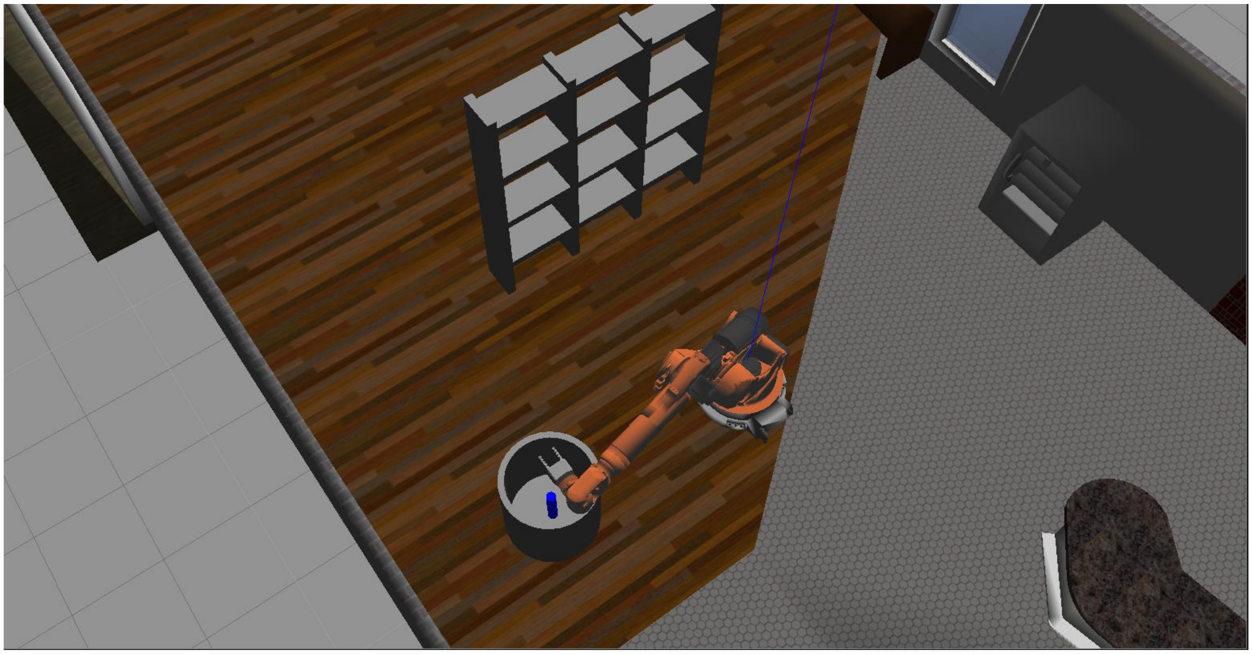
When we find the wrist center, we easily find theta1,2,3 using cosine law(acos). Afterwards, we use transpose, and find R3_6. We find theta 4,5,6; but we can improve them since we don't want our robot perform to much additional movement. We use atan2 function to find the angle (-pi,pi) and we force the joints to go to pi+alpha, if it's located between(pi/2,pi), because it's less movement then if it goes through zero angle. The same procedure applies for negative angles. Fortunately, min, maxangles of joints 4,6 are large enough.

ANNEXE:

Launch the project:

```
robond@udacity:~$ cd ~/catkin_ws/src/RoboND-Kinematics-Project/kuka_arm/scripts
```





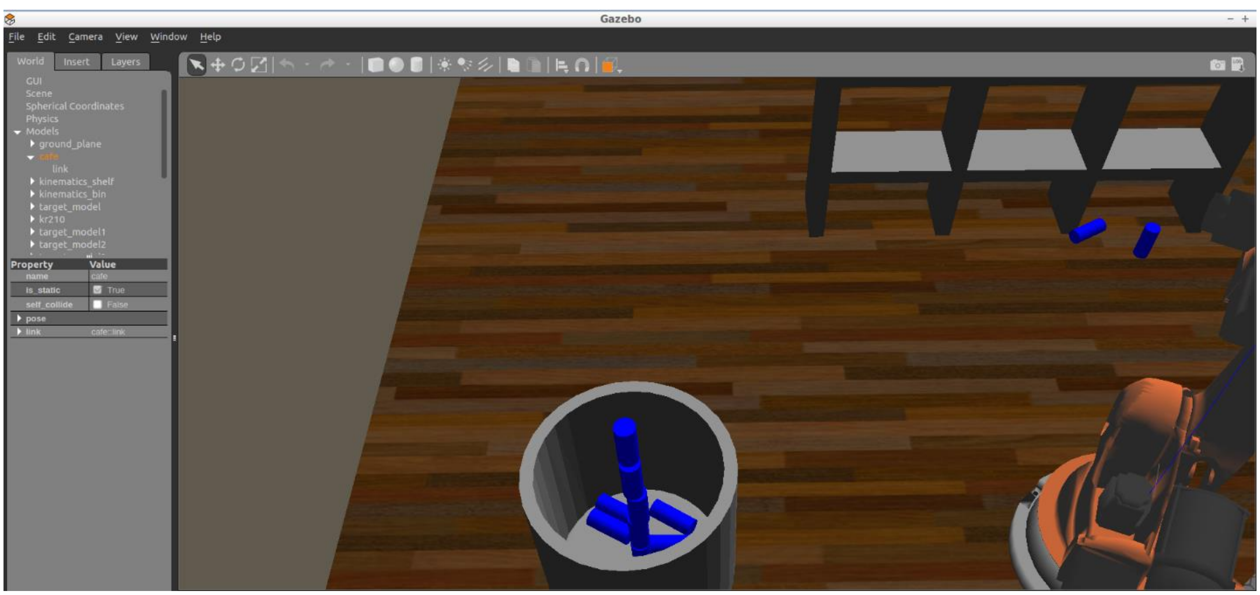


Figure 5(complete pick&place)

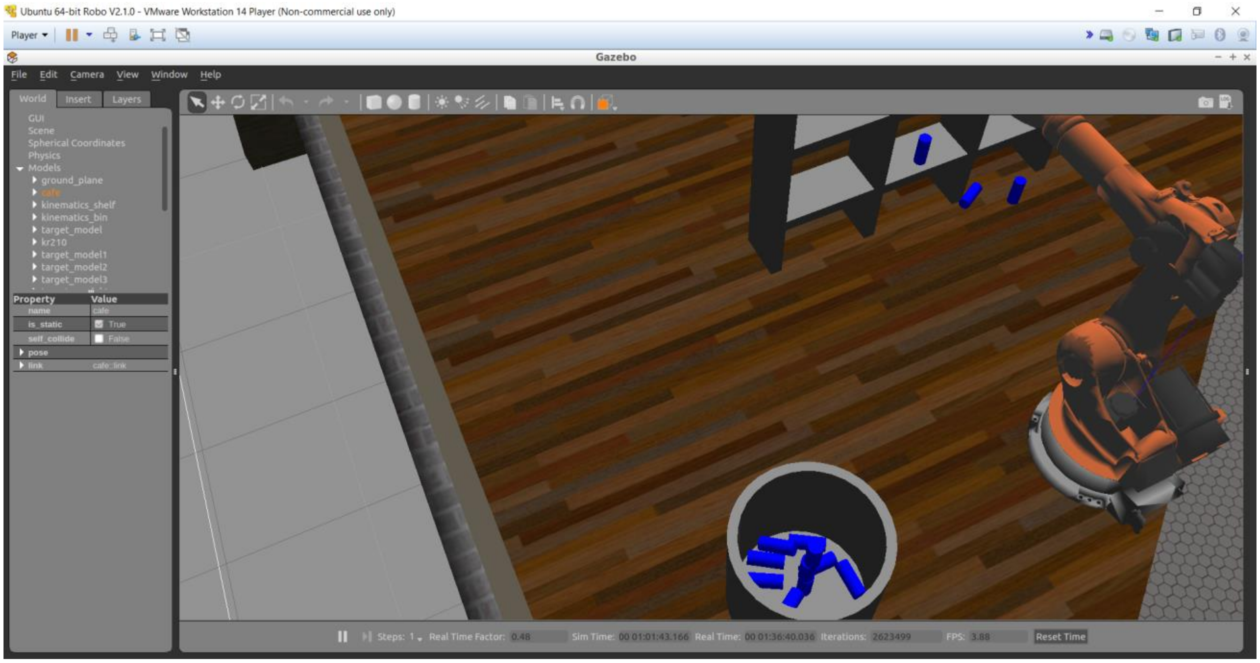
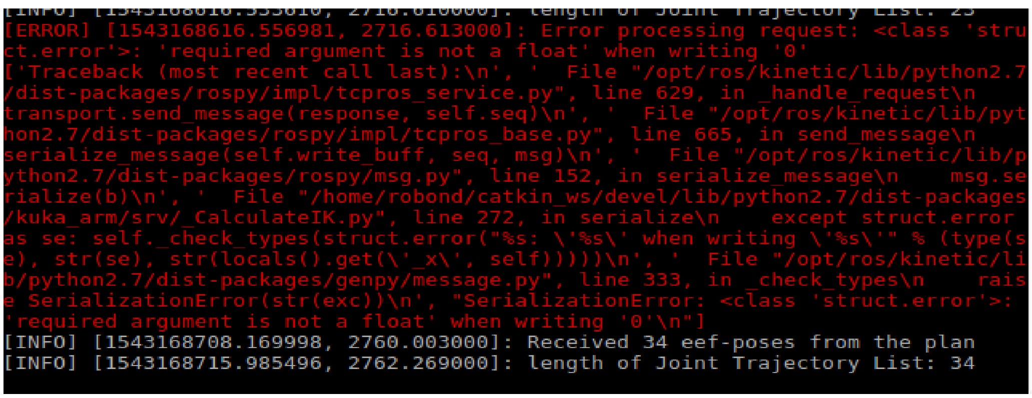


Figure 6(10/12)



Conclusion: Kinematics analysis was done using URDF file (joints section) as well as kinematics part of this project. There are multiple solutions, which means that in order for the movement to be efficient, we need to choose between several solutions depending on each case.

I was able to reach 9/10 and 10/12 result though I got an error of calculation for requests that contained many poses. The computation was quite slow, and sometimes there is a lot of unnecessary movements particularly for joint 5. Though the method is quite consistent, there are still a lot to be improved (transition to numpy, implementation of a class and a lambify functions).