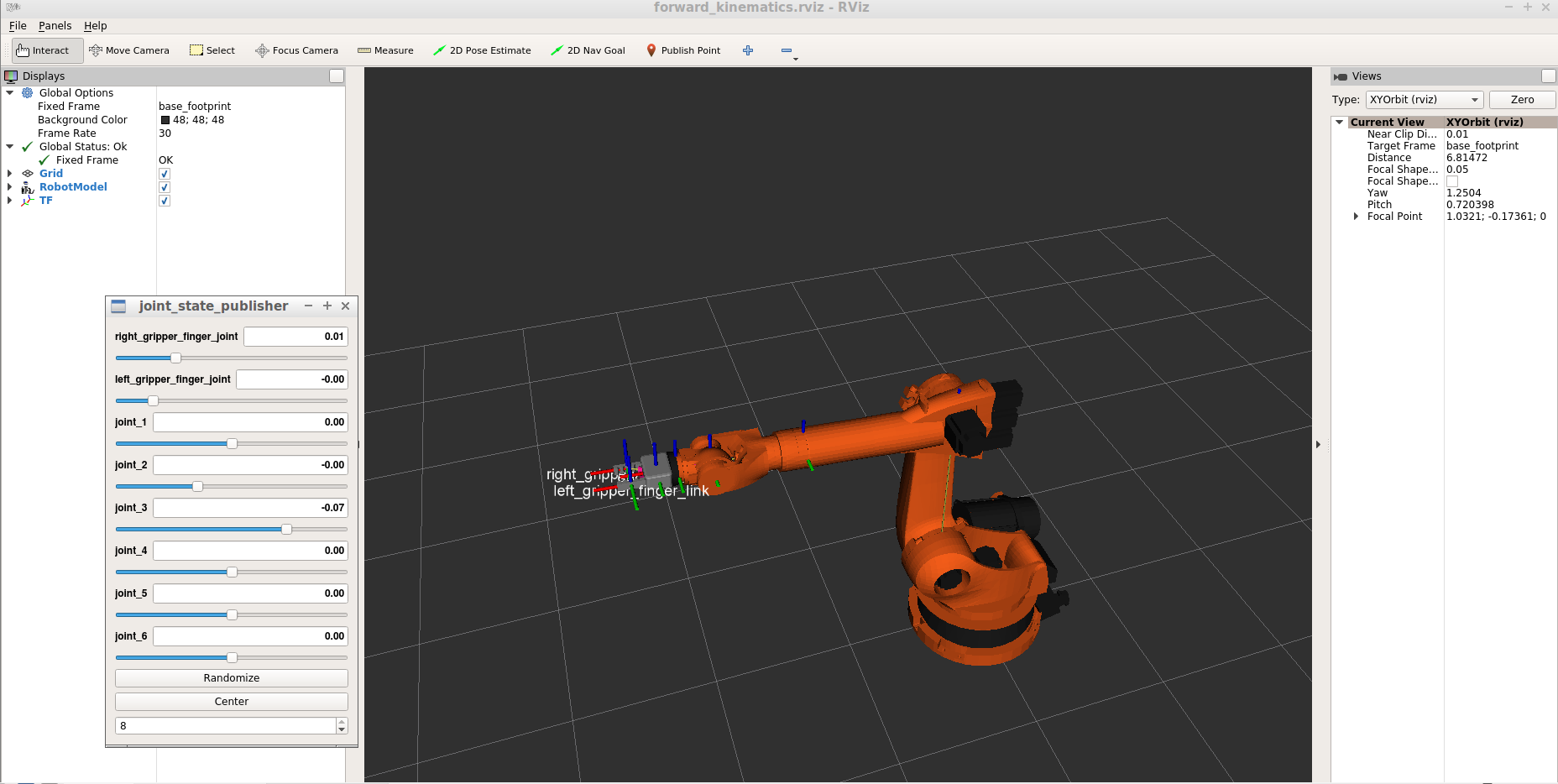
**README (Pick and Place)**

**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).

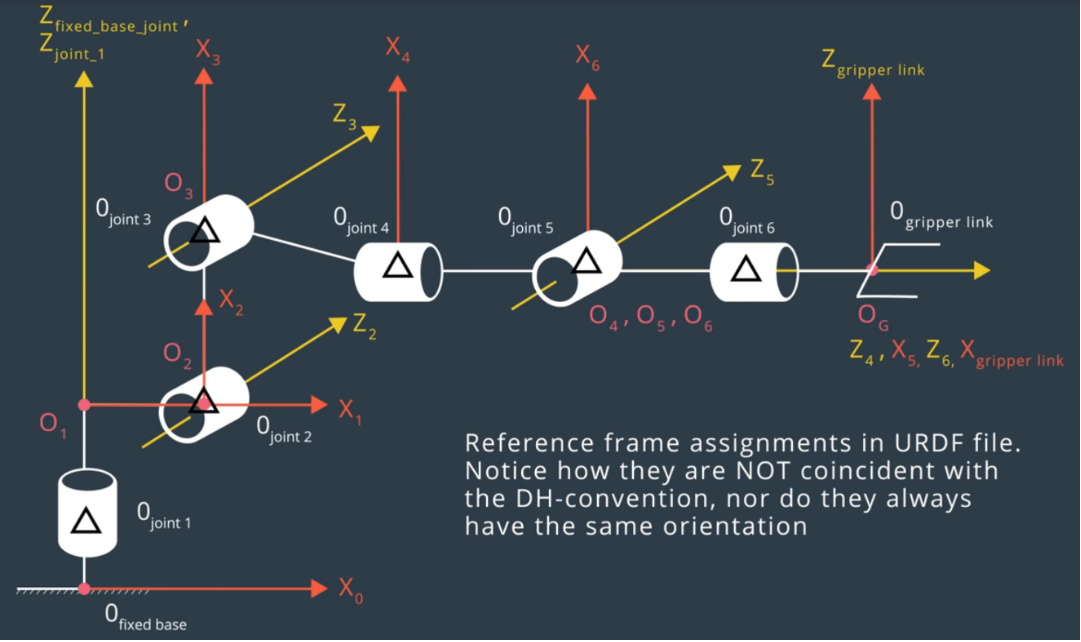
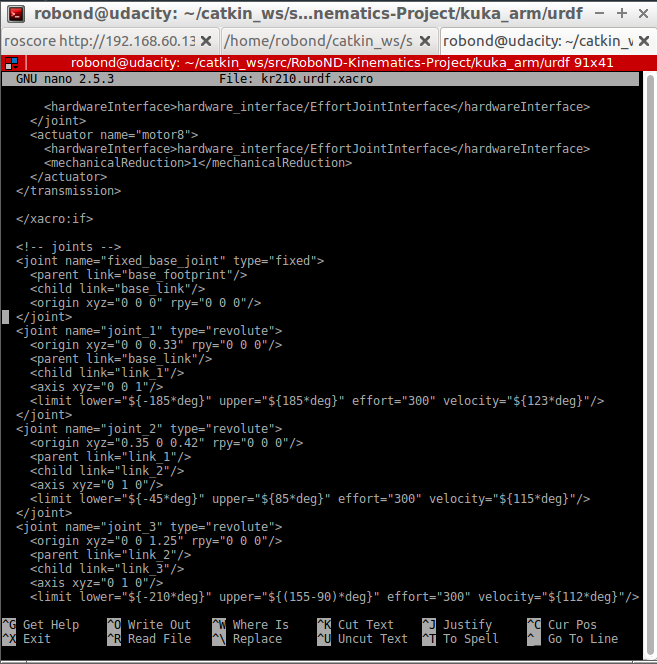
The first observations show that the robot lies essentially in the x,z plane.

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 of 0.33+0.42=0.75, which is the projection from the distance from O0 to O1. From O3 to O4, we have the offset of d=0.96+0.54=1.5. Note that O4 O5 O6 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 oder to align the frame EE and the bas\_frame, we need to rotate the gripper intrinsically around z-axis about 180degrees and y axis by the angle of –pi/2.

| **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 |

**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.**

| cos(thetai) | -sin(thetai) | 0 | ai-1 |
| --- | --- | --- | --- |
| sin(thetai) \* cos(alphai-1) | cos(thetai) \* cos(alphai-1) | -sin(alphai-1) | -sin(alphai-1) \* di |
| sin(thetai) \* sin(alphai-1) | cos(thetai) \* sin(alphai-1) | cos(alphai-1) | cos(alphai-1) \* di |
| 0 | 0 | 0 | 1 |

The general transform matrix is:

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(alphai-1)=0**)

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

**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 |

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 |

**Joint5–> Joint6**

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

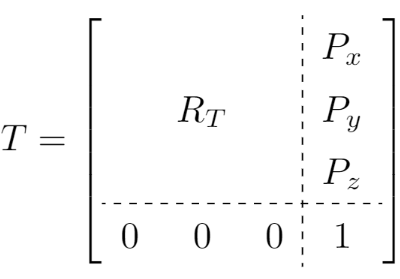
**Joint6–> end Gripper Link**

| 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 :**

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) |

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) |

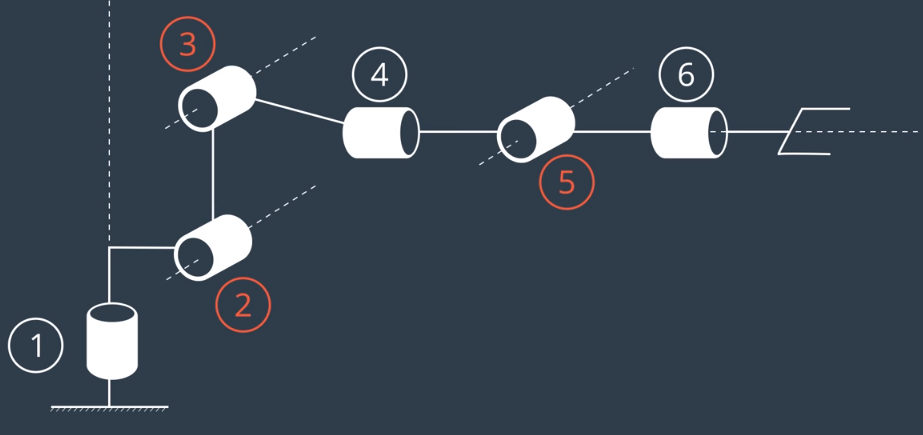
**There is also rotation error due to the fact that the the frames in URDF file doesn’t coincident with DH-convention. We apply intrisinsic rotations along z axis 180 degrees and the y axis by angle of -90 degrees. The error matrix of end gripper (Rot\_z(180)\*Rot\_y(-90)):**

| **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 O5, so we replace these three cylindrical joints with a spherical one that can have any orientation and has the wrist center at O5. 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:

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

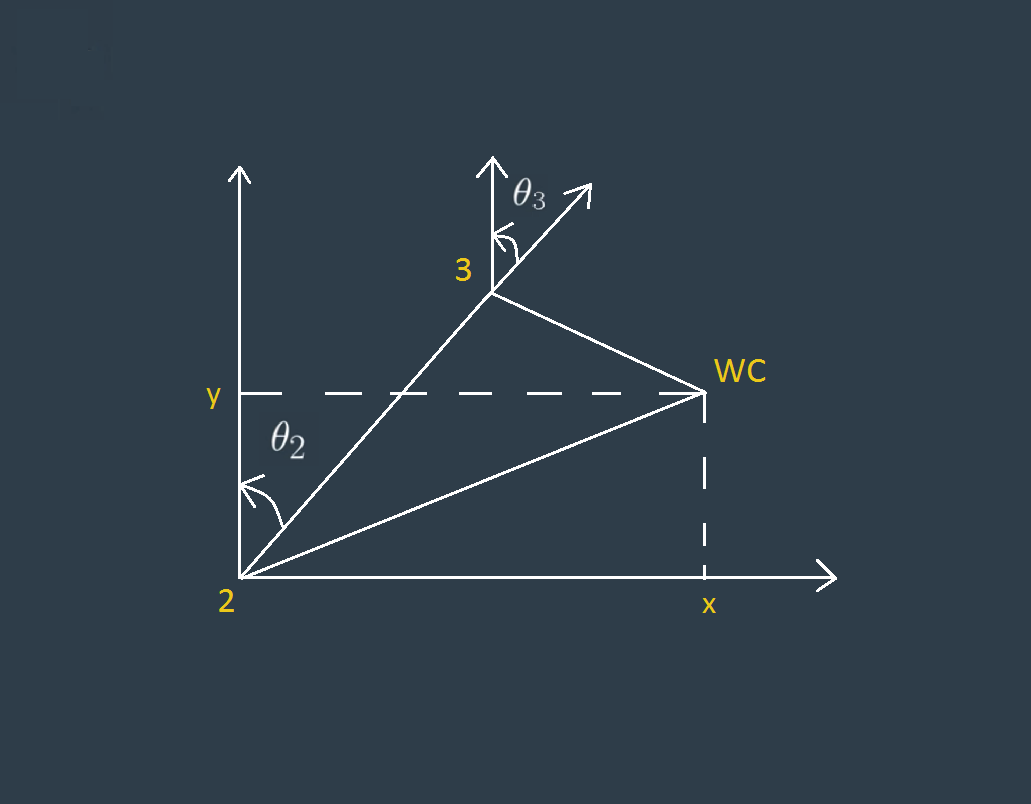
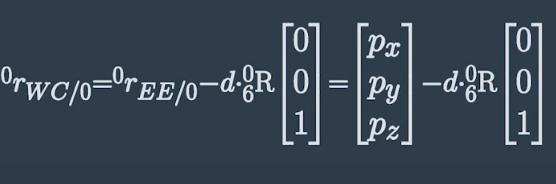
[](https://github.com/udacity/RoboND-Kinematics-Project/blob/master/misc_images/misc3.png)

Figure 1(calculating the theta1,2,3 from the wrist center)

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



Code assosiated: WC=EE-(0.303)\*ROT\_EE[:,2]

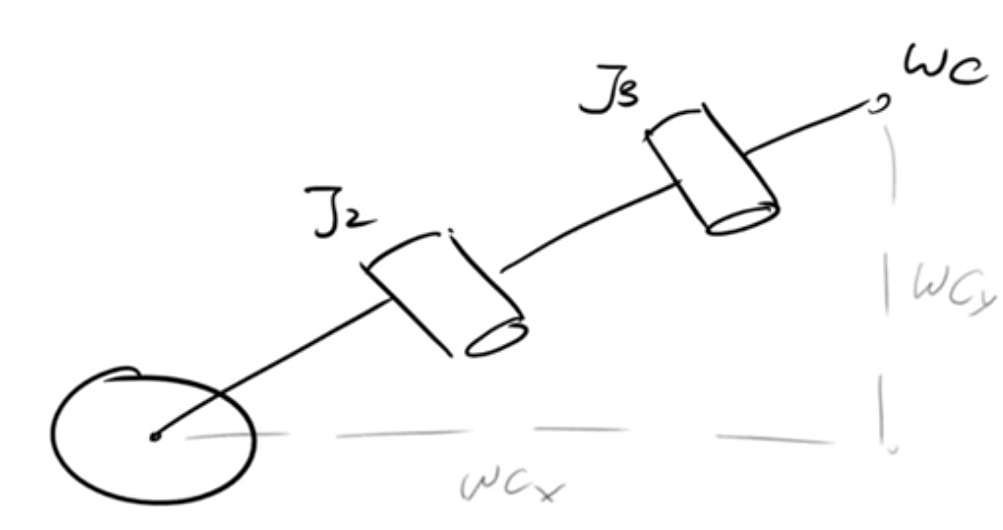


Figure 2(calculating theta1)

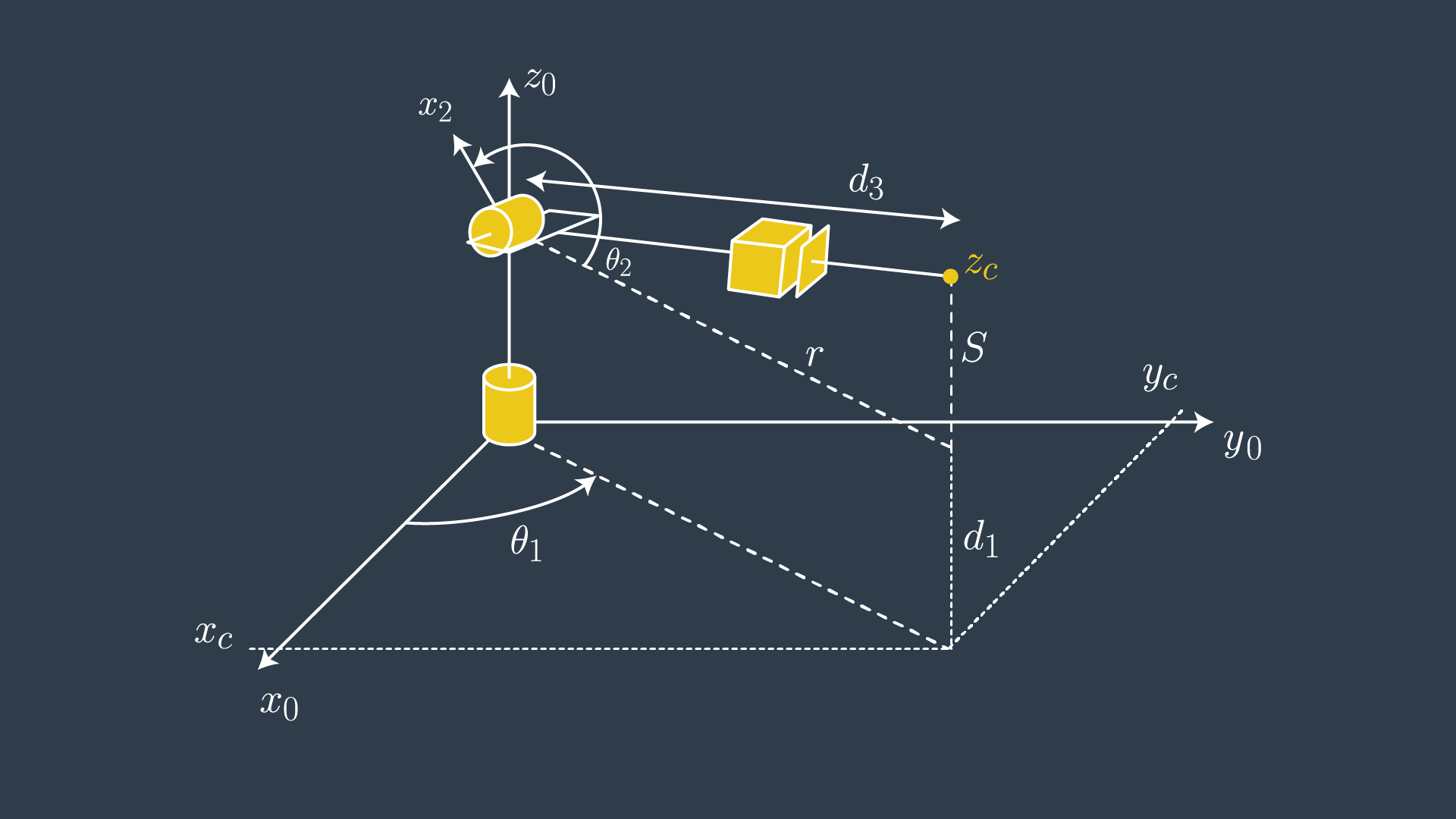
excerpt from the code:

Figure 3(similar configuration for joint 1)

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)=1800, so theta2=1800 -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)

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(q₄)⋅sin(q₆) + cos(q₄)⋅cos(q₅)⋅cos(q₆)** | **-sin(q₄)⋅cos(q₆) - sin(q₆)⋅cos(q₄)⋅cos(q₅)** | **-sin(q₅)⋅cos(q₄)** |
| --- | --- | --- |
| sin(q₅)⋅cos(q₆) | -sin(q₅)⋅sin(q₆) | cos(q₅) |
| -sin(q₄)⋅cos(q₅)⋅cos(q₆) - sin(q₆)⋅cos(q₄) | sin(q₄)⋅sin(q₆)⋅cos(q₅) - cos(q₄)⋅cos(q₆) | sin(q₄)⋅sin(q₅) |

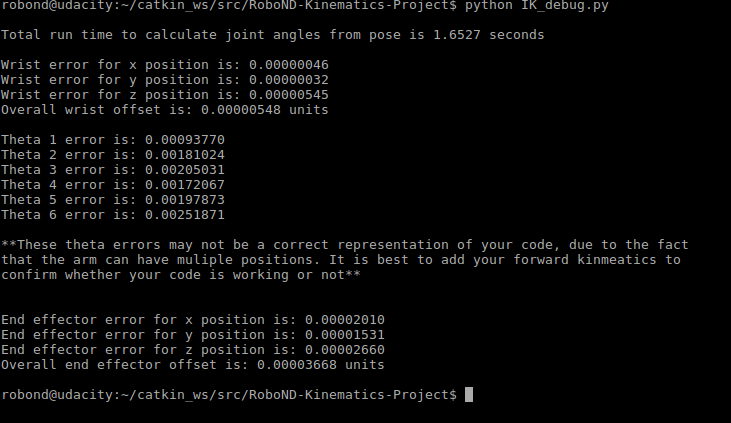
theta4=arctan(R3\_6[2,2], -R3\_6[0,2])

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]).

**Project Implementation**

**After completing IK\_debug.py and implementing it, I obtained the following results:**



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.

And just for fun, another example image:

Errors: [ERROR] [1541287735.629370028, 1136.091000000]: Found empty JointState message

[ERROR] [1541287692.341225473, 1111.719000000]: Failed to call service calculate\_ik

ANNEXE:

Launch the project: 