**Project: Kinematics Pick & Place**

**Steps to complete the project:**

1. Set up my ROS Workspace.
2. Download or clone the project repository into the ***src*** directory of my ROS Workspace.
3. Experiment with the forward\_kinematics environment and get familiar with the robot.
4. Launch in demo mode.
5. Perform Kinematic Analysis for the robot following the project rubric.
6. Fill in the IK\_server.py with my Inverse Kinematics code.

**Rubric Points**

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

**Kinematic Analysis**

**Obtaining D-H parameters**

ROS Visualization tool was used to view the 3D model of the Kuka KR210 to interact with the robot model and view it at the home position from multiple views to better understand it.

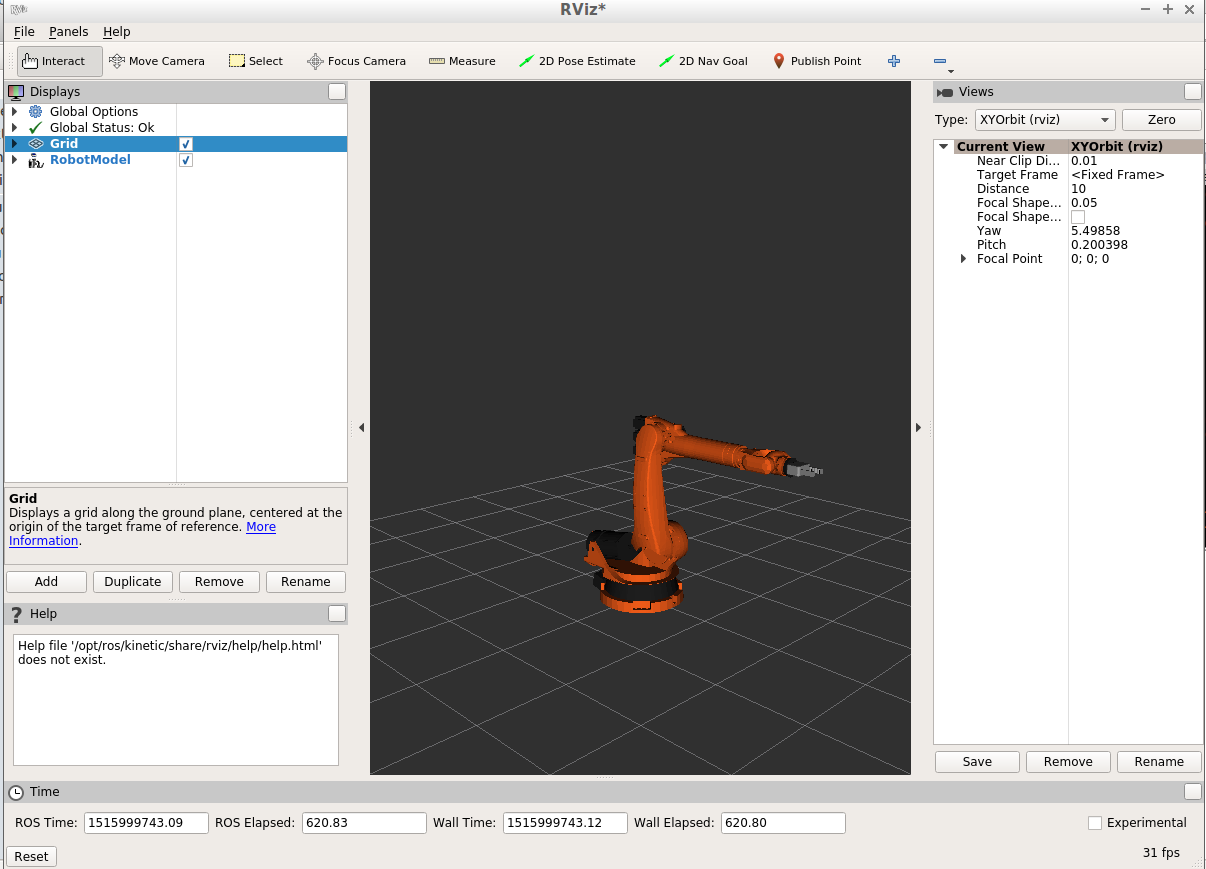


Figure . Kuka KR210 3D model in RViz.

To better understand the movement of the robotic arm and understand each axis’s type, a special tool was used to interact with the robot model with interactive markers.

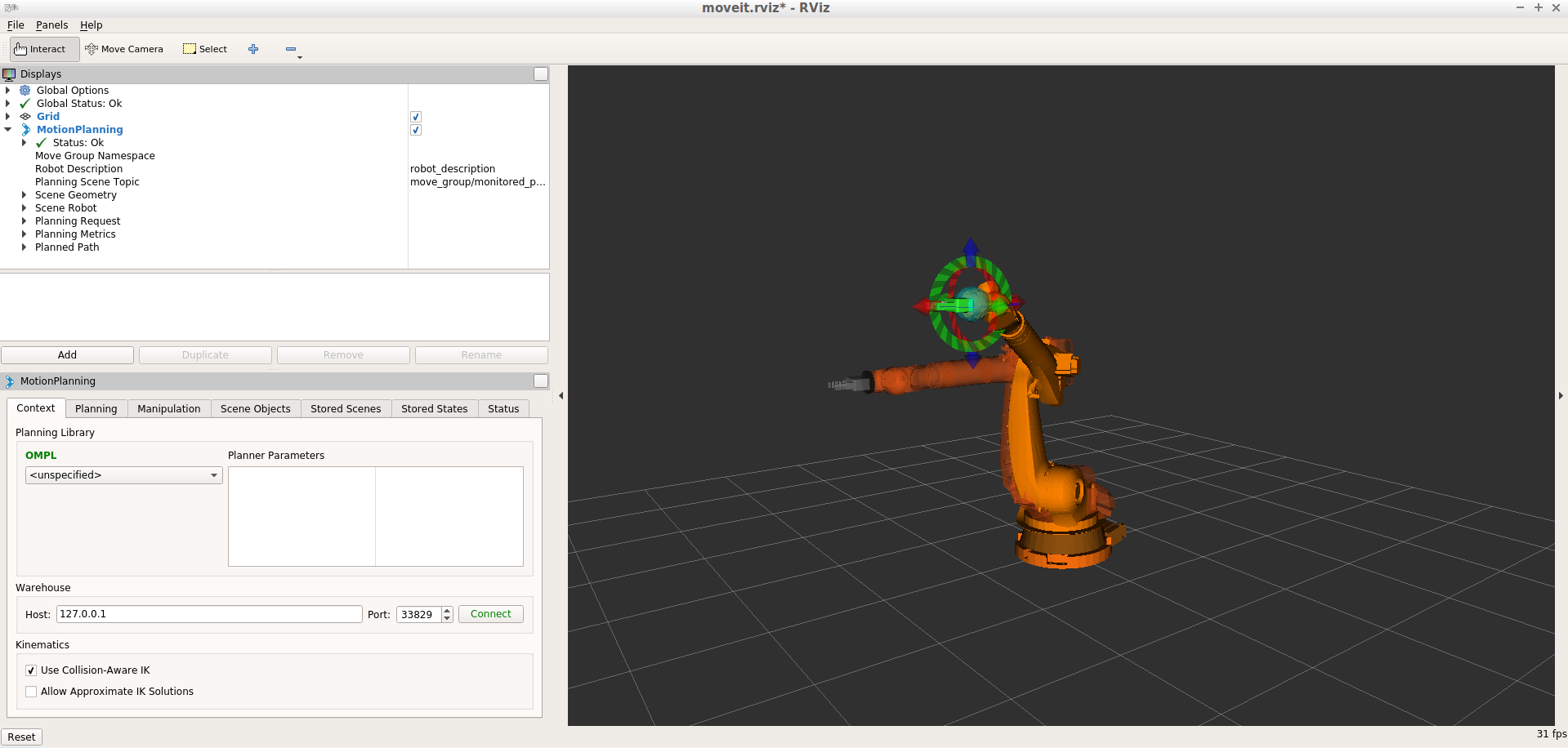


Figure . Using interactive markers for Kuka KR210 in RViz.

After understanding that Kuka kr210 robot has six revolute joints and seeing how the links and joints are made up when the robot is at home position were all joint angles are equal to zero, then the kinematic model was easily obtained following the Denavit-Hartenberg Convention.

Denavit-Hartenberg Convention is a convention used to attach a coordinate system to each link of a manipulator. The coordinate systems are attached according to the following four rules:

1. Z-axis is in the direction of the joint axis.
2. X-axis is perpendicular to both Zn and Zn-1.
3. Y-axis follows the right-hand rule.
4. The X-axis must intersect the Zn-1 axis.

The coordinate systems of the ground and end effector links do not follow these rules. The coordinate system of the ground link can be chosen based on convenience as long as the Z0 -axis is aligned with the axis of joint 1. The coordinate system of the end effector can also be chosen based on convenience as long as its X-axis is normal to the last joint axis.

The D-H parameters kinematically describe any serial manipulator by specifying four parameters for each link, which are (Spong, 2012):

* Link length : the distance from to measured along .
* Link twist : the angle from to measured about .
* Link offset : the distance from to measured along .
* Joint angle : the angle from to measured about .

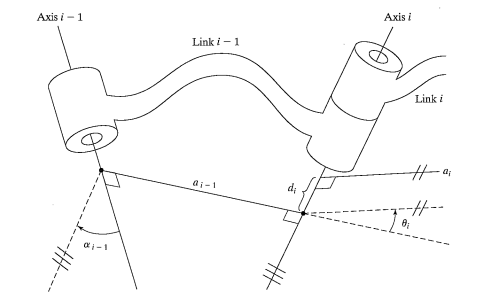


Figure . Denavit-Hartenberg Convention.

To get from frame i-1 to frame I, the following steps must be followed (Cao, 2015):

1. Translate along by .
2. Rotate about current Z-axis of angle.
3. Translate along current X-axis by .
4. Rotate about current X-axis of angle.

If the joint is a revaluate then the Joint angle will be the only variable parameter and if the joint is a prismatic joint, the Link offset will be the variable parameter.

The following graph shows the Link-frame assignments that was done for the robotic arm type that is used in the project.

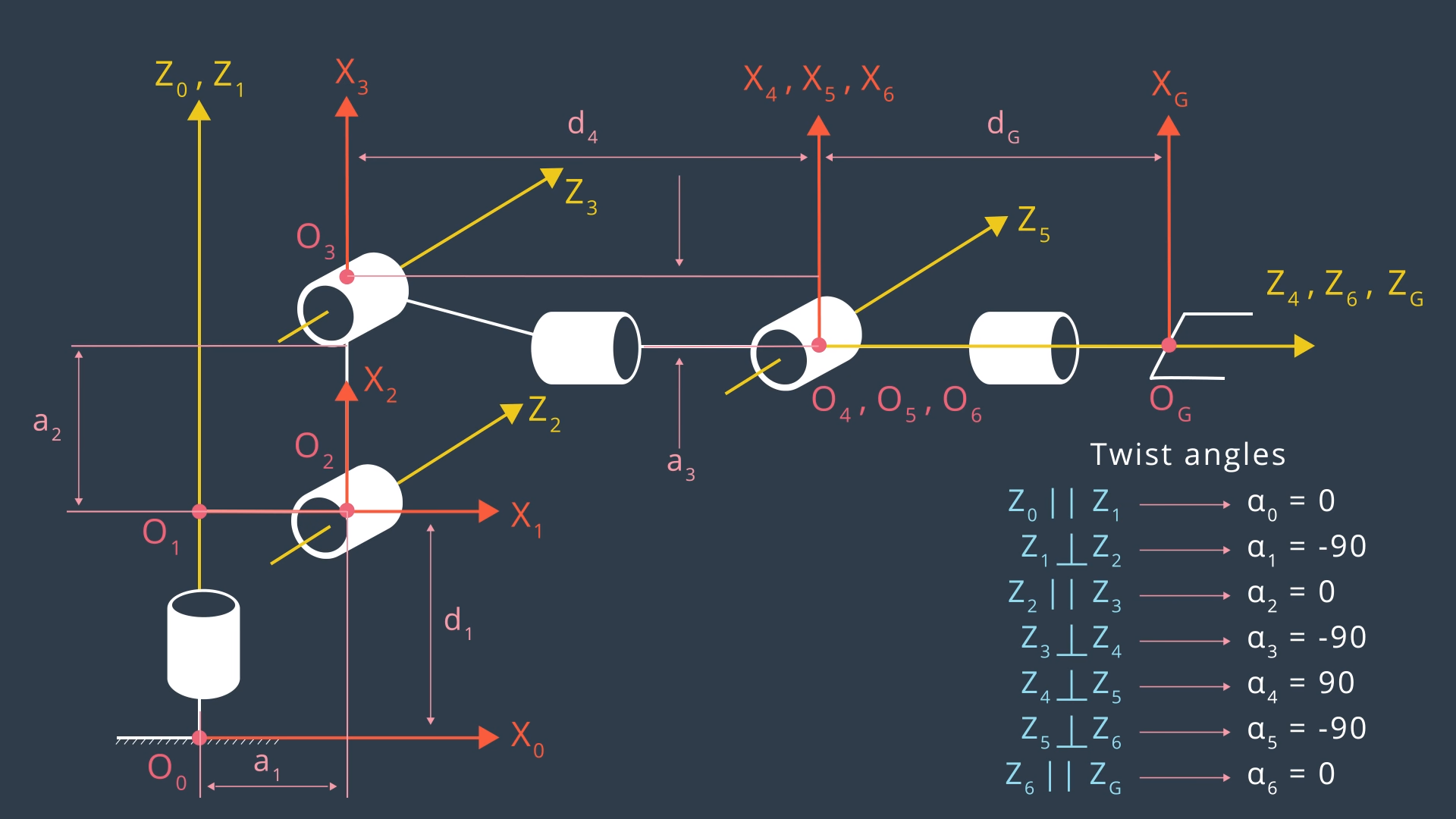


Figure . Kinematic Model of the robotic arm showing the D-H parameters.

The URDF file was used to obtain numerical values of and . Unified Robot Description Format or urdf, is an XML format used in ROS for representing a robot model. URDF file can be used to define a robot model, its kinodynamic properties, visual elements and even model sensors for the robot.

The following figure shows the reference frame assignments in the URDF file represented by the black triangles as the origin of these frames. The figure shows the relative location of joint {i-1} to joint {i} as described in the kr210.urdf.xacro file. These values were used to obtain the to obtain numerical values of and since the kinematic DH frame assignment defers from the reference frame assignments in the URDF file.

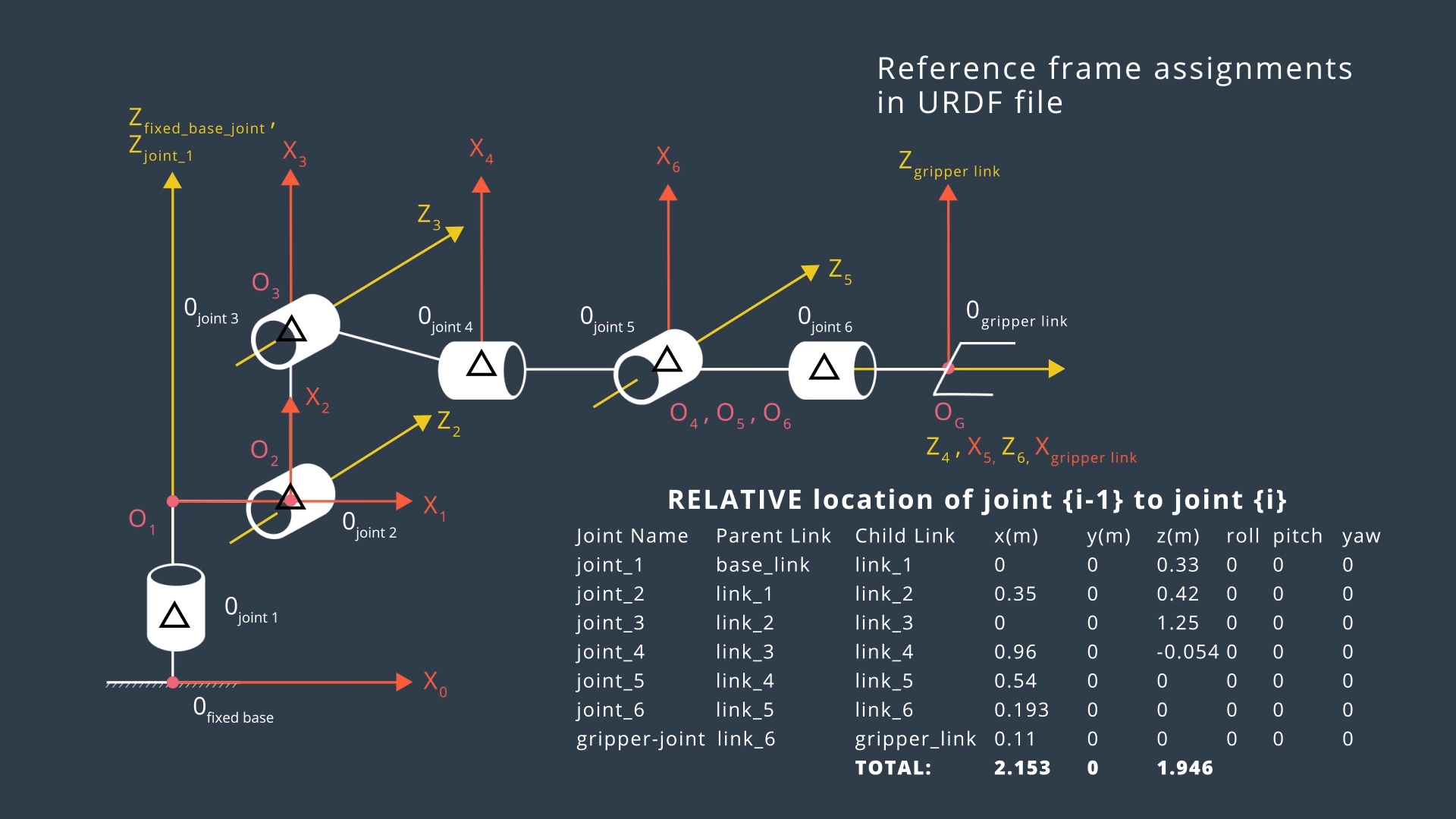


Figure . Reference frame assignments in URDF file.

This table represents the link parameters for the Kuka KR210.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | -1 | -1 |  |  |
| 1 | 0 | 0 | 0.75 |  |
| 2 | -90 | 0.35 | 0 | -90 |
| 3 | 0 | 1.25 | 0 |  |
| 4 | -90 | -0.054 | 1.5 |  |
| 5 | 90 | 0 | 0 |  |
| 6 | -90 | 0 | 0 |  |
| 7 | 0 | 0 | 0.303 |  |

Table . D-H parameters for the serial manipulator.

**Finding the Transforms**

Using the DH convention and parameters, transformation matrices relating two successive coordinate systems can be established. The coordinate system can be obtained through successive rotations and translations of the coordinate system as follows:

1. Translating the coordinate system along the axis by distance establishes the following transformation matrix:
2. Rotating the displaced coordinate system about the -axis by an angle establishes the following transformation matrix:
3. Translating the displaced coordinate system along the -axis by a distance establishes the following transformation matrix:
4. Rotating the displaced coordinate system about the -axis by an angle establishes the following transformation matrix:

The resulting transformation matrix that transforms the coordinate system to the coordinate system is

which can be expanded to

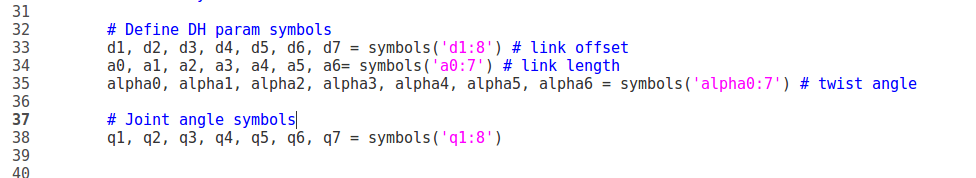
Therefore, once the link frames have been defined and the corresponding link parameters found, developing the kinematic equations is straightforward. From the values of the link parameters, the individual link-transformation matrices can be computed. Then, the link transformations can be multiplied together to find the single transformation that relates frame [N} to frame {0}:

This transformation, will be a function of all n joint variables. The Cartesian position and orientation of the last link can be computed by to find the robot’s joint-positions.

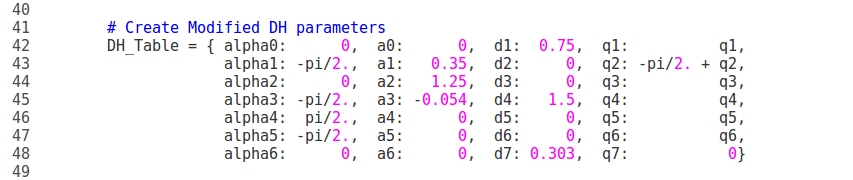
This means, for a given position and orientation of the tool frame, values for the joint variables can be calculated via the inverse kinematics. The analytical equations of the inverse kinematics can be derived from the transformation matrices (Spong, 2012).

**Generating D-H Homogeneous Transformation Matrices using software**

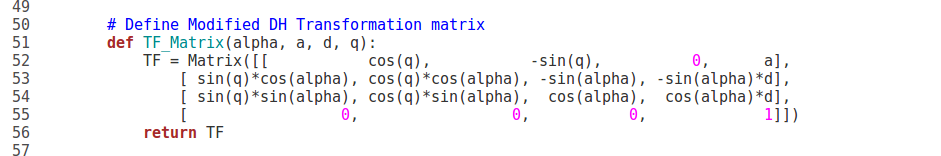
First the DH parameter symbols were defined in the IK\_server.py script as the following:

****

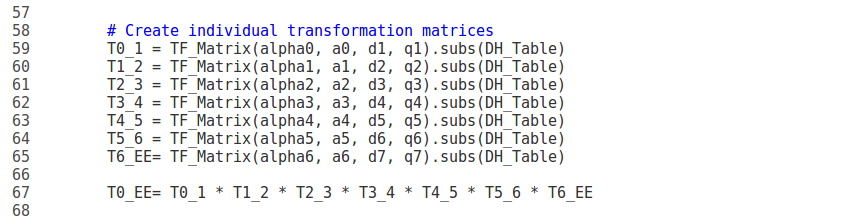
Then, the DH parameters table was created in a dictionary format, the values were obtained from table 1.



Then a DH Transformation matrix function was defined.



This function was used to create individual transformation matrices about each joint as the following:



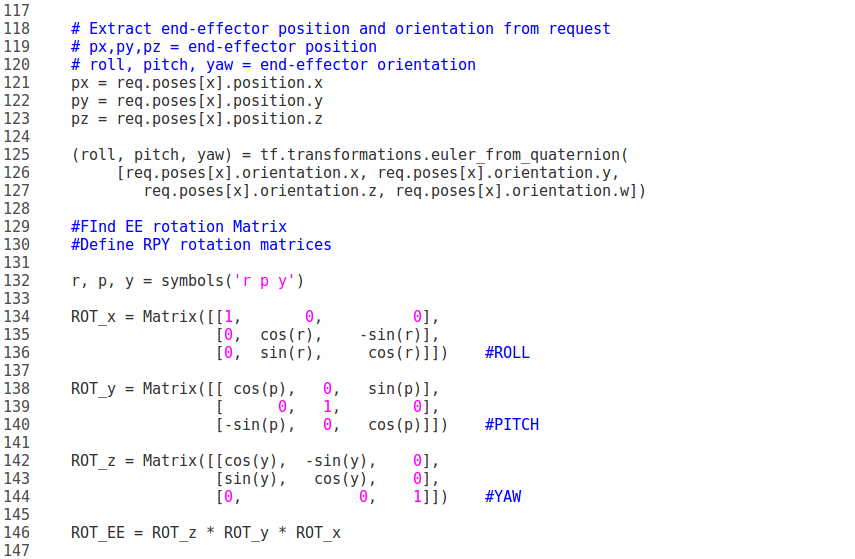
The output of each individual transform matrix is:

Finally, the transformation matrix from the base frame to the end effector was creating as the following line on code shows.

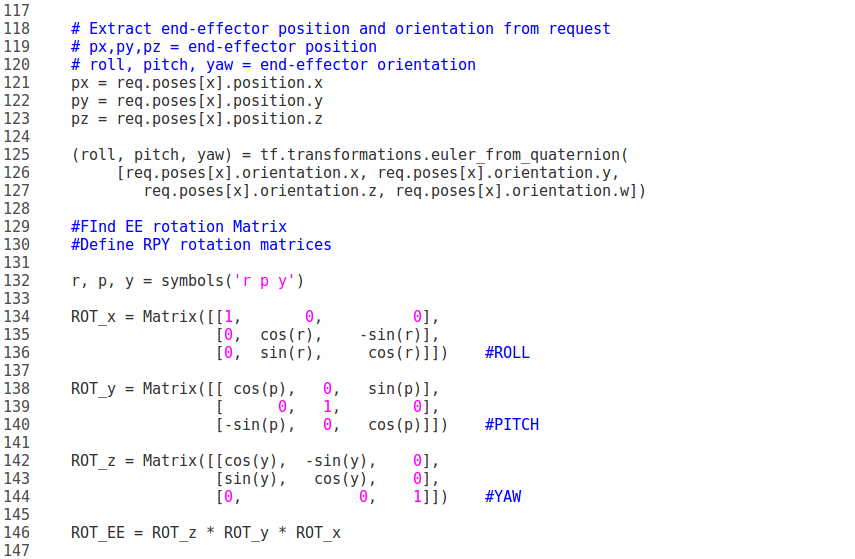
Screen Clipping

To generate a generalized homogeneous transform between base\_link and gripper\_link using only end-effector(gripper) pose these steps are followed:

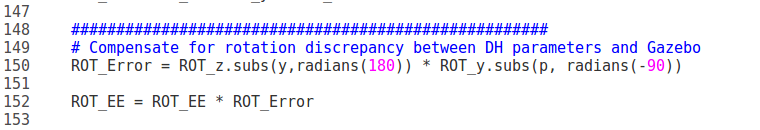
1. Extract the end effector position and orientation from request



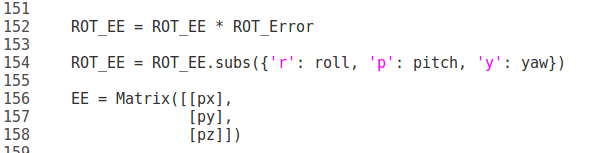
1. Find end effector rotation matrix by first defining the roll, pitch and yaw rotation matrices.



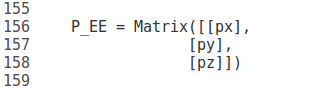
1. Compensate for rotation discrepancy between DH parameter and Gazebo and define the end effector rotation matrix



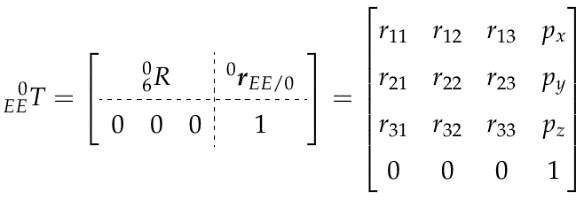
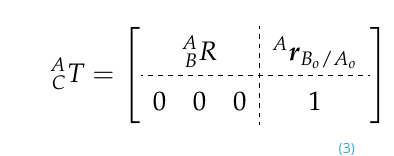
1. Substitute the end effector roll, pitch and yaw rotation values in the previously defined end effector rotation matrix.



1. Define the end effector translation matrix as the following:



1. To combine both the rotation and translation matrices in one matrix as the following illustration shows the bellow code is used.



Scaling

Position vector

r

Rotation matrix

r

T0\_EE= ROT\_EE.row\_join(P\_EE)

T0\_EE = T0\_EE.col\_join(Matrix([[0, 0, 0, 1]]))

**Finding the Inverse Kinematics**

Since the Kuka KR210 robot is considered a spherical wrist robot arm due to it having the axes of their hand joints intersecting at one common point. Where, these joints must be of the revolute type. The robot arm basic structure which is composed of the 1st three joints, can have either revolute or prismatic joints

The common point of intersection of a spherical wrist robot arm is called the wrist center. The advantage of such a design is that it kinematically decouples the position and orientation of the end effector. Where, a six degree of freedom serial manipulator with a spherical wrist would use the first three joints to control the position of the wrist center while the last three joints would orient the end effector as needed.

**Finding the wrist center**

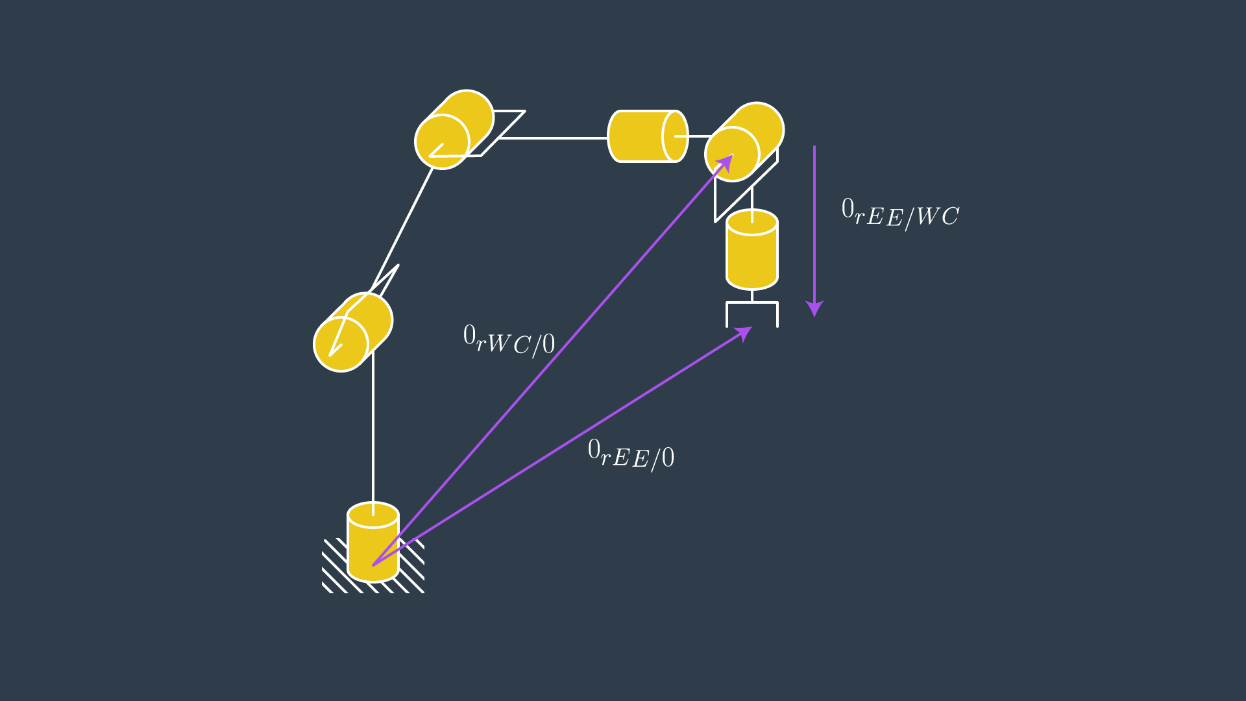


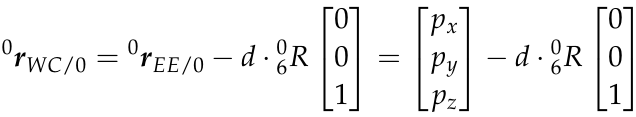
Figure . Wrist center (WC) and end effector (EE) relative to the base frame "0".

Looking at the above figure it is seen that the location of the wrist center (WC) and end effector (EE) relative to the base frame "0" is given by, *0rWC/0* and *0rEE/0* , respectively. The location of the EE relative to the WC is given by, ​*0rEE/WC*​ .

In order to find the wrist center the following steps were followed:

1. place the origin of frames 4, 5, and 6 coincident with the WC.
2. find the location of the WC relative to the base frame.

z4 was chosen parallel to z6 and pointing from the WC to the EE, so that the displacement would be a simple translation along z6. The magnitude of this displacement (d) depends on the dimensions of the manipulator and are defined in the URDF file. See figure below of the urdf file frame assignment where the x translations between link 5 to gripper link define the value of d as (0.193 +0.11 = 0.303). Further, since r13, r23, and r33 define the Z-axis of the EE relative to the base frame, then the Cartesian coordinates of the WC is found by:



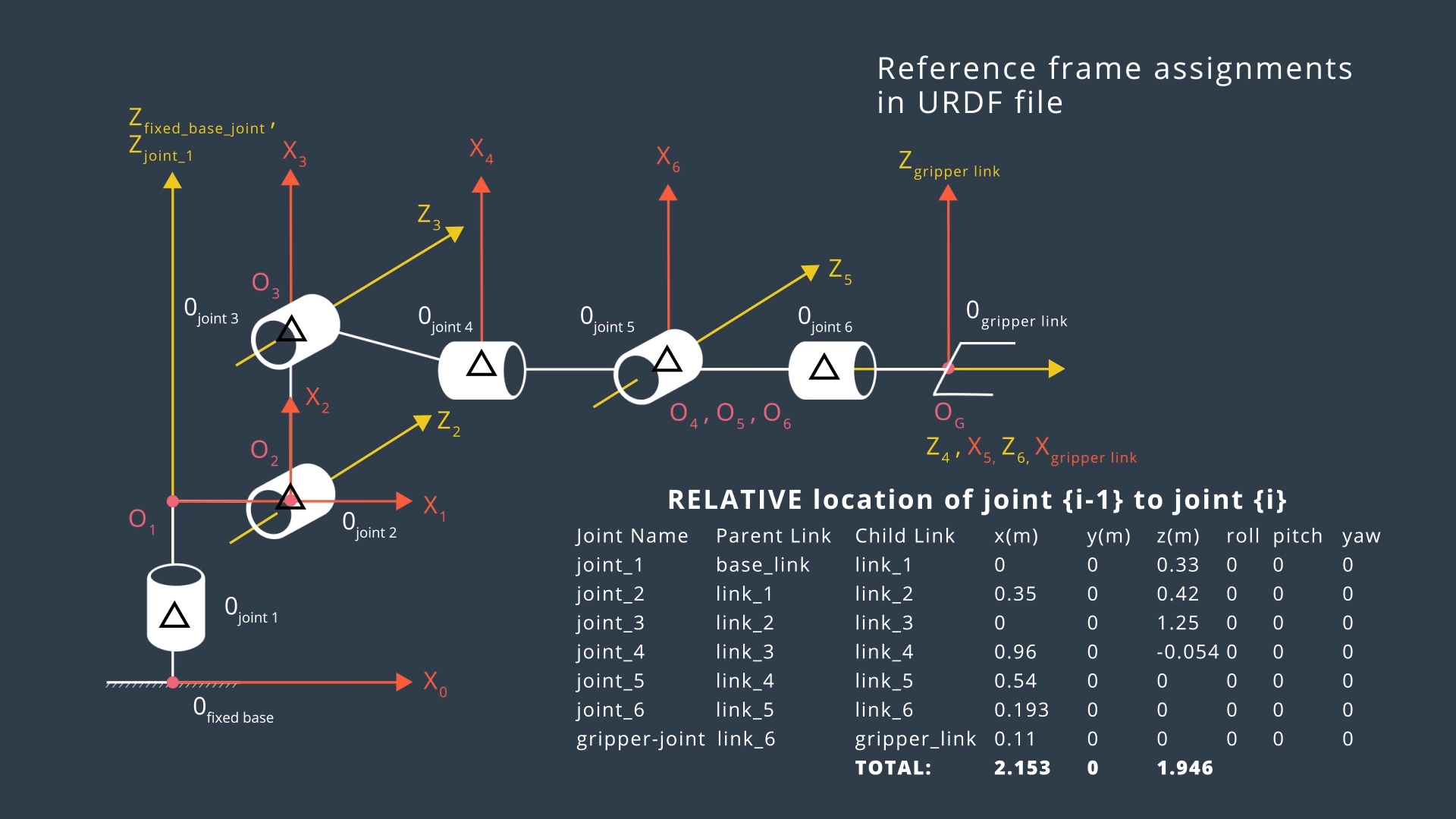


Figure . Dimensions of the manipulator as shown in the URDF file.

**Finding first 3 angles using the wrist center**

By finding the wrist center 𝜃1 is easily found by considering zc to be the wrist center of a spherical wrist:

https://d17h27t6h515a5.cloudfront.net/topher/2017/May/591e56d2_codecogseqn/codecogseqn.gif

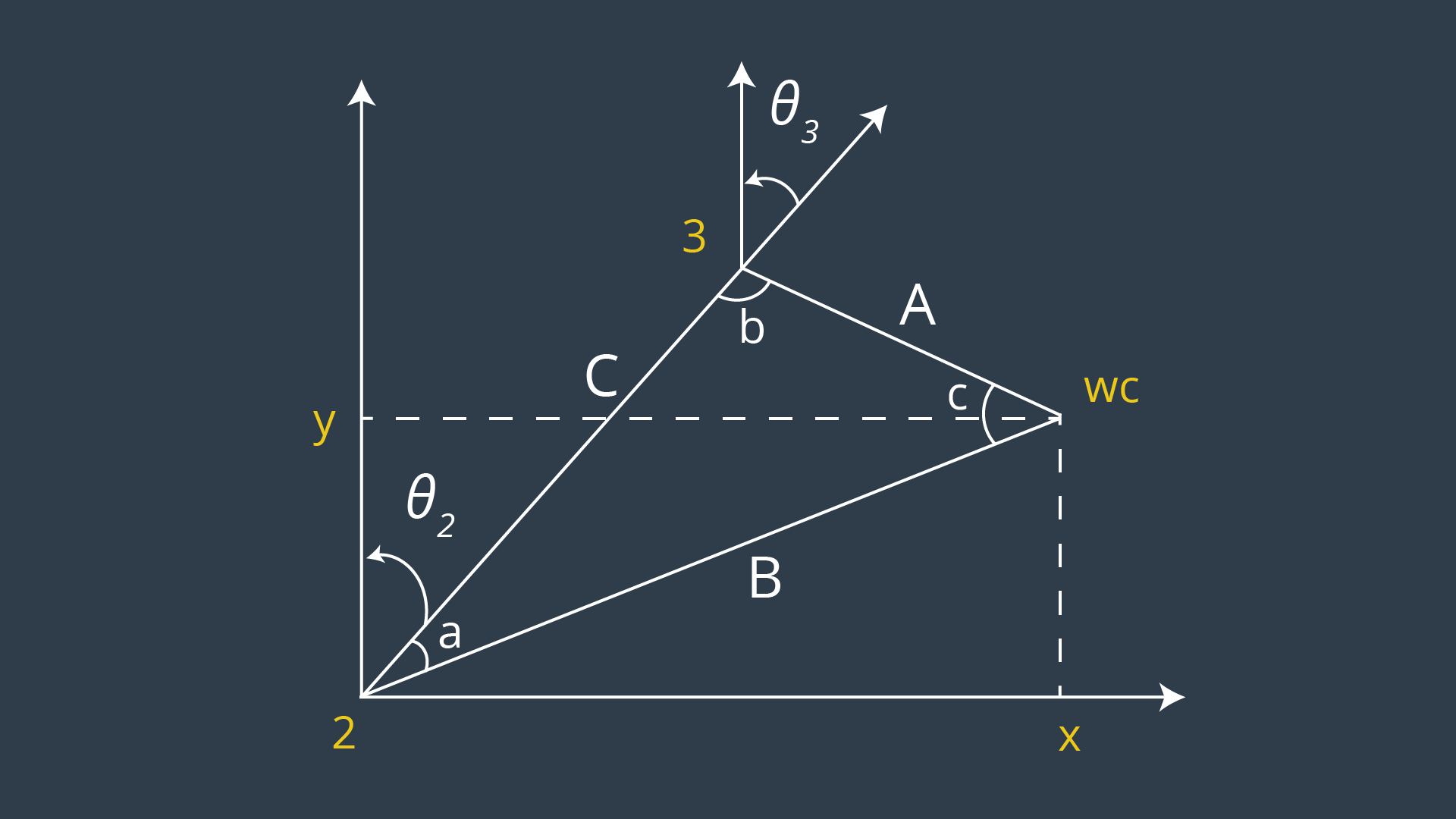
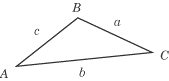
To solve for 𝜃2, 𝜃1 was imaged to equal to 0 and project links 2, 3 and 4 onto the z-y plane corresponding to the world reference frame.

Figure . Illustration to find joint angle 2 and 3.

Since the above figure represents a non-right triangle then the Law of Cosines is used to solve for it as the illustration below shows.

http://img.sparknotes.com/content/testprep/bookimgs/sat2/math2c/0005/c2=a2+b2-2abcos(c).gif

Sides A,B and C are found using the urdf file as the following:

A= 0.96+0.54 (x axis translation from link\_3 to link\_5)

C= 1.25 ( z axis translation from link\_2 to link\_3)

B= =

Angles a, b and c are found as:

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))

and therefore 𝜃2  and 𝜃3 are found as:

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

theta3 = pi / 2 - (angle\_b + 0.036) # 0.036 accounts for sag in link4 of -0.054m

**Finding last 3 angles using the wrist center**

Since the last three joints are used to orient the end effector, then, to find the value of these angles:

1. Using the individual DH transforms we can obtain the resultant transform and hence resultant rotation by:

R0\_6 = R0\_1\*R1\_2\*R2\_3\*R3\_4\*R4\_5\*R5\_6

1. Since the overall RPY (**R**oll **P**itch **Y**aw) rotation between base\_link and gripper\_link must be equal to the product of individual rotations between respective links, following holds true:

R0\_6 = Rrpy

1. calculate via application of homogeneous transforms up to the WC.

R0\_3 = R0\_1\*R1\_2\*R2\_3

1. find a set of Euler angles corresponding to the rotation matrix,

**[[](https://classroom.udacity.com/nanodegrees/nd209/parts/c199593e-1e9a-4830-8e29-2c86f70f489e/modules/8855de3f-2897-46c3-a805-628b5ecf045b/lessons/87c52cd9-09ba-4414-bc30-24ae18277d24/concepts/3bc41e14-e43d-4105-887c-8268a7402750)](https://classroom.udacity.com/nanodegrees/nd209/parts/c199593e-1e9a-4830-8e29-2c86f70f489e/modules/8855de3f-2897-46c3-a805-628b5ecf045b/lessons/87c52cd9-09ba-4414-bc30-24ae18277d24/concepts/3bc41e14-e43d-4105-887c-8268a7402750)**

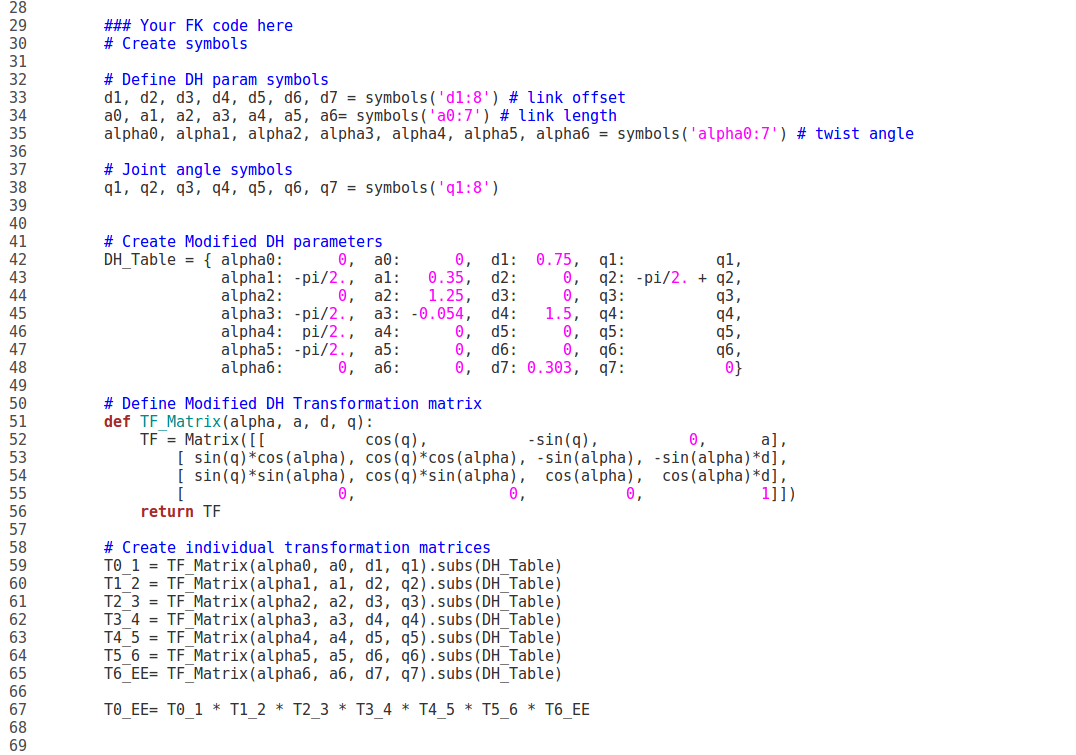
By finding this rotation matrix then the Euler angles can be found where, 𝜃4 represents roll rotation, 𝜃5 pitch rotation and 𝜃6 the yaw rotation.

theta5 = atan2(sqrt(r13\*\*2 + r33\*\*2), r23)  
    if sin(theta5 ) < 0:  
        theta4  = atan2(-r33, r13)  
        theta6 = atan2(r22, -r21)  
    else:  
        theta4 = atan2(r33, -r13)  
        theta6 = atan2(-r22, r21)

Since theta5 has a square root then the answer could have multiple solutions one in the minus and one positive, therefore to avoid unnecessary rotations only the following formulas were used in the code.

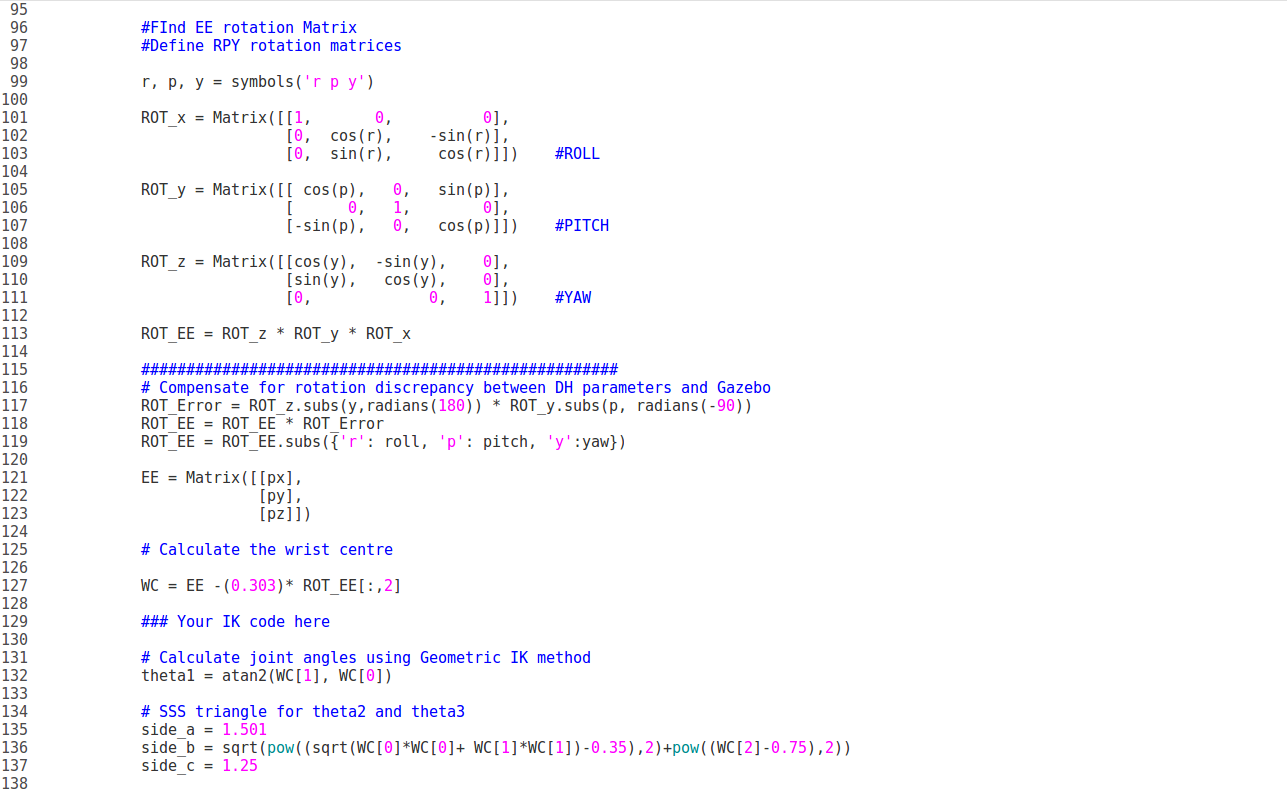
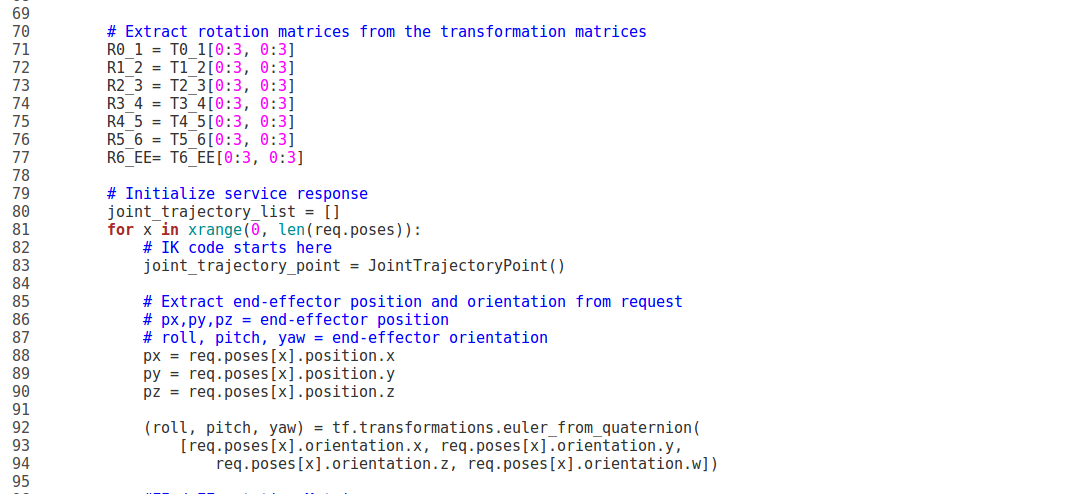
**Project Implementation**

For the IK\_server.py file, first the symbols were defined for the DH parameters, then a DH parameter table was creating listing the parameters from table1. Later, the DH transformation matrix was defined in a function to be later used to create individual transformation matrices between each frame from the base frame to the end effector frame. All the indvidual transformation matrices were later multiplied to generate the homogenous transformation matrix from of the end effector relative to the base frame.

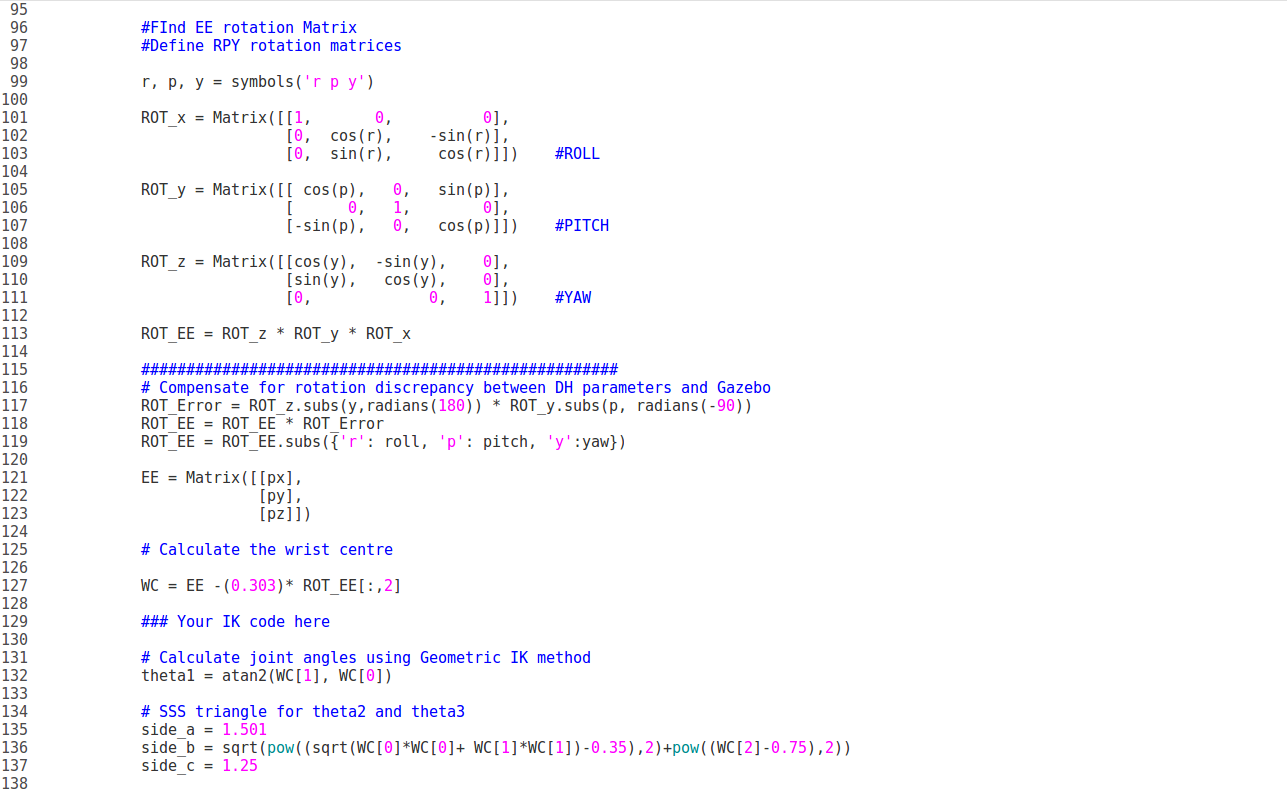
****

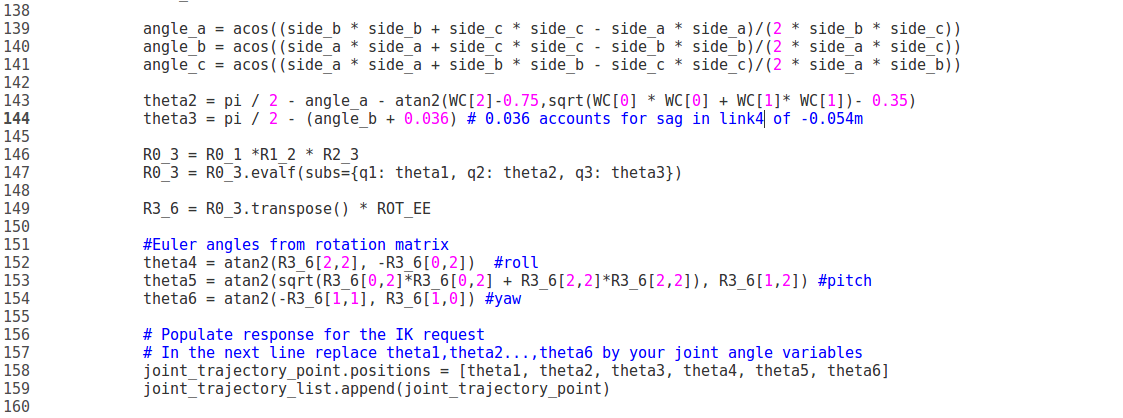
after that the individual rotation matrices were extracted from the above transformation matrices.

As for the inverse kinematics part of the code, first the end effector position and orientation were extracted from the request message, then rotation matrics from the end effector relative to the base was created along with the position vector of the end effector relative to the base.

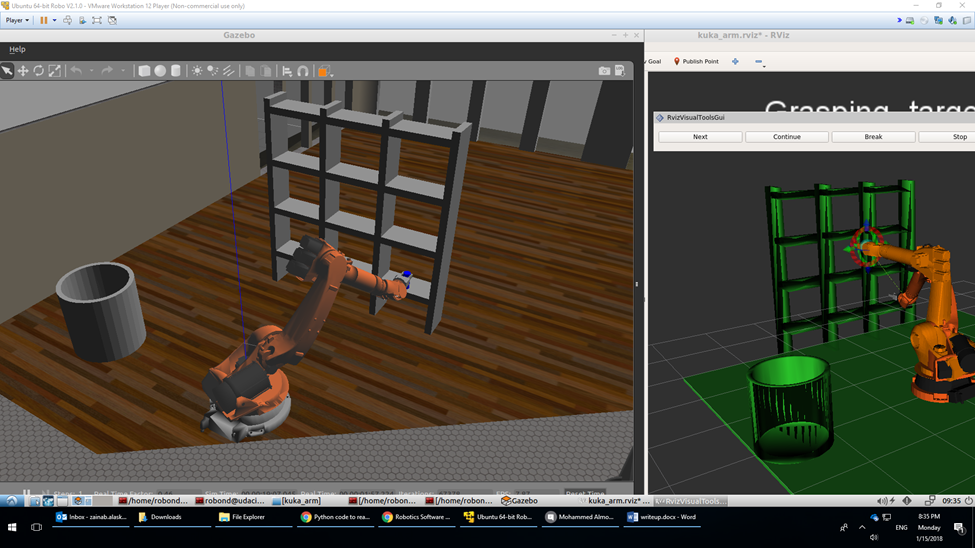
****

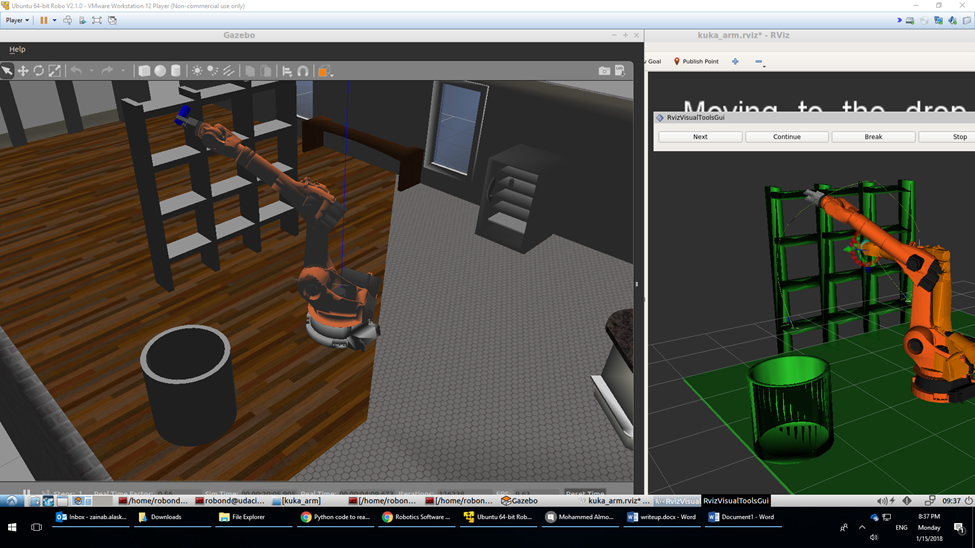
After that, the wrist center position was calculation and later each joint angle was found using the discussed formulas in the previous section.

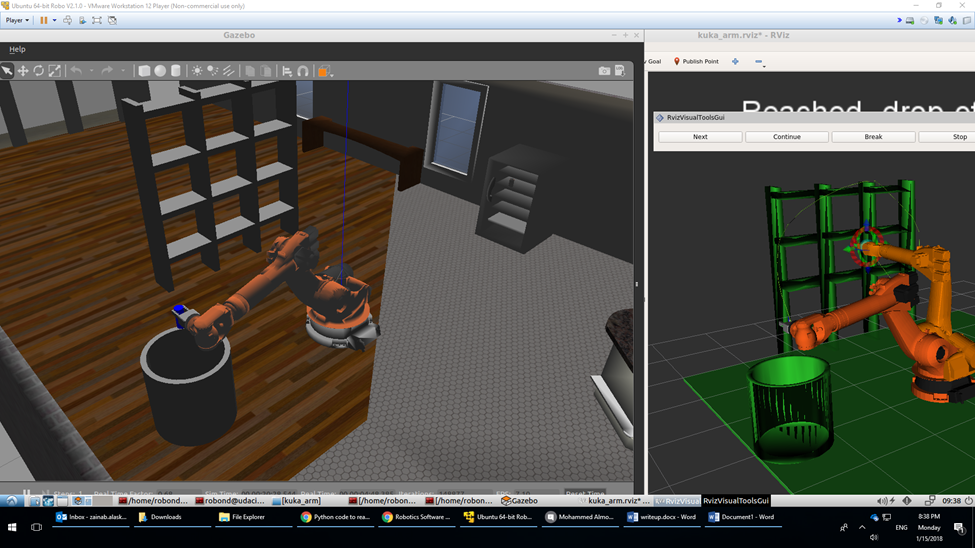
****

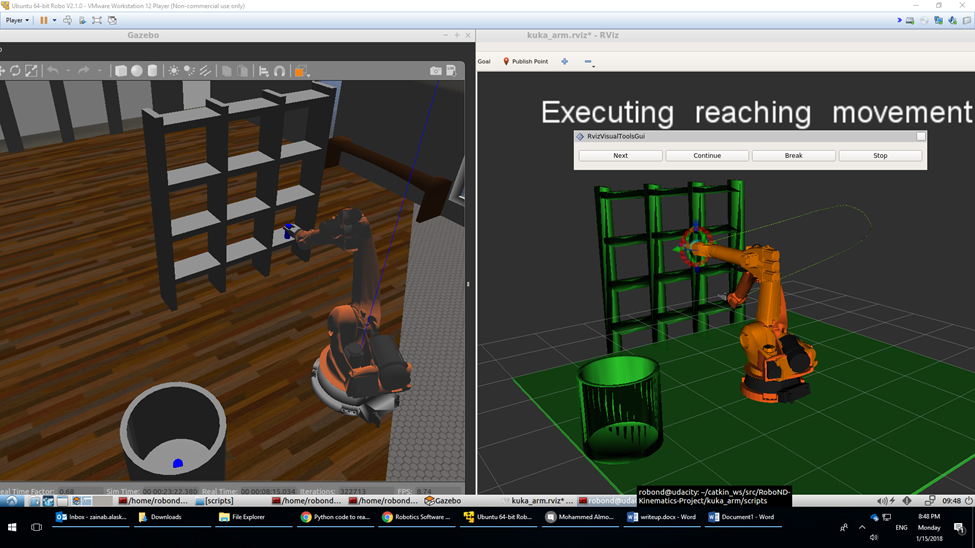


the following screenshot demonstrate the performance of the IK\_server.py script.









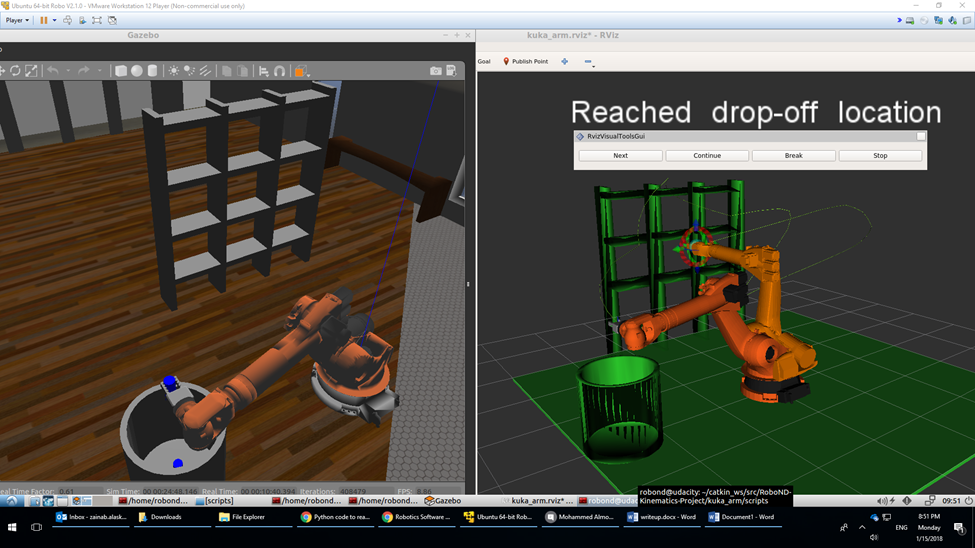


Figure . Performance of the Kuka robot using the IK\_server.py

The robot manages to successfully pick and place the object in the bin each time, and it follows the planned trajectory path.

To better enhance the code and make it faster, the program can only calculate the IK solutions to the first and last cartesian points in the planned path and then interpolate between the first angles and last angles in the path. This, will provide a faster approach and a smarter path, as it would be linear and short.