Udacity: Robotic Arm Pick & Place Report

Shane Reynolds

March 2, 2018

Contents

1 Introduction & Background

The Amazon Pick and Place Robotics Challenge is a competition designed to help increase collaboration between the industrial and robotics research communities. Amazon has successfully implemented a number of robotic systems which largely eliminate the need for activities like searching and walking in their fulfilment centres, however, one of the main challenges that Amazon is yet to solve is picking and stowing objects reliably in an unstructured environment. To successfully achieve this objective, there are a number of tasks that need to be successfully completed. These include:

1. Identification of the target object in the unstructured environment;
2. Manipulator path planning to the object;
3. Successful execution of a reach and grasp manoeuvre; and
4. Physical relocation of the grasped object to the desired location.

Figure 1: Kuka KR210 anthropomorphic industrial robot with 6 degrees of freedom

Path planning and execution of the desired manoeuvre is largely a solved problem. The move execution is dependent on a field of robotics called inverse kinematics. The inverse kinematics (IK) of a robot is the mathematical conversion of position in Cartesian space to the joint angles which allows the robot end effector to reach the desired position. Briefly, the end effector position in space can be thought of in 2 separate domains: Cartesian world coordinates, or Joint Angle space. Typically, analytical work is done in Cartesian space - three dimensional space is the native environment that humans live in and is easier to conceptualise. Robots, however, position themselves by making adjustments to electrical or hydraulic actuators - these actuators receive instructions based on Joint Angle. This project explores the IK derivation, and implementation, for the Kuka KR210. The KR210 is a 6 degree of freedom (dof) anthropomorphic robotic arm shown in Figure 1. The project culminates with the implementation of an IK server, which is a ROS service receiving a series of points in Cartesian space (world coordinate frame), and returning a vector of Joint Angles after applying the IK transform. The implementation will be undertaken in ROS, which utilises simulation engines Rviz and Gazebo. Figures 2 and 3 show the Kuka KR210 in simulation.

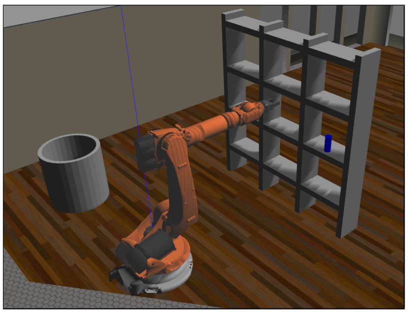

Figure 2: A picture of the KR210 in Gazebo

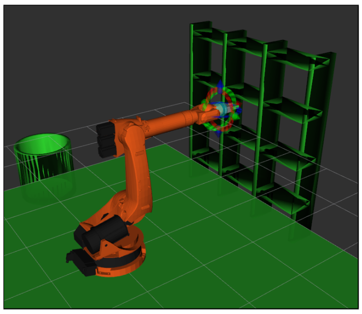

Figure 3: A picture of the KR210 in Rviz

2 Methods & Implementation

2.1 Determining the Denavit-Hartenburg Parameters

Forward and Inverse kinematic analysis relies heavily on successful specification of transformation matrices between the physical elements of the robot, which we call links. In order to determine these transformation matrices, we need to assign coordinate frames to the robot links. Doing this in an arbitrary fashion will often result in the determination of 6 parameters for each transformation matrix, which makes this process undesirably complex. Denavit and Hartenburg (1955) determined an algorithmic approach to the assignment of coordinate frames to the robot's links which reduces the number of parameters needed to describe each transformation matrix to 4. Assuming that $\hat{x}_i$, $\hat{y}_i$, and $\hat{z}_i$ are the x, y, and z axes respectively for coordinate frame i, then these parameters are defined in Table 1.

Table 1: Description of the Denavit-Hartenburg parameters

Parameter	**Description**
α_{i-1}	Twist angle, and is determined by the angle between the $\hat{z}_{i-1}$ and $\hat{z}_i$, measured about the $\hat{x}_{i-1}$ axis
a_{i-1}	Distance from $\hat{z}_{i-1}$ to $\hat{z}_i$ measured along $\hat{x}_{i-1}$, where $\hat{x}_{i-1}$ is orthogonal to $\hat{z}_{i-1}$, and $\hat{x}_{i-1}$ is orthogonal to $\hat{z}_i$
d_i	Signed distance between $\hat{x}_i$ and $\hat{x}_{i-1}$, measured along $\hat{z}_i$
θ_i	Angle between $\hat{x_{i-1}}$ and $\hat{x}_i$, measured about $\hat{z}_i$

The KR210 has a base link, 6 degrees of freedom, and an end effector. Each of the links require a coordinate frame assignment, making a total requirement of 8 coordinate frames. Each of the joints were systematically labelled from 1 to 6, starting with the joint closest to the `base_link`. Following this, each of the links were assigned a number from 0 to 7. It must be noted that link 0 is actually the `base_link`, and link 7 is the `end_effector`. For the sake of simplicity, the `base_link` and `end_effector` will retain their names throughout this report. Coordinate frames were assigned to the links according to the DH procedure. Each link can be thought of as being associated with a joint. The `base_link` is associated with the fixed ground, link 1 is associated with joint 1, and so on. To assign the coordinate frame to a link, DH requires the $\hat{z}_i$ coordinate axis for link i to pass through the joint i axis of rotation. To start the DH convention of coordinate frame assignment, the `base` frame is assigned arbitrarily. Each $\hat{x}_i$ axis, for coordinate frame i, is determined using $\hat{z}_i$, and $\hat{z}_{1+1}$. The $\hat{x}_i$ axes are assigned dependent on whether the $\hat{z}_i$, and $\hat{z}_{i+1}$ axes are:

1. **Skewed:** if the $\hat{z}_i$ and $\hat{z}_{i+1}$ axes are skewed, then the $\hat{x}_i$ axis is assigned along the normal from $\hat{z}_i$ to $\hat{z}_{i+1}$.

2. **Intersecting:** if the $\hat{z}_i$ and $\hat{z}_{i+1}$ axes intersect, then the $\hat{x}_i$ axis is assigned in an arbitrary position such that it is normal to the plane formed by $\hat{z}_i$ and $\hat{z}_{i+1}$

3. **Coincident:** if the $\hat{z}_i$ and $\hat{z}_{i+1}$ axes are parallel or coincident, then the $\hat{x}_i$ axis assignment is arbitrary along the $\hat{z}_i axis$

To reiterate, the $\hat{x}_i$ axis is assigned dependent on the geometric orientation of the $\hat{z}_i$, and $\hat{z}_{i+1}$ axes. The $\hat{y}_i$ axis is assigned to complete the right handed coordinate frame assignment. The full DH coordinate frame assignment for the KR210 can be seen in Figure 4.

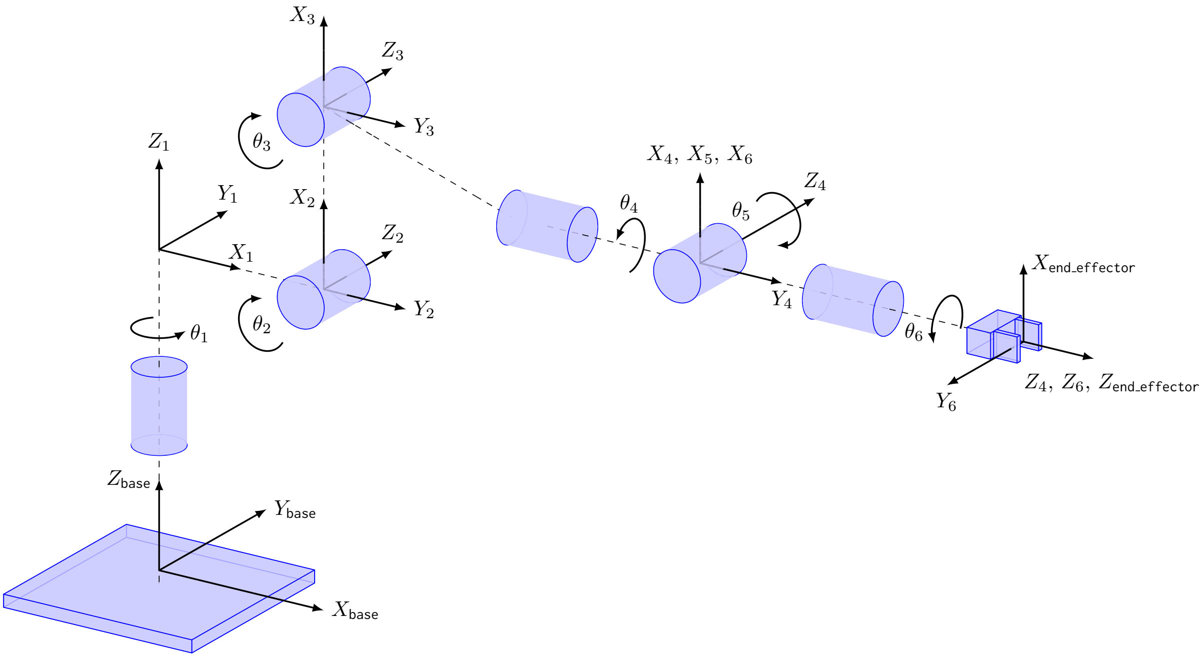

Figure 4: Sexy robot drawing

The DH parameters, which are used to specify the transformations from one coordinate frame to another, are determined once the coordinate frames have been assigned. The full DH parameter specification can be seen in Table 2. It must be noted that the values for d_i and a_{i-1} were found using the unified robot description format (urdf) file, which contains the model specifications for Gazebo. The full urdf file can be seen in Appendix A. The DH parameter specifications are used in conjunction with equation (1) to specify the transformation matrices from coordinate frame $i-1$ to coordinate frame i - this is further explored in Section 2.2.

$$
{}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cdot \cos\alpha_{i-1} & \cos\theta_i \cdot \cos\alpha_{i-1} & -\sin\alpha_{i-1} & -d_i \cdot \sin\alpha_{i-1} \\ \sin\theta_i \cdot \sin\alpha_{i-1} & \cos\theta_i \cdot \sin\alpha_{i-1} & \cos\alpha_{i-1} & d_i \cdot \cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}
$$

2.2 Forward Kinematics

The forward kinematics problem is concerned with taking the agular positions of the individual joints and finding the position of the robot's end effector in three dimensional Cartesian space. This section of the report is broken down into three subsections:

Table 2: DH parameter table

${}^{i-1}T_i$	d_i	θ_i	α_{i-1}	a_{i-1}
${}^{\text{base}}T_1$	0.750	θ_1	0	0.000
${}^{1}T_2$	0.000	$\theta_2 - \frac{\pi}{2}$	$-\pi/2$	0.350
${}^{2}T_3$	0.000	θ_3	0	1.250
${}^{3}T_4$	1.500	θ_4	$-\pi/2$	-0.054
${}^{4}T_5$	0.000	θ_5	$\pi/2$	0.000
${}^{5}T_6$	0.000	θ_6	$-\pi/2$	0.000
${}^{6}T_{\text{end\_eff}}$	0.303	0	0	0.000

1. Derivation of the transformation matrices, ${}^{i-1}T_i$
2. The correction of transformation matrices NEED TO FINISH THIS DESCRIPTION
3. Verification of transformation matrices

2.2.1 Derivation of the transformation matrices

Using the DH parameters derived in Section 2.1, in addition to equation (1), we can derive a series of matrices, ${}^{i-1}T_i$, which describe the transformation of a vector in one coordinate frame i, to that of another coordinate frame $i-1$.

The individual transformation matrices are shown below:

$$
{}^{\text{base}}T_1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0.75 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\quad
{}^{1}T_2 = \begin{bmatrix} \sin\theta_2 & \cos\theta_2 & 0 & 0.35 \\ 0 & 0 & 1 & 0 \\ \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{2}T_3 = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & 1.25 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\quad
{}^{3}T_4 = \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & -0.054 \\ 0 & 0 & 1 & 1.5 \\ -\sin\theta_4 & -\cos\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{4}T_5 = \begin{bmatrix} \cos\theta_5 & -\sin\theta_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin\theta_5 & \cos\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\quad
{}^{5}T_6 = \begin{bmatrix} \cos\theta_6 & -\sin\theta_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin\theta_6 & -\cos\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{5}T_{\text{end\_eff}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.303 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

The full transformation from the `base_link` to the `end_effector` is determined by matrix multiplication, shown in equation.

$$
{}^{\text{base\_link}}T_{\text{end\_effector}} = {}^{\text{base\_link}}T_1 \cdot {}^{1}T_2 \cdot {}^{2}T_3 \cdot {}^{3}T_4 \cdot {}^{4}T_5 \cdot {}^{5}T_6 \cdot {}^{6}T_{\text{end\_effector}} \tag{2}
$$

For the sake of brevity, the full specification for equation (2), which is ${}^{\text{base\_link}}T_{\text{end\_effector}}$, has not been shown - showing this transformation matrix without evaluating θ_i would take several paragraphs.

2.2.2 Correction of Transformation Matrices

The orientation of the DH base frame, shown in Figure 3, has been selected so that it aligns with the world simulation frame (defined in the urdf file). It must be noted, however, that the final DH frame for the gripper does not have the same orientation as the urdf file since the DH algorithm was employed to assign the frames. This has ramifications for any forward kinematic analysis we perform using transformation matrices defined with DH parameters. The difference in frame orientation can be better understood by comparing Figures 5 and 6.

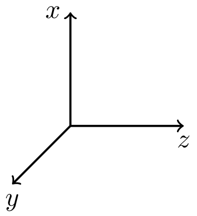

Figure 5: The orientation of the DH frame with respect to the world frame which is located at the base of the robot, shown in Figure 6

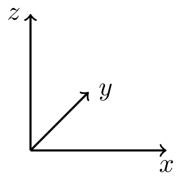

Figure 6: The orientation of thhe world coordinate frame.

To provide a correction to the transformation matrix, ${}^{\text{base}}T_{\texttt{end\_eff}}$, two steps need to be taken:

1.
2.

2.2.3 Verification of Transformation Matrices

To provide some assurance that the Python implementation is correct, analysis was undertaken using a script called `forwardKinematics.py`, which can be found in Appendix B. The ROS launch script, `forward_kinematics.launch`, provides a simulation of the robot in which the joint angles could be manually adjusted, and the position and orientation of the robot's frames observed. This is shown in Figure 5.

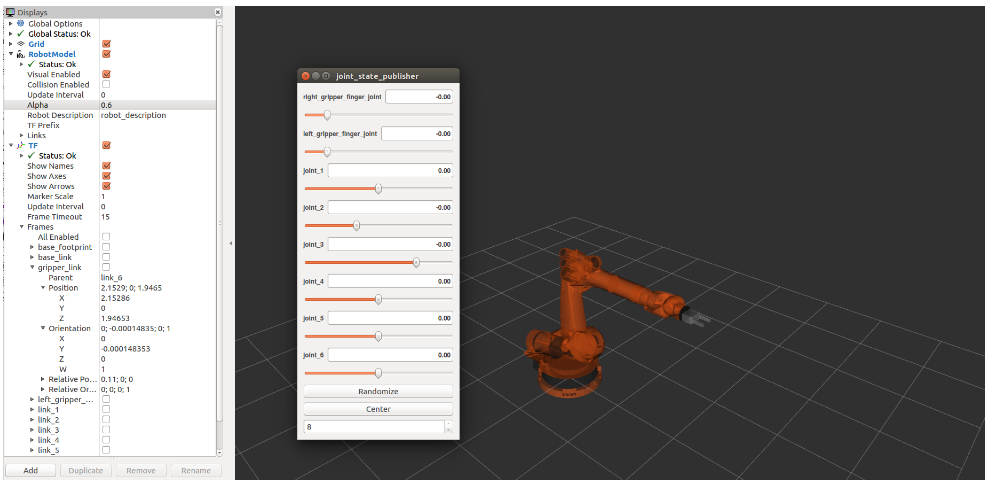

Figure 7: text

Rviz reports the postion of the gripper frame as a position vector, however, the gripper orientation is reported in Quarternions, which have to be converted to a rotation matrix in order to compare this to the calculated results from `forwardKinematics.py`. The Python script `forwardKinematics.py` contains four test cases with varying angles θ_i, such that $i \in \{1, 2, 3, 4, 5, 6\}$. These test cases can be seen in Table 3.

Table 3: Test cases for the joint angles, which were entered into the ROS similation to observe the end effector position and orientation.

Test Case	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
Case 1	0	0	0	0	0	0
Case 2	-0.65	0.45	-0.37	0.96	0.78	0.46
Case 3	-0.79	-0.11	-2.34	1.96	1.14	-3.69
Case 4	-2.99	-0.12	0.94	4.06	1.29	-4.15

Each case was entered into the simulation, and the position vector was recorded, along with the orientation Quaternion. The recorded Qaternion was converted to a rotation matrix from which roll, pitch, and yaw can be read from the main diagonal of the matrix. The results of this process can be seen in Table 4.

Table 4: Position vector, Quaternion, and rotation matrix observered from the manual simulation of the KR210.

Case	**Position Vector**	**Quaternion**	**Rotation Matrix**
1	$\begin{bmatrix}2.153 & 0.000 & 1.946\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}$	$\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$
2	$\begin{bmatrix}2.167 & -1.429 & 1.560\end{bmatrix}$	$\begin{bmatrix}0.698 & 0.183 & -0.153 & 0.674\end{bmatrix}$	$\begin{bmatrix}0.886 & 0.462 & 0.033\\ 0.049 & -0.022 & -0.998\\ -0.461 & 0.886 & -0.043\end{bmatrix}$
3	$\begin{bmatrix}-0.566 & 0.940 & 2.993\end{bmatrix}$	$\begin{bmatrix}0.612 & 0.488 & 0.388 & 0.485\end{bmatrix}$	$\begin{bmatrix}0.221 & 0.221 & 0.949\\ 0.975 & -0.051 & -0.215\\ 0.001 & 0.973 & -0.227\end{bmatrix}$
4	$\begin{bmatrix}-1.393 & 0.017 & 0.915\end{bmatrix}$	$\begin{bmatrix}0.013 & -0.229 & 0.901 & 0.368\end{bmatrix}$	$\begin{bmatrix}-0.728 & -0.669 & -0.145\\ 0.657 & -0.623 & -0.423\\ 0.192 & -0.403 & 0.895\end{bmatrix}$

Running the `forwardKinematics.py` script outputs the numerical, corrected transformation matrix, ${}^{\mathtt{i-1}}T_{\mathtt{i}}$, for each of the four test cases. The results can be seen in Table 5. Comparing the simulation results shown in Table 4 with the calcuated results from the transformation matrix, shown in Table 5, we see that position vecotrs and rotation matrices are almost identical for all four test cases. There are some small variations, however, this can be, in part, attributed to numerical truncation when performing calculations.

Table 5: Calculated position vector and orientation matrix using the matrix transfrom from the end effector to the base link.

Case	Position Vector	Rotation Matrix
1	$\begin{bmatrix}2.153 & 0 & 1.946\end{bmatrix}$	$\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$
2	$\begin{bmatrix}2.167 & -1.428 & 1.562\end{bmatrix}$	$\begin{bmatrix}0.887 & 0.460 & 0.030\\ 0.049 & -0.029 & -0.998\\ -0.458 & 0.887 & -0.049\end{bmatrix}$
3	$\begin{bmatrix}-0.573 & 0.941 & 2.99\end{bmatrix}$	$\begin{bmatrix}0.216 & 0.221 & 0.951\\ 0.976 & -0.049 & -0.210\\ 0.001 & 0.974 & -0.227\end{bmatrix}$
4	$\begin{bmatrix}-1.389 & 0.022 & 0.916\end{bmatrix}$	$\begin{bmatrix}-0.723 & -0.674 & -0.145\\ 0.662 & -0.619 & -0.423\\ 0.195 & 0.402 & 0.894\end{bmatrix}$

2.3 Inverse Kinematics

Inverse kinematics (IK) is the process of determining the joint angles for each degrees of freedom in a robotic system, given the position of the robot's end effector in Cartesian space. The Kuka KR210 has 6 degrees of freedom, and hence, there are 6 joint angles which need to be determined. The anthropomorphic arm design allows us to exploit the geometry in the final 3 joints of the system. It is the intersection of the axes of rotation that is the important geometrical characteristic. This configuration of the final three joints is refered to as a spherecal wrist. This allow kinematic decoupling of the first three joints from the last three joints, providing the facility to find a closed form solution to the problem. The closed form solution is presented below, in two parts: first three joint angles, and final three joint angles

2.3.1 First Three Joint Angles (θ_1, θ_2, and θ_3)

Consider the first three joints in a pose shown in Figure 8. The point $(\omega_x, \omega_y, \omega_z)$ represents the location of the spherical wrist. Using this information, the first joint angle, θ_1, can be found using basic trigonometry:

$$\theta_1 = \text{atan2}\left(\frac{\omega_y}{\omega_x}\right) \tag{3}$$

To determine expressions for θ_2 and θ_3, we focus our attention on joints 2 and 3, as shown in Figure 9. Considering the triangle formed by the vertices at J_2, J_3, and W, we can find an experssion for η, as follows:

$$\begin{aligned}180^o &= \eta + 92.06^o + \theta_3\\ \eta &= 87.94^o - \theta_3\\ \eta &= 1.5348\text{rad} - \theta_3\end{aligned} \tag{4}$$

Employing the cosine rule on the triangle formed by vertices J_2, J_3, and W, we note that:

$$L^2 = k_1^2 + k_2^2 - 2 \cdot k_1 \cdot k_2 \cdot \cos\eta$$

Rearranging this expression we get the following expression for $\cos\eta$:

$$\cos\eta = \frac{k_1^2 + k_2^2 - L^2}{2 \cdot k_1 \cdot k_2} := D \tag{5}$$

Equation (5), along with Pythagoras' theorem, allows us to determine an expression for $\sin\eta$:

$$\sin\eta = \sqrt{1 - D^2} \tag{6}$$

Using equations (5) and (6), we get:

$$\begin{aligned} \tan\eta &= \frac{\sin\eta}{\cos\eta} = \frac{\sqrt{1-D^2}}{D} \\ \eta &= \text{atan2}\left(\frac{\sqrt{1-D^2}}{D}\right) \end{aligned} \tag{7}$$

Substituting equation (4) into equation (7), we arrive at an expression for θ_3:

$$\theta_3 = 1.5348\text{rad} - \text{atan2}\left(\frac{\sqrt{1-D^2}}{D}\right) \tag{8}$$

Again, considering the triangle formed by the vertices J_2, J_3, and W, we can use the cosine rule to get the following expression:

$$k_2^2 = k_1^2 + L^2 - 2 \cdot k_1 \cdot L \cdot \cos\xi$$

Rearranging the above equation we find an expression for $\cos\xi$:

$$\cos\xi = \frac{k_1^2 + L^2 - k_2^2}{2 \cdot k_1 \cdot L} := K \tag{9}$$

Equation (9), along with Pythagoras' theorem, allows us to determine an expression for $\sin\xi$:

$$\sin\xi = \sqrt{1 - K^2} \tag{10}$$

Using equations (9) and (10), we get:

$$\begin{aligned} \tan\xi &= \frac{\sin\xi}{\cos\xi} = \frac{\sqrt{1-K^2}}{K} \\ \xi &= \text{atan2}\left(\frac{\sqrt{1-K^2}}{K}\right) \end{aligned} \tag{11}$$

Now, considering the right angled triangle formed between the line connecting points J_2 and W, we can find and expression for α:

$$\alpha = \text{atan2}\left(\frac{s}{r}\right) \tag{12}$$

Finally, we note that the angle between assigned DH coordinate axes $\hat{x}_1$ and $\hat{x}_2$ is $\pi/2 - \theta_2$, and hence we get the following expression:

$$\begin{aligned} \frac{\pi}{2} - \theta_2 &= \alpha + \xi \\ \theta_2 &= \frac{\pi}{2} - \alpha - \xi \end{aligned} \tag{13}$$

Substituing equations (11) and (12) into equation (13), we arrive at an expression for θ_2:

$$\theta_2 = \frac{\pi}{2} - \text{atan2}\left(\frac{s}{r}\right) - \text{atan2}\left(\frac{\sqrt{1-K^2}}{K}\right) \tag{14}$$

2.3.2 Final Three Joint Angles (θ_4, θ_5, and θ_6)

To find the final three joint angles, θ_4, θ_5, and θ_6 we expolit the following relationship between the FK rotation matrix ${}^0R_6$, and the desired orientation of the end gripper, R_{rpy}, which is given in an IK problem. We note that:

$$^{\text{base}}R_6 = R_{rpy} \tag{15}$$

It is important to note that the orientation of the robotic frame at joint 6 will be the same as the `end_effector` orientation, since the `end_effector` is simply a translation from joint 6 , with no rotation. From our FK analysis, we have the ability to break up the rotation matrix, in equation (15), as follows:

$$^{\text{base}}R_6 = {}^{\text{base}}R_3 \cdot {}^3R_6 = R_{rpy} \tag{16}$$

Hence, rearranging equation (16), we get:

$$^3R_6 = \left({}^{\text{base}}R_3\right)^{-1} \cdot R_{rpy}$$

Since any rotation matrix is orthogonal, we can write:

$$^3R_6 = \left({}^{\text{base}}R_3\right)^{T} \cdot R_{rpy} \tag{17}$$

3 Results & Conclusion

Discussion of the results of the robot kinematics.

4 Further Enhancements

The wrist turns around a bit - need to make sure that this doesn't happen and why this is happening. It happens because the wrist has more that 360 twist capability, however, the way it is set up does not allow for this (it has hard clamps at $-\pi$ and π). Also an error tracking capability would be nice on the project also.

5 Appendix A

```
<?xml version="1.0"?>
<robot name="kuka_arm" xmlns:xacro="http://www.ros.org/wiki/xacro">

  <!--Define constants-->
  <xacro:property name="pi" value="3.141592654"/>
  <xacro:property name="deg" value="0.017453293"/><!--degrees to radians-->
  <xacro:property name="mass0" value="150" />
  <xacro:property name="mass1" value="138" />
  <xacro:property name="mass2" value="95" />
  <xacro:property name="mass3" value="71" />
  <xacro:property name="mass4" value="17" />
  <xacro:property name="mass5" value="7" />
  <xacro:property name="mass6" value="0.5" />
  <xacro:property name="gripper_mass" value="0.3" />
  <xacro:property name="finger_mass" value="0.1" />
  <xacro:property name="gripper_type" value="two_finger" /><!--value="vacuum" or "two_finger"-->
  <xacro:property name="vacuum_gripper_radius" value="0.1" />

  <!--Import gazebo elements-->
  <xacro:include filename="$(find kuka_arm)/urdf/kr210.gazebo.xacro" />

  <!--Links-->
  <link name="base_footprint"/>

  <link name="base_link">
    <inertial>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <mass value="${mass0}"/>
      <inertia ixx="60" ixy="0" ixz="0" iyy="70" iyz="0" izz="100"/>
    </inertial>
    <visual>
      <origin xyz="0 0 0" rpy="${-pi/2} 0 0"/>
      <geometry>
        <mesh filename="package://kuka_arm/meshes/kr210l150/visual/base_link.dae"/>
      </geometry>
```

```
      <material name="">
        <color rgba="0.75294 0.75294 0.75294 1"/>
      </material>
    </visual>
    <collision>
      <origin xyz="0 0 0" rpy="${-pi/2} 0 0"/>
      <geometry>
        <mesh filename="package://kuka_arm/meshes/kr210l150/collision/base_link.stl"/>
      </geometry>
    </collision>
  </link>
  <link name="link_1">
    <inertial>
      <origin xyz="0 0 0.4" rpy="0 0 0"/>
      <mass value="${mass1}"/>
      <inertia ixx="30" ixy="0" ixz="0" iyy="50" iyz="0" izz="50"/>
    </inertial>
    <visual>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <mesh filename="package://kuka_arm/meshes/kr210l150/visual/link_1.dae"/>
      </geometry>
      <material name="">
        <color rgba="0.75294 0.75294 0.75294 1"/>
      </material>
    </visual>
    <collision>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <mesh filename="package://kuka_arm/meshes/kr210l150/collision/link_1.stl"/>
      </geometry>
    </collision>
  </link>
  <link name="link_2">
    <inertial>
      <origin xyz="0 0 0.44751" rpy="0 0 0"/>
      <mass value="${mass2}"/>
      <inertia ixx="40" ixy="0" ixz="0" iyy="40" iyz="0" izz="10"/>
    </inertial>
    <visual>
      <origin xyz="0 0 0" rpy="${-pi/2} 0 0"/>
      <geometry>
        <mesh filename="package://kuka_arm/meshes/kr210l150/visual/link_2.dae"/>
      </geometry>
      <material name="">
        <color rgba="0.75294 0.75294 0.75294 1"/>
      </material>
    </visual>
    <collision>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <mesh filename="package://kuka_arm/meshes/kr210l150/collision/link_2.stl"/>
      </geometry>
    </collision>
  </link>
  <link name="link_3">
    <inertial>
      <origin xyz="0.18842 0.18344 -0.042799" rpy="0 0 0"/>
      <mass value="${mass3}"/>
      <inertia ixx="1" ixy="0" ixz="0" iyy="10" iyz="0" izz="10"/>
    </inertial>
    <visual>
      <origin xyz="0 -0.184 0" rpy="${-pi/2} 0 0"/>
      <geometry>
        <mesh filename="package://kuka_arm/meshes/kr210l150/visual/link_3.dae"/>
      </geometry>
      <material name="">
        <color rgba="0.75294 0.75294 0.75294 1"/>
      </material>
    </visual>
    <collision>
      <origin xyz="0 -0.184 0" rpy="0 0 0"/>
      <geometry>
        <mesh filename="package://kuka_arm/meshes/kr210l150/collision/link_3.stl"/>
      </geometry>
    </collision>
  </link>
  <link name="link_4">
    <inertial>
      <origin xyz="0.27146 -0.007326 0" rpy="0 0 0"/>
      <mass value="${mass4}"/>
      <inertia ixx="0.18" ixy="0" ixz="0" iyy="0.55" iyz="0" izz="0.64"/>
    </inertial>
    <visual>
      <origin xyz="0 0 0" rpy="${-pi/2} 0 0"/>
      <geometry>
        <mesh filename="package://kuka_arm/meshes/kr210l150/visual/link_4.dae"/>
      </geometry>
      <material name="">
        <color rgba="0.75294 0.75294 0.75294 1"/>
      </material>
    </visual>
    <collision>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <mesh filename="package://kuka_arm/meshes/kr210l150/collision/link_4.stl"/>
      </geometry>
    </collision>
  </link>
  <link name="link_5">
    <inertial>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <mass value="${mass5}"/>
      <inertia ixx="0.04" ixy="0" ixz="0" iyy="0.068" iyz="0" izz="0.068"/>
    </inertial>
    <visual>
      <origin xyz="0 0 0" rpy="${-pi/2} 0 0"/>
```

```
    <geometry>
      <mesh filename="package://kuka_arm/meshes/kr210l150/visual/link_5.dae"/>
    </geometry>
    <material name="">
      <color rgba="0.75294 0.75294 0.75294 1"/>
    </material>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <mesh filename="package://kuka_arm/meshes/kr210l150/collision/link_5.stl"/>
    </geometry>
  </collision>
</link>
<link name="link_6">
  <inertial>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <mass value="${mass6}"/>
    <inertia ixx="0.004" ixy="0" ixz="0" iyy="0.003" iyz="0" izz="0.003"/>
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <mesh filename="package://kuka_arm/meshes/kr210l150/visual/link_6.dae"/>
    </geometry>
    <material name="">
      <color rgba="0.75294 0.75294 0.75294 1"/>
    </material>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <mesh filename="package://kuka_arm/meshes/kr210l150/collision/link_6.stl"/>
    </geometry>
  </collision>
</link>

<!--Gripper links-->
<!--Vacuum gripper-->
<xacro:if value="${gripper_type == 'vacuum'}">
<link name="gripper_link">
  <inertial>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <mass value="${gripper_mass}"/>
    <inertia ixx="0.005" ixy="0" ixz="0" iyy="0.00" iyz="0" izz="0.00"/>
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <sphere radius="${vacuum_gripper_radius}"/>
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <sphere radius="${vacuum_gripper_radius}"/>
    </geometry>
  </collision>
</link>
<joint name="gripper_joint" type="fixed">
  <parent link="link_6"/>
  <child link="gripper_link"/>
  <origin xyz="0.0375 0 0" rpy="0 0 0"/>
</joint>
</xacro:if>

<!--Two-finger gripper-->
<xacro:if value="${gripper_type == 'two_finger'}">
<link name="gripper_link">
  <inertial>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <mass value="${gripper_mass}"/>
    <inertia ixx="0.0001" ixy="0" ixz="0" iyy="0.0002" iyz="0" izz="0.0001"/>
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <mesh filename="package://kuka_arm/meshes/gripper/gripper_base.dae"/>
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <mesh filename="package://kuka_arm/meshes/gripper/gripper_base.dae"/>
    </geometry>
  </collision>
</link>
<link name="right_gripper_finger_link">
  <inertial>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <mass value="${finger_mass}"/>
    <inertia ixx="0.002" ixy="0" ixz="0" iyy="0.0002" iyz="0" izz="0.0002"/>
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <mesh filename="package://kuka_arm/meshes/gripper/finger_right.dae"/>
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <mesh filename="package://kuka_arm/meshes/gripper/finger_right_collision.dae"/>
    </geometry>
  </collision>
</link>
<link name="left_gripper_finger_link">
```

```
    <inertial>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <mass value="${finger_mass}"/>
      <inertia ixx="0.002" ixy="0" ixz="0" iyy="0.0002" iyz="0" izz="0.0002"/>
    </inertial>
    <visual>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <mesh filename="package://kuka_arm/meshes/gripper/finger_left.dae"/>
      </geometry>
    </visual>
    <collision>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <mesh filename="package://kuka_arm/meshes/gripper/finger_left_collision.dae"/>
      </geometry>
    </collision>
  </link>

  <joint name="right_gripper_finger_joint" type="prismatic">
    <origin rpy="0 0 0" xyz="0.15 -0.0725 0" />
    <parent link="gripper_link" />
    <child link="right_gripper_finger_link" />
    <axis xyz="0 1 0" />
    <limit effort="100" lower="-0.01" upper="0.06" velocity="0.05" />
    <dynamics damping="0.7" />
  </joint>
  <joint name="left_gripper_finger_joint" type="prismatic">
    <origin rpy="0 0 0" xyz="0.15 0.0725 0" />
    <parent link="gripper_link" />
    <child link="left_gripper_finger_link" />
    <axis xyz="0 -1 0" />
    <limit effort="100" lower="-0.01" upper="0.06" velocity="0.05" />
    <dynamics damping="0.7" />
  </joint>
  <joint name="gripper_joint" type="fixed">
    <parent link="link_6"/>
    <child link="gripper_link"/>
    <origin xyz="0.11 0 0" rpy="0 0 0"/><!--0.087-->
    <axis xyz="0 1 0" />
  </joint>

  <transmission name="tran7">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="left_gripper_finger_joint">
      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
    </joint>
    <actuator name="motor7">
      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>

  <transmission name="tran8">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="right_gripper_finger_joint">
      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
    </joint>
    <actuator name="motor8">
      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>

  </xacro:if>

  <!-- joints -->
  <joint name="fixed_base_joint" type="fixed">
    <parent link="base_footprint"/>
    <child link="base_link"/>
    <origin xyz="0 0 0" rpy="0 0 0"/>
  </joint>
  <joint name="joint_1" type="revolute">
    <origin xyz="0 0 0.33" rpy="0 0 0"/>
    <parent link="base_link"/>
    <child link="link_1"/>
    <axis xyz="0 0 1"/>
    <limit lower="${-185*deg}" upper="${185*deg}" effort="300" velocity="${123*deg}"/>
  </joint>
  <joint name="joint_2" type="revolute">
    <origin xyz="0.35 0 0.42" rpy="0 0 0"/>
    <parent link="link_1"/>
    <child link="link_2"/>
    <axis xyz="0 1 0"/>
    <limit lower="${-45*deg}" upper="${85*deg}" effort="300" velocity="${115*deg}"/>
  </joint>
  <joint name="joint_3" type="revolute">
    <origin xyz="0 0 1.25" rpy="0 0 0"/>
    <parent link="link_2"/>
    <child link="link_3"/>
    <axis xyz="0 1 0"/>
    <limit lower="${-210*deg}" upper="${(155-90)*deg}" effort="300" velocity="${112*deg}"/>
  </joint>
  <joint name="joint_4" type="revolute">
    <origin xyz="0.96 0 -0.054" rpy="0 0 0"/>
    <parent link="link_3"/>
    <child link="link_4"/>
    <axis xyz="1 0 0"/>
    <limit lower="${-350*deg}" upper="${350*deg}" effort="300" velocity="${179*deg}"/>
  </joint>
  <joint name="joint_5" type="revolute">
    <origin xyz="0.54 0 0" rpy="0 0 0"/>
    <parent link="link_4"/>
    <child link="link_5"/>
    <axis xyz="0 1 0"/>
    <limit lower="${-125*deg}" upper="${125*deg}" effort="300" velocity="${172*deg}"/>
  </joint>
```

```
  <joint name="joint_6" type="revolute">
    <origin xyz="0.193 0 0" rpy="0 0 0"/>
    <parent link="link_5"/>
    <child link="link_6"/>
    <axis xyz="1 0 0"/>
    <limit lower="${-350*deg}" upper="${350*deg}" effort="300" velocity="${219*deg}"/>
  </joint>

  <!--Transmission and actuators-->
  <transmission name="tran1">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="joint_1">
      <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
    </joint>
    <actuator name="motor1">
      <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>

  <transmission name="tran2">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="joint_2">
      <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
    </joint>
    <actuator name="motor2">
      <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>

  <transmission name="tran3">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="joint_3">
      <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
    </joint>
    <actuator name="motor3">
      <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>

  <transmission name="tran4">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="joint_4">
      <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
    </joint>
    <actuator name="motor4">
      <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>

  <transmission name="tran5">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="joint_5">
      <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
    </joint>
    <actuator name="motor5">
      <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>

  <transmission name="tran6">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="joint_6">
      <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
    </joint>
    <actuator name="motor6">
      <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>

</robot>
```