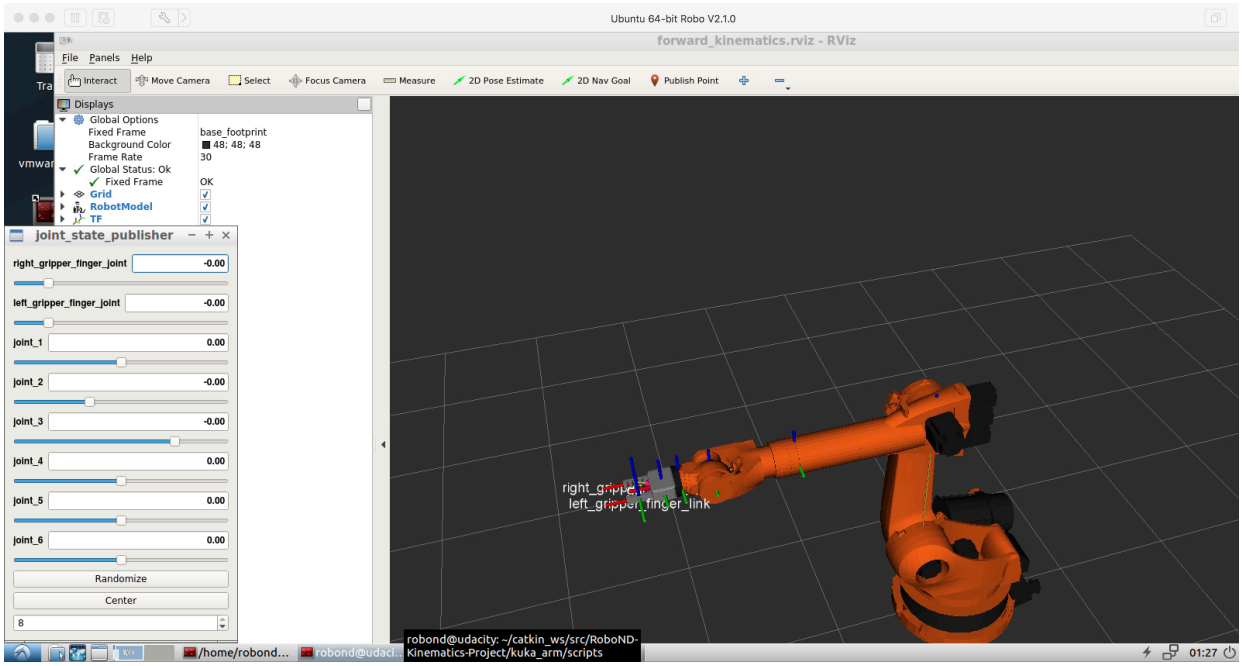
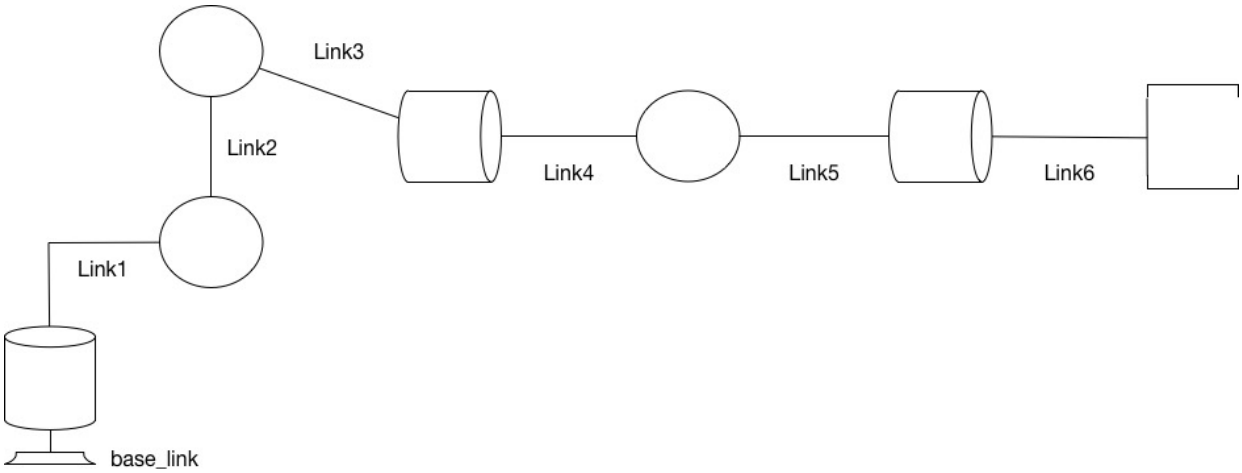
RoboND-Kinematics-Project-writeup

Kinematic Analysis

1. Run the forward_kinematics demo and evaluate the kr210.urdf.xacro file to perform kinematic analysis of Kuka KR210 robot and derive its DH parameters.





alpha = arm twist angle

a = arm link length

d = arm link offset

theta = arm joint angle

joint	alpha	a	d	theta
1	0	0	0.75	q1
2	-p2/2	0.35	0	q2-pi/2
3	0	1.25	0	q3
4	-pi/2	-0.054	1.50	q4
5	pi/2	0	0	q5
6	-pi/2	0	0	q6
gripper	0	0	0.303	0

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.

Standard Homogenous Transformation matrix from frame i-1 to frame i using DH Parameters:

$$T_{i-1}^{i} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & a_{i-1} \\ sin(\theta_i)cos(\alpha_{i-1}) & cos(\theta_i)cos(\alpha_{i-1}) & -sin(\alpha_{i-1}) & sin(\alpha_{i-1})d_i \\ sin(\theta_i)sin(\alpha_{i-1}) & cos(\theta_i)sin(\alpha_{i-1}) & cos(\alpha_{i-1}) & cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

we get the following transformation matrices about each joint with respect to the previous joint:

$T_0^1 =$

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

$T_1^2 =$

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

$T_2^3 =$

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

$T_3^4 =$

cos(q4)	**-sin(q4)**	**0**	**-0.054**
0	0	1	1.5
-sin(q4)	-cos(q4)	0	0
0	0	0	1

$T_4^5 =$

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

$T_5^6 =$

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

$T_6^G =$

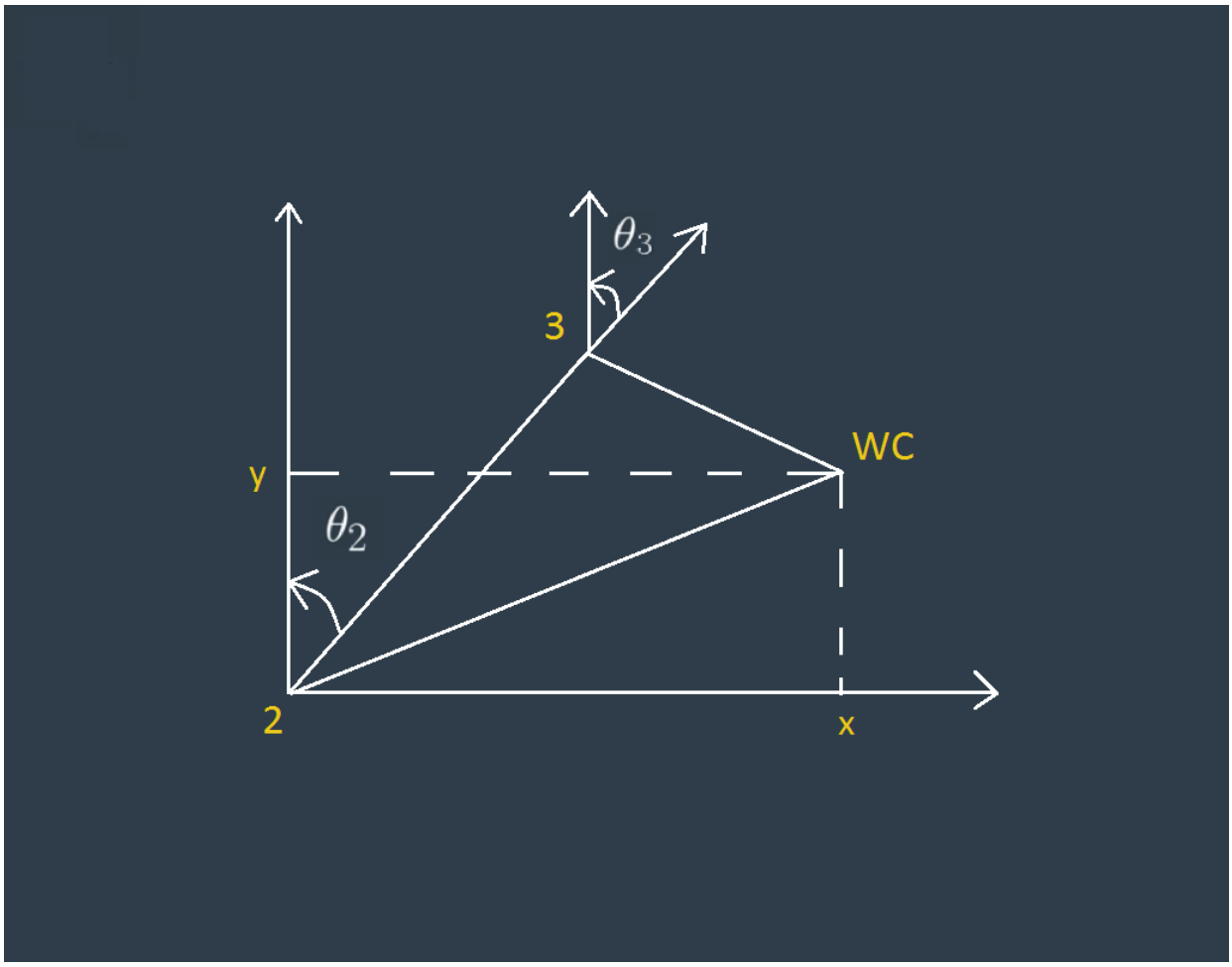
1	**0**	**0**	**0**
0	1	0	0
0	0	1	0.303
0	0	0	1

Generalized homogeneous transform:

$T_0^G = T_0^1 \cdot T_1^2 \cdot T_2^3 \cdot T_3^4 \cdot T_4^5 \cdot T_5^6 \cdot T_6^G$

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

And here's where you can draw out and show your math for the derivation of your theta angles.



Rotation matrices

ROLL

$$ROT_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(r) & -sin(r) \\ 0 & sin(r) & cos(r) \end{bmatrix}$$

PITCH

$$ROT_y = \begin{bmatrix} (cos(p) & 0 & sin(p) \\ 0 & 1 & 0 \\ -sin(p) & 0 & cos(p) \end{bmatrix}$$

YAW

$$ROT_z = \begin{bmatrix} (cos(y) & -sin(y) & 0 \\ sin(y) & cos(y) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$ROT_{EE} = ROT_z \cdot ROT_y \cdot ROT_x$$

180^o clockwise rotation and 90^o clockwise rotation

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

Extract end-effector position and orientation from request. But since roll, pitch, and yaw values for the gripper are returned in quaternions, we can use the transformations.py module from the TF package. Now that we can know WC.

```
ROT_EE = ROT_EE.subs({'r': roll, 'p': pitch, 'y': yaw})
EE = Matrix([[px], [py], [pz]])
WC = EE - (0.303) * ROT_EE[:,2]
```

$\theta_1 = atan2(WC_y, WC_x)$

$\theta_2 = \frac{\pi}{2} - A - [atan2(W \cdot C_z - 0.75, \sqrt{(W \cdot C_x^2 + W \cdot C_y^2 - 0.35)}]$

$\theta_3 = \frac{\pi}{2} - [B + 0.036]$

Calculate joint angles using Geomatric IK method

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

Inverse Orientation problems

$R_6^0 = R_1^0 \cdot R_2^1 \cdot R_3^2 \cdot R_4^3 \cdot R_5^4 \cdot R_6^5$

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 = ROT_EE
```

We can substitute the values we calculated for joints 1 to 3 in their respective individual rotation matrices and pre-multiply both sides of the above equation by inv(R0_3) which leads to:

```
R3_6 = inv(R0_3) * ROT_EE
```

The matrix on the RHS (Right Hand Side of the equation) does not have any variables after substituting the joint angle values, and hence comparing LHS (Left Hand Side of the equation) with RHS will result in equations for joint 4, 5, and 6.

```
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

theta4 = atan2(R3_6[2,2], -R3_6[0,2])
theta5 = atan2(sqrt(R3_6[0,2]*R3_6[0,2] + R3_6[2,2]*R3_6[2,2]), R3_6[1,2])
theta6 = atan2(-R3_6[1,1], R3_6[1,0])
```

Project Implementation

1. Fill in the `IK_server.py` file with properly commented python code for calculating Inverse Kinematics based on previously performed Kinematic Analysis. Your code must guide the robot to successfully complete 8/10 pick and place cycles. Briefly discuss the code you implemented and your results.

Creates Homogeneous Transform Matrix from DH parameters

```
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
```

Define DH Parameters:

```
d1, d2, d3, d4, d5, d6, d7 = symbols('d1:8')
a0, a1, a2, a3, a4, a5, a6 = symbols('a0:7')
alpha0, alpha1, alpha2, alpha3, alpha4, alpha5, alpha6 = symbols('alpha0:7')
q1, q2, q3, q4, q5, q6, q7 = symbols('q1:8')
```

Define DH Transformation Matrix

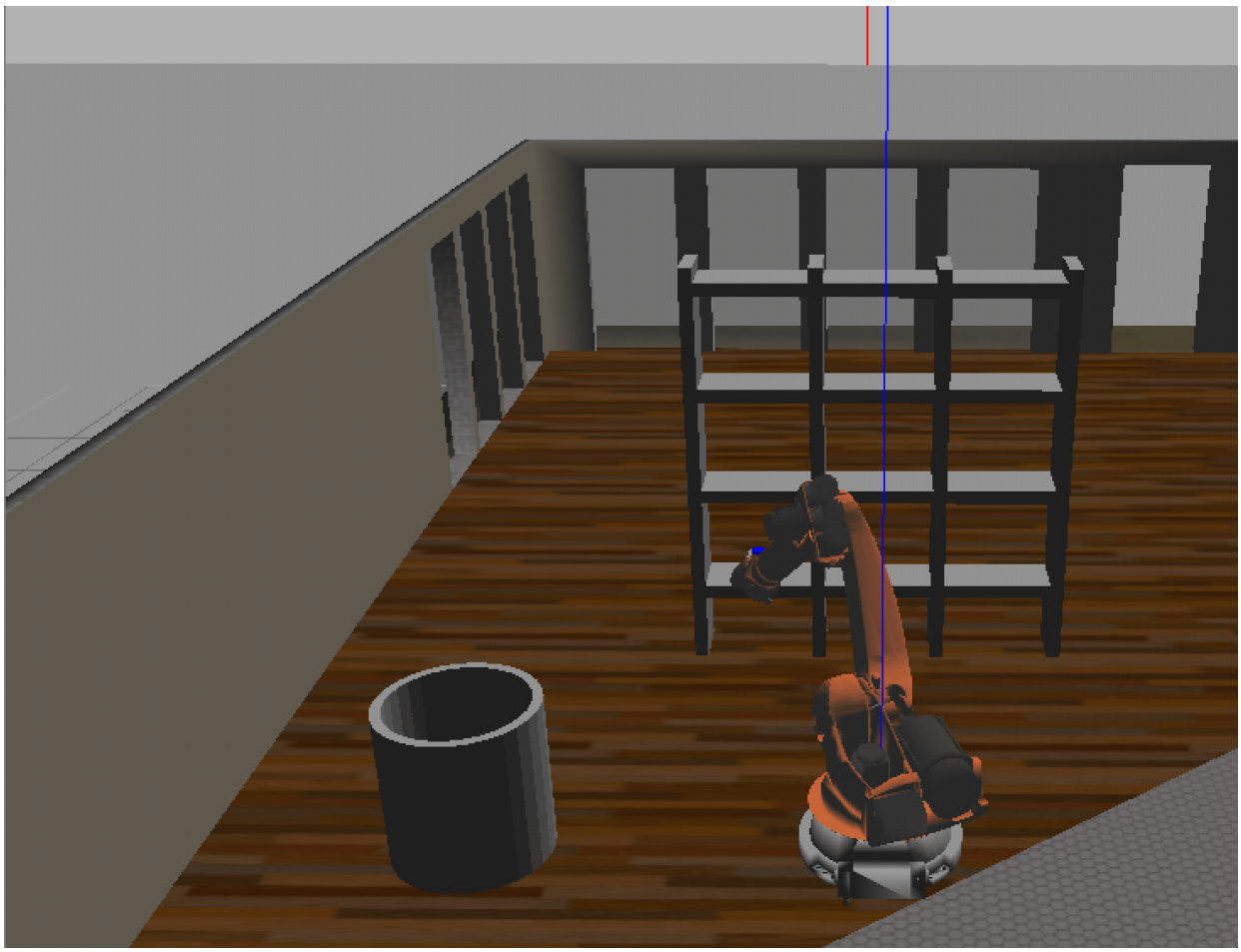
```
DH_Table = {alpha0: 0,  a0: 0,      d1: 0.75,   q1: q1,
            alpha1: -pi/2., a1: 0.35,   d2: 0,      q2: -pi/2. + q2,
            alpha2: 0,  a2: 1.25,   d3: 0,      q3: q3,
            alpha3: -pi/2., a3: -0.054,     d4: 1.5,    q4: q4,
            alpha4: pi/2,   a4: 0,      d5: 0,      q5: q5,
            alpha5: -pi/2., a5: 0,      d6: 0,      q6: q6,
            alpha6: 0,  a6: 0,      d7: 0.303,  q7: 0}
```

Generalized homogeneous transform

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

move to blue cylindrical target

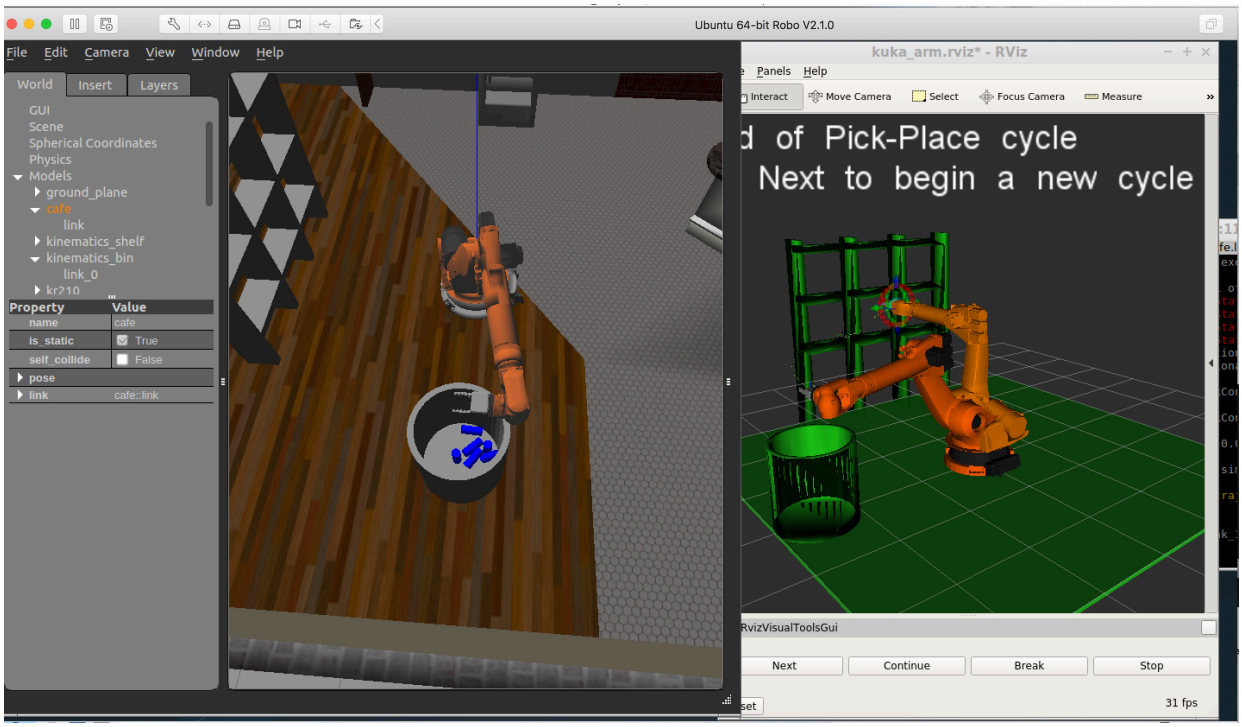
pick the target



move to drop location

release the target

Shot of bin with 10 samples



This is my first time to use rviz and Gazebo, I found they are extremely challenging to debug. My code may not be efficient enough, I can introduce LU decomposition in finding the inverse of matrix.