Yang Ding
Project 2

Kinematics Analysis:

DH parameters

Figure 1 shows the layout of the KR210 robotic arm in a skeleton view. Coordinate system 0 to 7 are labeled. The triangle used to calculate the rotation angle of joint 2 and 3 is marked, with three sides a, b and c labeled, which have the corresponding angle A, B and C. A typed DH parameter table is provided in table 1.

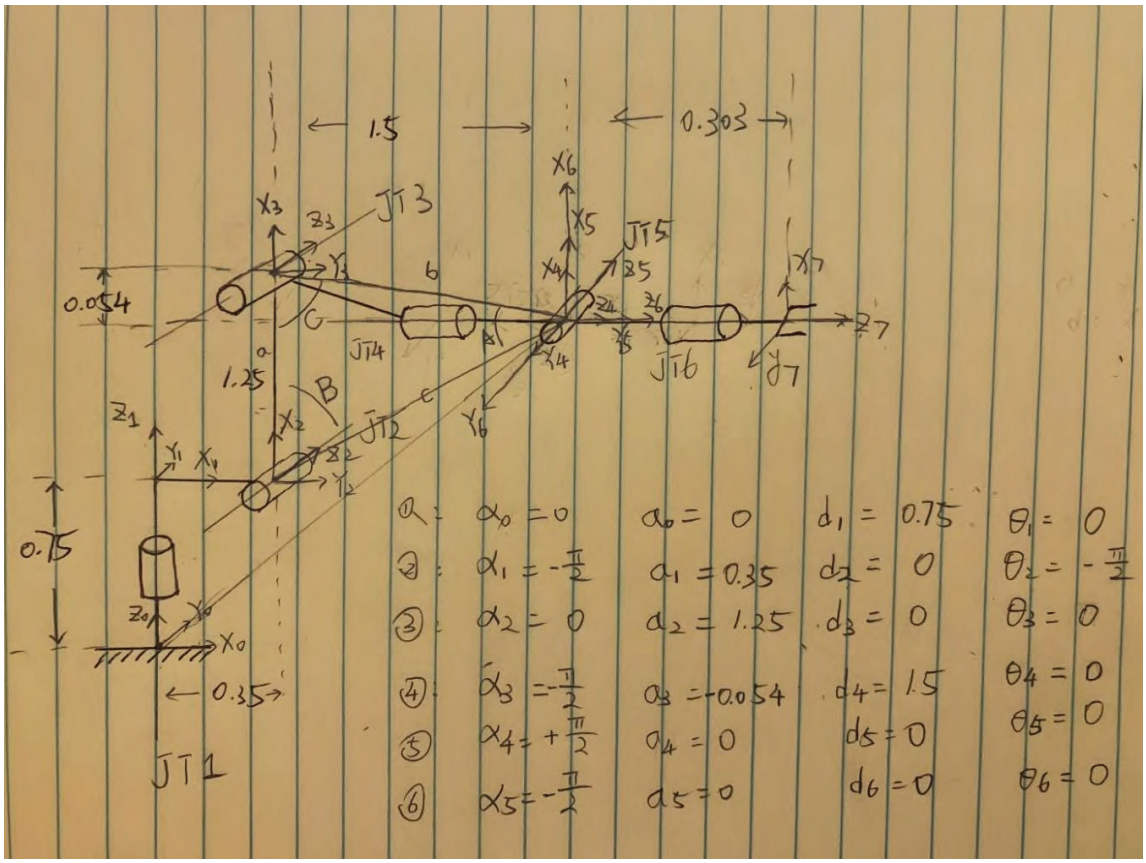


Figure 1. Skeleton View of KR210

From 0 to 1	$\alpha_0 = 0$	$a_0 = 0$	$d_1 = 0.75$	$\theta_1 = q_1$
From 1 to 2	$\alpha_1 = -\frac{\pi}{2}$	$a_1 = 0.35$	$d_2 = 0$	$\theta_2 = q_2 - \frac{\pi}{2}$
From 2 to 3	$\alpha_2 = 0$	$a_2 = 1.25$	$d_3 = 0$	$\theta_3 = q_3$
From 3 to 4	$\alpha_3 = -\frac{\pi}{2}$	$a_3 = -0.054$	$d_4 = 1.5$	$\theta_4 = q_4$
From 4 to 5	$\alpha_4 = \frac{\pi}{2}$	$a_4 = 0$	$d_5 = 0$	$\theta_5 = q_5$
From 5 to 6	$\alpha_5 = -\frac{\pi}{2}$	$a_5 = 0$	$d_6 = 0$	$\theta_6 = q_6$
From 6 to 7	$\alpha_6 = 0$	$a_6 = 0$	$d_7 = 0.303$	$\theta_7 = 0$

Table 1. DH parameters

Yang Ding
Project 2

Homogeneous Transformation (Forward Kinematics)

Using the format, the transformation for each joint is achieved.

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The transformation matrix for each joint (1 to 6), and T67 is the transformation from frame 6 to the end effector, t1…t6 stand for θ_1 …θ_6

```
T01 =

[ cos(t1), -sin(t1), 0,   0]
[ sin(t1),  cos(t1), 0,   0]
[       0,        0, 1, 3/4]
[       0,        0, 0,   1]

T12 =

[  cos(t2 - pi/2), -sin(t2 - pi/2), 0, 7/20]
[               0,               0, 1,    0]
[ -sin(t2 - pi/2), -cos(t2 - pi/2), 0,    0]
[               0,               0, 0,    1]

T23 =

[ cos(t3), -sin(t3), 0, 5/4]
[ sin(t3),  cos(t3), 0,   0]
[       0,        0, 1,   0]
[       0,        0, 0,   1]
```

```
T34 =

[  cos(t4), -sin(t4), 0, -27/500]
[        0,        0, 1,     3/2]
[ -sin(t4), -cos(t4), 0,       0]
[        0,        0, 0,       1]

T45 =

[ cos(t5), -sin(t5),  0, 0]
[       0,        0, -1, 0]
[ sin(t5),  cos(t5),  0, 0]
[       0,        0,  0, 1]

T56 =

[  cos(t6), -sin(t6), 0, 0]
[        0,        0, 1, 0]
[ -sin(t6), -cos(t6), 0, 0]
[        0,        0, 0, 1]

T67 =

    1.0000         0         0         0
         0    1.0000         0         0
         0         0    1.0000    0.3030
         0         0         0    1.0000
```

Multiplying them in order:

$$T_{0\_EE} = T_{01} \cdot T_{12} \cdot T_{23} \cdot T_{34} \cdot T_{45} \cdot T_{56} \cdot T_{67}$$

The result is complex and lengthy, and it is shown in the next page: each row is marked in different color.

Yang Ding
Project 2

T0_EE =

[cos(t6)*(cos(t5)*sin(t1)*sin(t4) + cos(t2 + t3)*cos(t1)*sin(t5) + cos(t1)*cos(t2)*cos(t4)*cos(t5)*sin(t3) + cos(t1)*cos(t3)*cos(t4)*cos(t5)*sin(t2)) - sin(t6)*(cos(t1)*cos(t2)*sin(t3)*sin(t4) - cos(t4)*sin(t1) + cos(t1)*cos(t3)*sin(t2)*sin(t4))

- cos(t6)*(cos(t1)*cos(t2)*sin(t3)*sin(t4) - cos(t4)*sin(t1) + cos(t1)*cos(t3)*sin(t2)*sin(t4)) - sin(t6)*(cos(t5)*sin(t1)*sin(t4) + cos(t2 + t3)*cos(t1)*sin(t5) + cos(t1)*cos(t2)*cos(t4)*cos(t5)*sin(t3) + cos(t1)*cos(t3)*cos(t4)*cos(t5)*sin(t2))

cos(t2 + t3)*cos(t1)*cos(t5) - sin(t5)*(sin(t1)*sin(t4) + cos(t1)*cos(t2)*cos(t4)*sin(t3) + cos(t1)*cos(t3)*cos(t4)*sin(t2))

(7*cos(t1))/20 + (5*cos(t1)*sin(t2))/4 - (3*cos(t1)*sin(t2)*sin(t3))/2 - (303*sin(t1)*sin(t4)*sin(t5))/1000 + (3*cos(t1)*cos(t2)*cos(t3))/2 - (27*cos(t1)*cos(t2)*sin(t3))/500 - (27*cos(t1)*cos(t3)*sin(t2))/500 + (303*cos(t1)*cos(t2)*cos(t3)*cos(t5))/1000 - (303*cos(t1)*cos(t5)*sin(t2)*sin(t3))/1000 - (303*cos(t1)*cos(t2)*cos(t4)*sin(t3)*sin(t5))/1000 - (303*cos(t1)*cos(t3)*cos(t4)*sin(t2)*sin(t5))/1000]

[cos(t6)*(cos(t2 + t3)*sin(t1)*sin(t5) - cos(t1)*cos(t5)*sin(t4) + cos(t2)*cos(t4)*cos(t5)*sin(t1)*sin(t3) + cos(t3)*cos(t4)*cos(t5)*sin(t1)*sin(t2)) - sin(t6)*(cos(t1)*cos(t4) + cos(t2)*sin(t1)*sin(t3)*sin(t4) + cos(t3)*sin(t1)*sin(t2)*sin(t4))

- cos(t6)*(cos(t1)*cos(t4) + cos(t2)*sin(t1)*sin(t3)*sin(t4) + cos(t3)*sin(t1)*sin(t2)*sin(t4)) - sin(t6)*(cos(t2 + t3)*sin(t1)*sin(t5) - cos(t1)*cos(t5)*sin(t4) + cos(t2)*cos(t4)*cos(t5)*sin(t1)*sin(t3) + cos(t3)*cos(t4)*cos(t5)*sin(t1)*sin(t2))

cos(t2 + t3)*cos(t5)*sin(t1) - sin(t5)*(cos(t2)*cos(t4)*sin(t1)*sin(t3) - cos(t1)*sin(t4) + cos(t3)*cos(t4)*sin(t1)*sin(t2))

(7*sin(t1))/20 + (5*sin(t1)*sin(t2))/4 - (27*cos(t2)*sin(t1)*sin(t3))/500 - (27*cos(t3)*sin(t1)*sin(t2))/500 + (303*cos(t1)*sin(t4)*sin(t5))/1000 - (3*sin(t1)*sin(t2)*sin(t3))/2 + (3*cos(t2)*cos(t3)*sin(t1))/2 + (303*cos(t2)*cos(t3)*cos(t5)*sin(t1))/1000 - (303*cos(t5)*sin(t1)*sin(t2)*sin(t3))/1000 - (303*cos(t2)*cos(t4)*sin(t1)*sin(t3)*sin(t5))/1000 - (303*cos(t3)*cos(t4)*sin(t1)*sin(t2)*sin(t5))/1000]

[- cos(t6)*(sin(t2 + t3)*sin(t5) - cos(t2 + t3)*cos(t4)*cos(t5)) - cos(t2 + t3)*sin(t4)*sin(t6)

sin(t6)*(sin(t2 + t3)*sin(t5) - cos(t2 + t3)*cos(t4)*cos(t5)) - cos(t2 + t3)*cos(t6)*sin(t4)

- sin(t2 + t3)*cos(t5) - cos(t2 + t3)*cos(t4)*sin(t5)

(5*cos(t2))/4 - (27*cos(t2)*cos(t3))/500 - (3*cos(t2)*sin(t3))/2 - (3*cos(t3)*sin(t2))/2 + (27*sin(t2)*sin(t3))/500 - (303*cos(t2)*cos(t5)*sin(t3))/1000 - (303*cos(t3)*cos(t5)*sin(t2))/1000 - (303*cos(t2)*cos(t3)*cos(t4)*sin(t5))/1000 + (303*cos(t4)*sin(t2)*sin(t3)*sin(t5))/1000 + 3/4]

[0, 0, 0, 1]

Yang Ding
Project 2

Given the orientation and position of the end gripper, the overall homogeneous transformation can be achieved in the following steps.

1. With the Euler angle or the quaternions, the rotation matrix of the end effector with respect to the end effector frame can be calculated as following. (y: yaw, p: pitch, r: roll)

$$R_{EE} = R_z \cdot R_y \cdot R_x = \begin{bmatrix} \cos y & -\sin y & 0 \\ \sin y & \cos y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos p & 0 & \sin p \\ 0 & 1 & 0 \\ -\sin p & 0 & \cos p \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos r & -\sin r \\ 0 & \sin r & \cos r \end{bmatrix}$$

2. Since there is difference in the coordinate system 0 and the end effector coordinate, we should correct this discrepancy using a rotation matrix given below.

$$R_{corr} = R_{01} \cdot R_{12} \cdot R_{23} \cdot R_{34} \cdot R_{45} \cdot R_{56} \cdot R_{67} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

3. Then the overall rotation matrix with respect to the ground frame 0 can be calculated as below.

$$R_{0\_EE} = R_{EE} \cdot R_{corr}$$

4. Combining the rotation matrix with the translation part, which is the position information of the end effector, we can get the overall homogeneous transformation in the following format.

$$T_{0\_EE} = \begin{bmatrix} & & & P_x \\ & R_{0\_EE} & & P_y \\ & & & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse Kinematics

There are two parts in this section: the first part is to calculate θ_1, θ_2 and θ_3 based on the desired location of the wrist center, and the second part is to calculate θ_4, θ_5 and θ_6 based on the desired orientation of the end effector.

1. Part 1

The wrist center location can be calculated using the following equation.

$$WC = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} - R_{0\_EE} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \cdot 0.303$$

After achieving the wrist center location, the rotation of the first joint is obvious: it can be calculated using the x and y components.

$$\theta_1 = atan2(WC(2), WC(1))$$

To calculate θ_2 and θ_3, the cosine law is applied to the triangle that has its vertices at joint 2, 3 and 5.

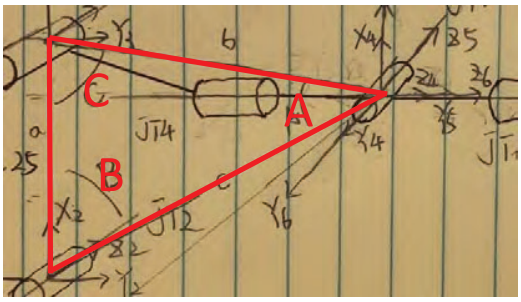


Figure 2. Triangles used to calculate θ_2 and θ_3

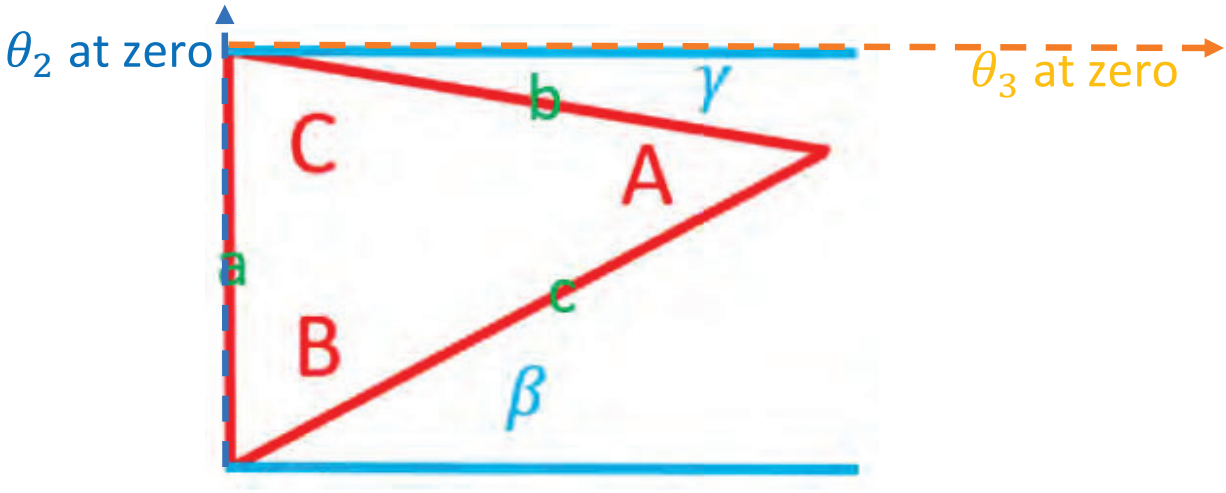


Figure 3. Simplified geometry

In this triangle, side a and side b stay fixed due to the geometry constrain.

$$\mathrm{a} = 1.25$$

$$\mathrm{b} = \sqrt{1.5^2 + 0.453^2} = 1.501$$

Side c is varying with the joint variables, and the following steps are applied to calculate c

1. Calculate the length of the projection of the wrist center vector from original onto the x-y plane.

$$\mathrm{Proj(WC)_{xy}} = \sqrt{WC(1)^2 + WC(2)^2}$$

2. Then the projection length should subtract the offset 0.35 to get the projection of side c onto the x-y plane

$$\mathrm{Proj(c)_{xy}} = \mathrm{Proj(WC)_{xy}} - 0.35$$

3. Applying Pythagorean theorem, the length of c can be calculated.

$$c = \sqrt{\left(Proj(c)_{xy}\right)^2 + (WC(3) - 0.75)^2}$$

Using a, b and c, the angles of the triangle can be calculated using Cosine Law.

$$\angle A = \mathrm{acos}((a^2 - b^2 - c^2)/(-2bc))$$

$$\angle B = \mathrm{acos}((b^2 - a^2 - c^2)/(-2ac))$$

$$\angle C = \mathrm{acos}((c^2 - b^2 - a^2)/(-2ba))$$

Then it is obvious from figure 3 that

$$\theta_2 = \frac{\pi}{2} - B - \beta$$

$$\theta_3 = \frac{\pi}{2} - C - \gamma$$

$$\beta = atan2(WC(3) - 0.75, Proj(c)_{xy})$$

$$\gamma = atan2(0.054, 1.5) = 0.036$$

2. Part 2

After the first three joint angles are determined, we can apply the forward kinematics to calculate the rotation matrix from the ground frame 0 to the frame 3.

$$R_{0\_3} = R_{0\_1} \cdot R_{1\_2} \cdot R_{2\_3}$$

Using the pre-calculated overall rotation matrix from the ground frame to the end effector, we can calculate the rotation matrix between the third frame to the end effector.

$$R_{3\_EE} = \left(R_{0\_3}\right)^T \cdot R_{0\_EE}$$

=

```
[   cos(t4)*cos(t5)*cos(t6) - sin(t4)*sin(t6), - cos(t6)*sin(t4) - cos(t4)*cos(t5)*sin(t6), -cos(t4)*sin(t5)]
[                           cos(t6)*sin(t5),                            -sin(t5)*sin(t6),          cos(t5)]
[ - cos(t4)*sin(t6) - cos(t5)*cos(t6)*sin(t4),   cos(t5)*sin(t4)*sin(t6) - cos(t4)*cos(t6),  sin(t4)*sin(t5)]
```

From the above rotation matrix, we can calculate θ_4, θ_5 and θ_6

$$-\tan\theta_4 = \frac{\sin\theta_4 \sin\theta_5}{-\cos\theta_4 \sin\theta_5} = \frac{R_{3\_EE}(3,3)}{R_{3\_EE}(1,3)}$$

$$\tan\theta_4 = \frac{\frac{R_{3\_EE}(3,3)}{R_{3\_EE}(1,3)}}{-1}$$

$$\theta_4 = atan2\left(R_{3\_EE}(3,3), -R_{3\_EE}(1,3)\right)$$

$$\frac{\sqrt{R_{3\_EE}(1,3)^2 + R_{3\_EE}(3,3)^2}}{R_{3\_EE}(2,3)} = \frac{\sin\theta_5}{\cos\theta_5} = \tan\theta_5$$

$$\theta_5 = atan2\left(\sqrt{R_{3\_EE}(1,3)^2 + R_{3\_EE}(3,3)^2},\ R_{3\_EE}(2,3)\right)$$

$$-\tan\theta_6 = \frac{-\sin\theta_6 \sin\theta_5}{\cos\theta_6 \sin\theta_5} = \frac{R_{3\_EE}(2,2)}{R_{3\_EE}(2,1)}$$

$$\tan\theta_6 = \frac{\frac{-R_{3\_EE}(2,2)}{R_{3\_EE}(2,1)}}{1}$$

$$\theta_6 = atan2\left(-R_{3\_EE}(2,2), R_{3\_EE}(2,1)\right)$$

Yang Ding
Project 2

Project Implementation

1. Set up and import the required packages

```
#!/usr/bin/env python

# Copyright (C) 2017 Udacity Inc.
# This file is part of Robotic Arm: Pick and Place project for Udacity
# Robotics nano-degree program
# All Rights Reserved.
# Author: Harsh Pandya

# import modules
import rospy
import tf
from kuka_arm.srv import *
from trajectory_msgs.msg import JointTrajectory, JointTrajectoryPoint
from geometry_msgs.msg import Pose
from sympy import *
```

2. Create a sub-function to generate Homogeneous Transformation, alpha is the link twist angle, a is the link length, d is the link offset and q is the joint variable.

```
def DH_transform(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
```

3. This is the sub-function used to calculate the joint variable for the six joints. The first step is to see if there is any data to be processed. If not, print an error and return -1

```
# IK handler service
def handle_calculate_IK(req):
    rospy.loginfo("Received %s eef-poses from the plan" % len(req.poses))
    if len(req.poses) < 1:
        print "No valid poses received"
        return -1
```

4. If there is data, first initialize the DH table using symbols.

```
30      else:
31          q1, q2, q3, q4, q5, q6, q7 = symbols('q1:8')
32          d1, d2, d3, d4, d5, d6, d7 = symbols('d1:8')
33          a0, a1, a2, a3, a4, a5, a6 = symbols('a0:7')
34          alpha0, alpha1, alpha2, alpha3, alpha4, alpha5, alpha6 = symbols('alpha0:7')
35
36          dh = {alpha0:      0, a0:      0, d1:  0.75, q1:        q1,
37                alpha1: -pi/2., a1:   0.35, d2:     0, q2: -pi/2.+q2,
38                alpha2:      0, a2:   1.25, d3:     0, q3:        q3,
39                alpha3: -pi/2., a3: -0.054, d4:   1.5, q4:        q4,
40                alpha4:  pi/2., a4:      0, d5:     0, q5:        q5,
41                alpha5: -pi/2., a5:      0, d6:     0, q6:        q6,
42                alpha6:      0, a6:      0, d7: 0.303, q7:         0}
```

5. Then call *DH_transform* function to perform homogeneous transform to get the transformation matrix between each two links including the ground frame and the end effector.

```
44          # Substitute DH_Table
45          T01 = DH_transform(alpha0, a0, d1, q1).subs(dh)
46          T12 = DH_transform(alpha1, a1, d2, q2).subs(dh)
47          T23 = DH_transform(alpha2, a2, d3, q3).subs(dh)
48          T34 = DH_transform(alpha3, a3, d4, q4).subs(dh)
49          T45 = DH_transform(alpha4, a4, d5, q5).subs(dh)
50          T56 = DH_transform(alpha5, a5, d6, q6).subs(dh)
51          T67 = DH_transform(alpha6, a6, d7, q7).subs(dh)
```

6. Generating two transformation matrices. 1: transformation between the ground frame and the frame 3; 2: transformation between the ground frame and the frame 7.

```
53          # Transform from Base link_0 to end effector (Gripper) Link_7
54          T03 = T01 * T12 * T23
55          T07 = T01 * T12 * T23 * T34 * T45 * T56 * T67
```

7. Initialize the rotation matrix about x, y and z respectively using symbols, and multiply them in z-y-x order to get the rotation matrix of Euler Angle combination.

```
60          # Find EE rotation matrix RPY (Roll, Pitch, Yaw)
61          r,p,y = symbols('r p y')
62
63          # Create rotation matrix
64          Rx = Matrix([[       1,       0,       0],
65                       [       0,  cos(r), -sin(r)],
66                       [       0,  sin(r),  cos(r)]])
67          # Pitch
68          Ry = Matrix([[  cos(p),       0,  sin(p)],
69                       [       0,       1,       0],
70                       [ -sin(p),       0,  cos(p)]])
71          # Yaw
72          Rz = Matrix([[  cos(y), -sin(y),       0],
73                       [  sin(y),  cos(y),       0],
74                       [       0,       0,       1]])
75
76          REE = Rz * Ry * Rx
```

8. Rcorr is the rotation matrix between the ground frame and the end effector, and this matrix is used to transform the perspective of the Euler rotation matrix from the end effector view to the ground view, so that the overall rotation matrix of the end effector can be achieved.

```
78          #Pre define the rotation matrix between the ground frame and the end effector frame
79          Rcorr = Matrix([[0, 0, 1],[0, -1, 0],[1, 0, 0]])
80          # Compensate for rotation discrepancy between DH parameters and Gazebo
81          REE = REE * Rcorr
```

9. After setting up everything symbolically, here begins the loop, in which the joint variables for each point in the planned path are calculated and then populated. The first step is to initialize the array to contain the resulting joint variables

```
83          # Initialize service response
84          joint_trajectory_list = []
85          for x in xrange(0, len(req.poses)):
86              # IK code starts here
87              joint_trajectory_point = JointTrajectoryPoint()
88
```

10 . Retrieve end effector xyz position and combine them into a column vector

```
89              # Extract end-effector position and orientation from request
90              # px,py,pz = end-effector position
91              px = req.poses[x].position.x
92              py = req.poses[x].position.y
93              pz = req.poses[x].position.z
94
95              # store EE position in a matrix
96              EE = Matrix([[px],
97                           [py],
98                           [pz]])
```

11 . Retrieve end effector orientation quaternion info and change it into the Euler vector

```
100             # Use the function to transform quaternion to Euler Angle
101             (roll,pitch,yaw) = tf.transformations.euler_from_quaternion(
102                 [req.poses[x].orientation.x,
103                  req.poses[x].orientation.y,
104                  req.poses[x].orientation.z,
105                  req.poses[x].orientation.w])
106
107             #Plug the value of roll, pitch and yaw into the sybolic rotation matrix
108             REE = REE.subs({'r': roll, 'p': pitch, 'y': yaw})
```

12 . Calculate the wrist center location with respect to the ground frame

```
110             # Calculate wrist Center
111             WC = EE - (0.303) * REE[:,2]
```

13 . Calculate the first joint variable, θ_1

```
113             # Calculate joint angles using Geometric IK method
114             # Calculate theta1
115             theta1 = atan2(WC[1],WC[0])
116
```

14 . Calculate the angles and side length of the triangle

```
117             #Calculate the three sides of the triangle
118             a = 1.25;
119             b = sqrt(1.5*1.5 + 0.054*0.054);
120             c = sqrt(pow((sqrt(WC[0]*WC[0] + WC[1]*WC[1]) - 0.35), 2) + pow((WC[2] - 0.75), 2))
121
122             #Calculate the angle of the vertices of the triangle
123             A = acos((a*a - b*b -c*c)/(-2*b*c));
124             B = acos((b*b - a*a -c*c)/(-2*a*c));
125             C = acos((c*c - a*a -b*b)/(-2*a*b));
```

15 . Calculate the second and third joint variables, θ_2 and θ_3

```
127             # Calculate theta2
128             theta2 = pi/2 - B - atan2(WC[2]-0.75, sqrt(WC[0]*WC[0]+WC[1]*WC[1])-0.35)
129             # Calculate theta3
130             theta3 = pi/2 - C - atan2(0.054,1.5);
```

16 . Calculate the rotation matrix for the last three joints

```
132             #Calculate the rotation matrix between frame 3 and the end effector frame
133             R03 = T03[0:3,0:3]
134             R03 = R03.evalf(subs={q1: theta1, q2: theta2, q3:theta3})
135             R36 = R03.transpose() * REE
136
```

17 . Calculate the last three joint variables, θ_4, θ_5 and θ_6

```
137             # Calculate theta5
138             theta5 = atan2(sqrt(R36[0,2]*R36[0,2] + R36[2,2]*R36[2,2]),R36[1,2])
139             theta4 = atan2(R36[2,2], -R36[0,2])
140             theta6 = atan2(-R36[1,1],R36[1,0])
```

18 . Populate the results

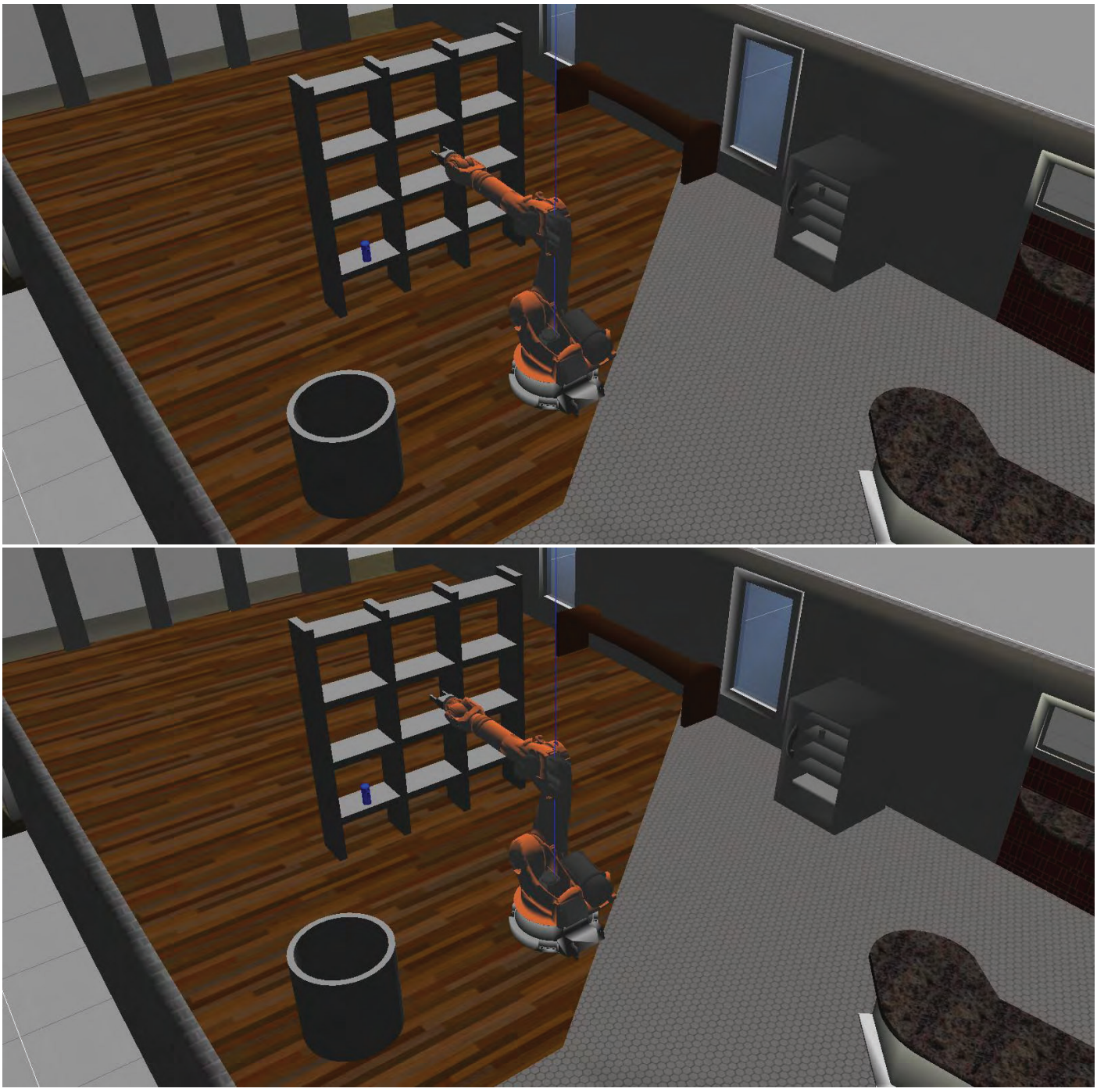
```
148             # Populate response for the IK request
149             # In the next line replace theta1,theta2...,theta6 by your joint angle variables
150             joint_trajectory_point.positions = [theta1, theta2, theta3, theta4, theta5, theta6]
151             joint_trajectory_list.append(joint_trajectory_point)
```

The result is not perfect. This code cannot let the robotic arm pick and drop the object located at the top right corner for some uncertain reason: it will drop the object during the half way. In addition, the last three joints tend to twist a lot along the path, and this could be due to the multiple solutions from the rotation matrix. After all, the code can give the correct solution to drive the arm in the right path for the slot 1 through 8.

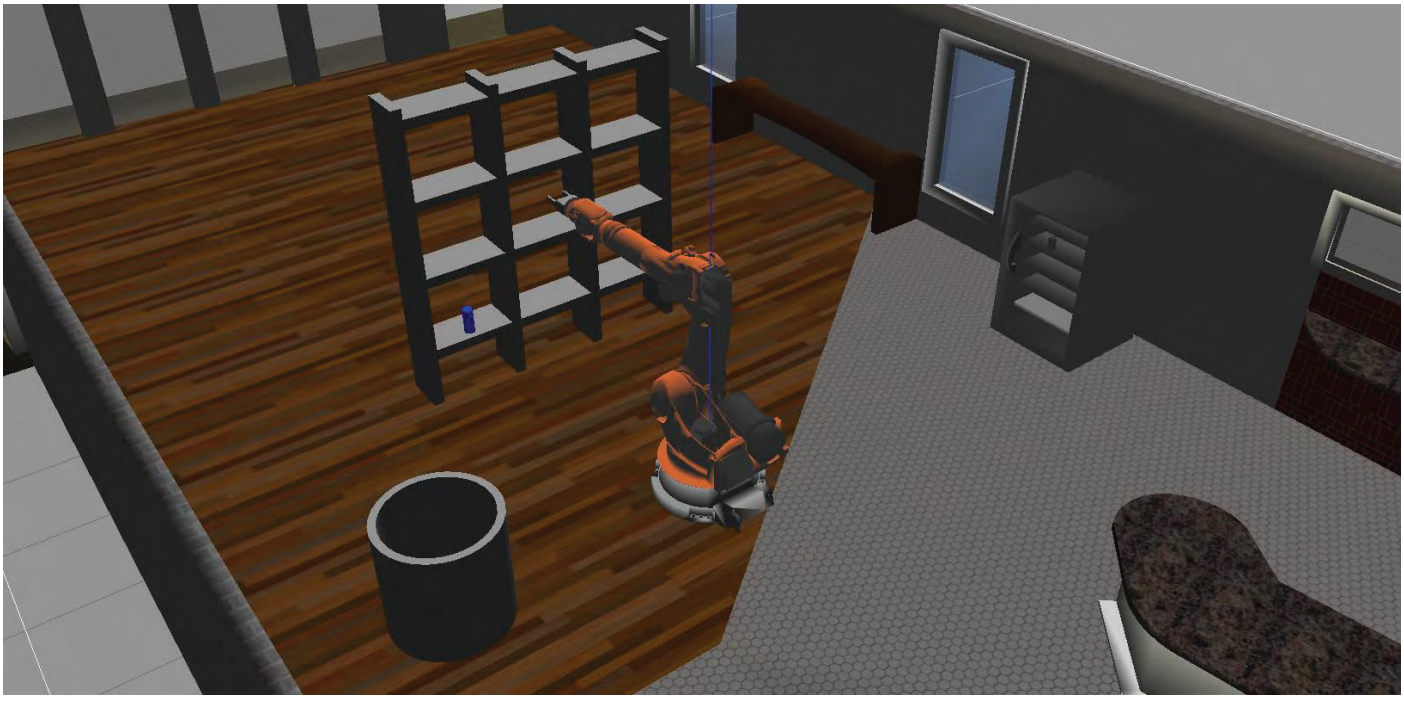
In the rest pages, 8 trials are presented. For each trial, there are sequential screen shots attached.

Yang Ding
Project 2

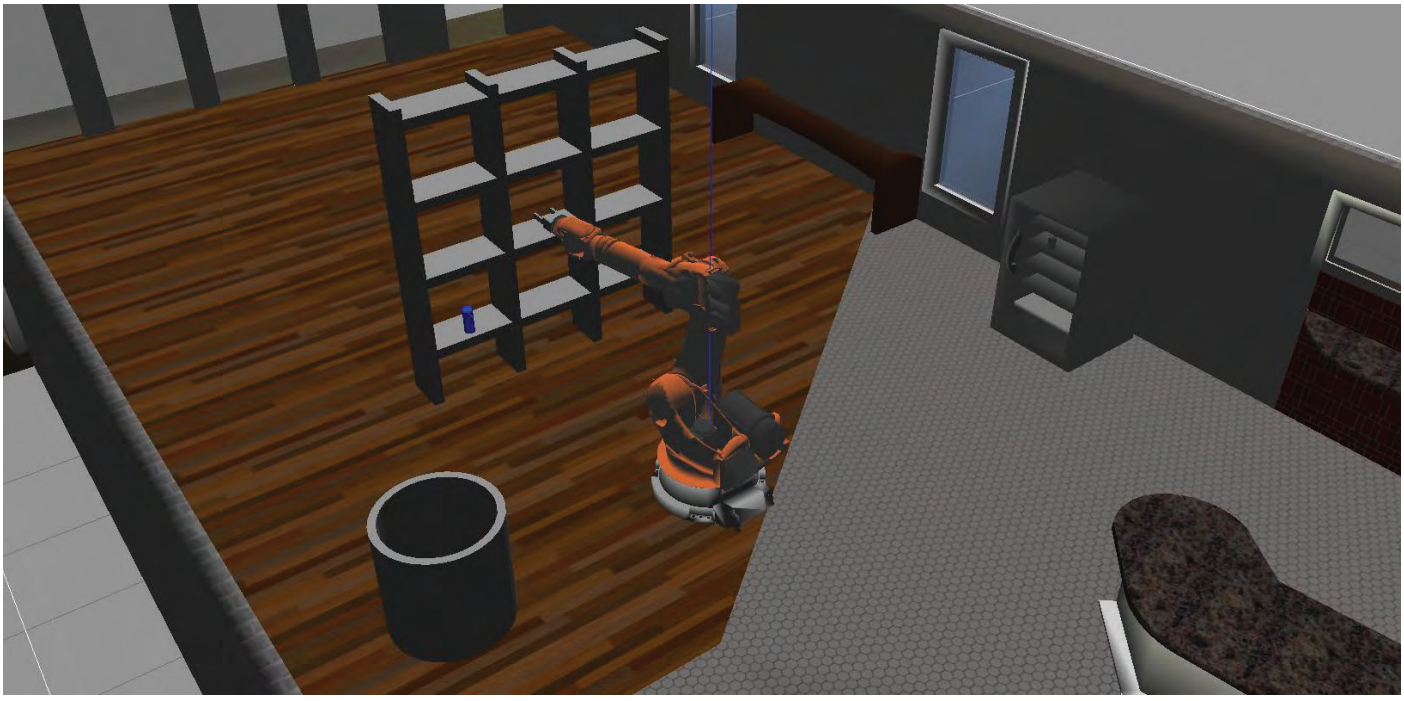
Slot 1



Yang Ding
Project 2



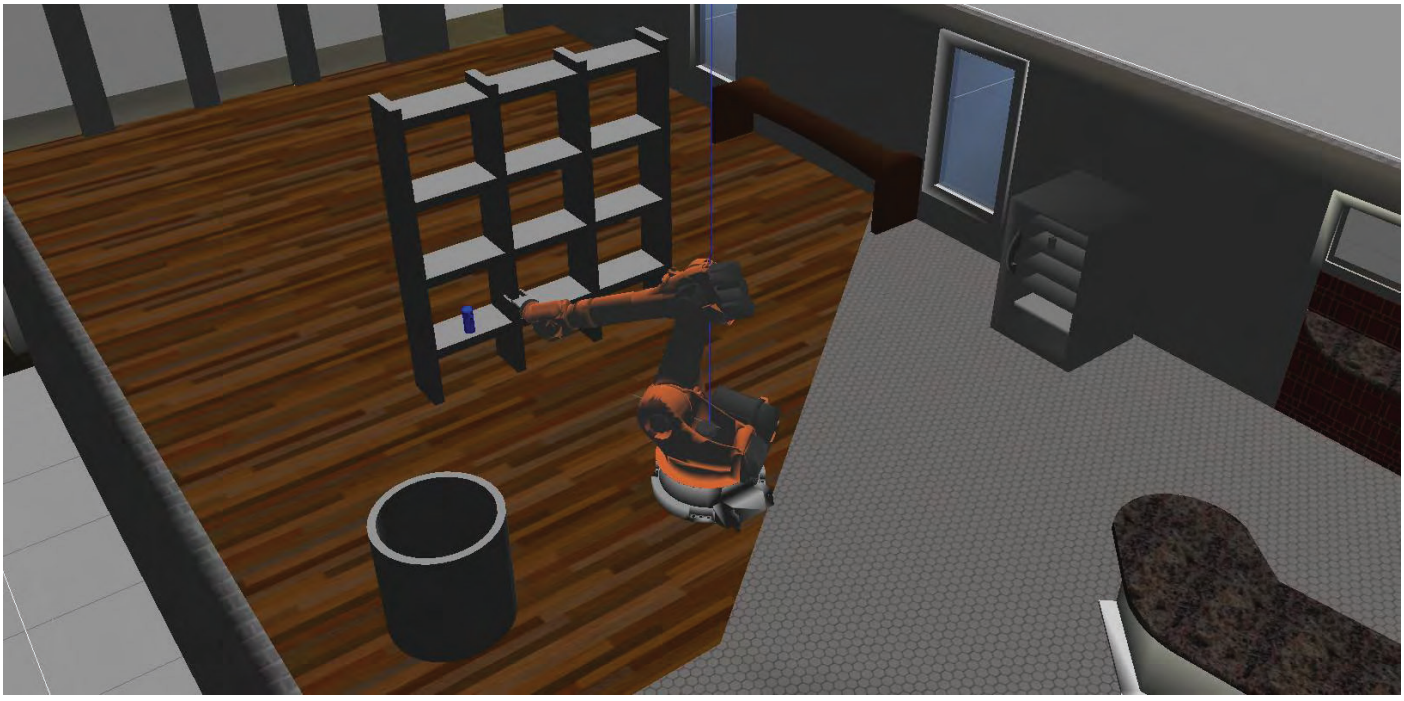
Yang Ding
Project 2



Yang Ding
Project 2

Yang Ding
Project 2

Yang Ding
Project 2



Yang Ding
Project 2

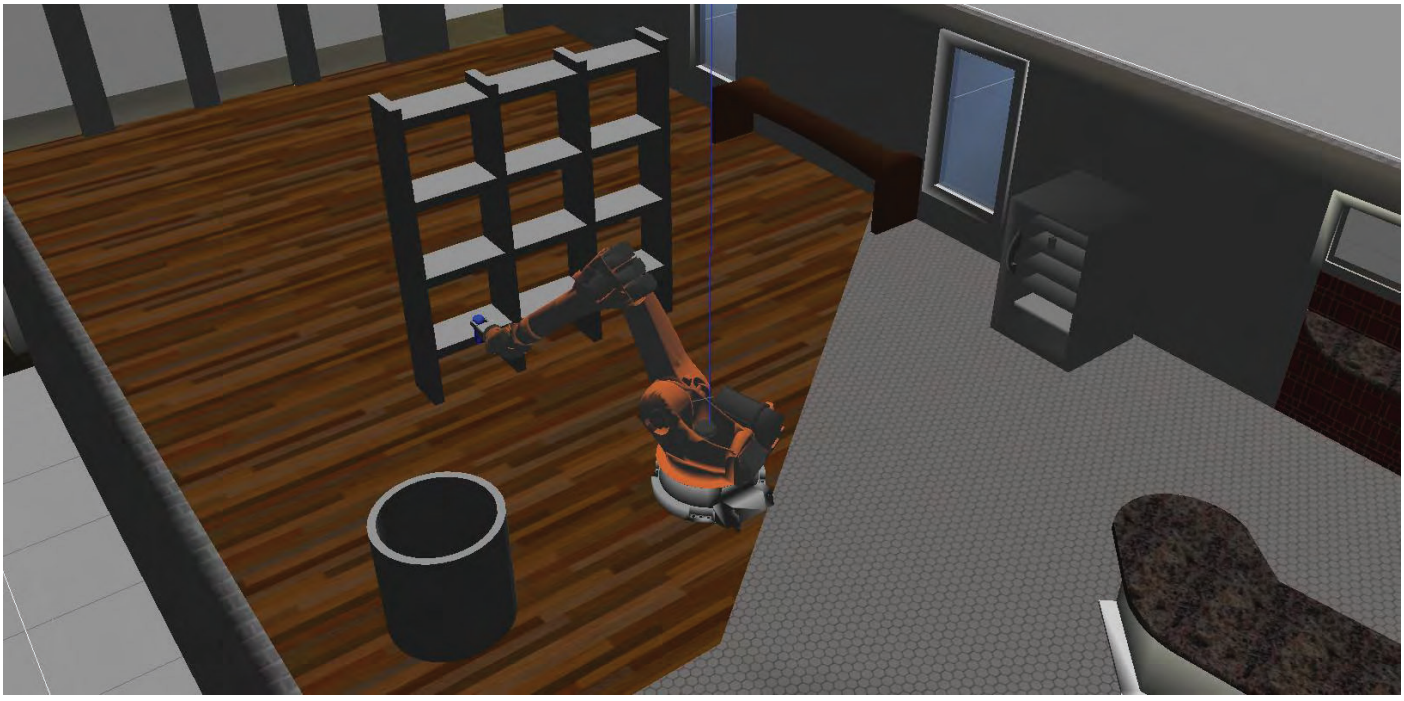
Yang Ding
Project 2

Yang Ding
Project 2

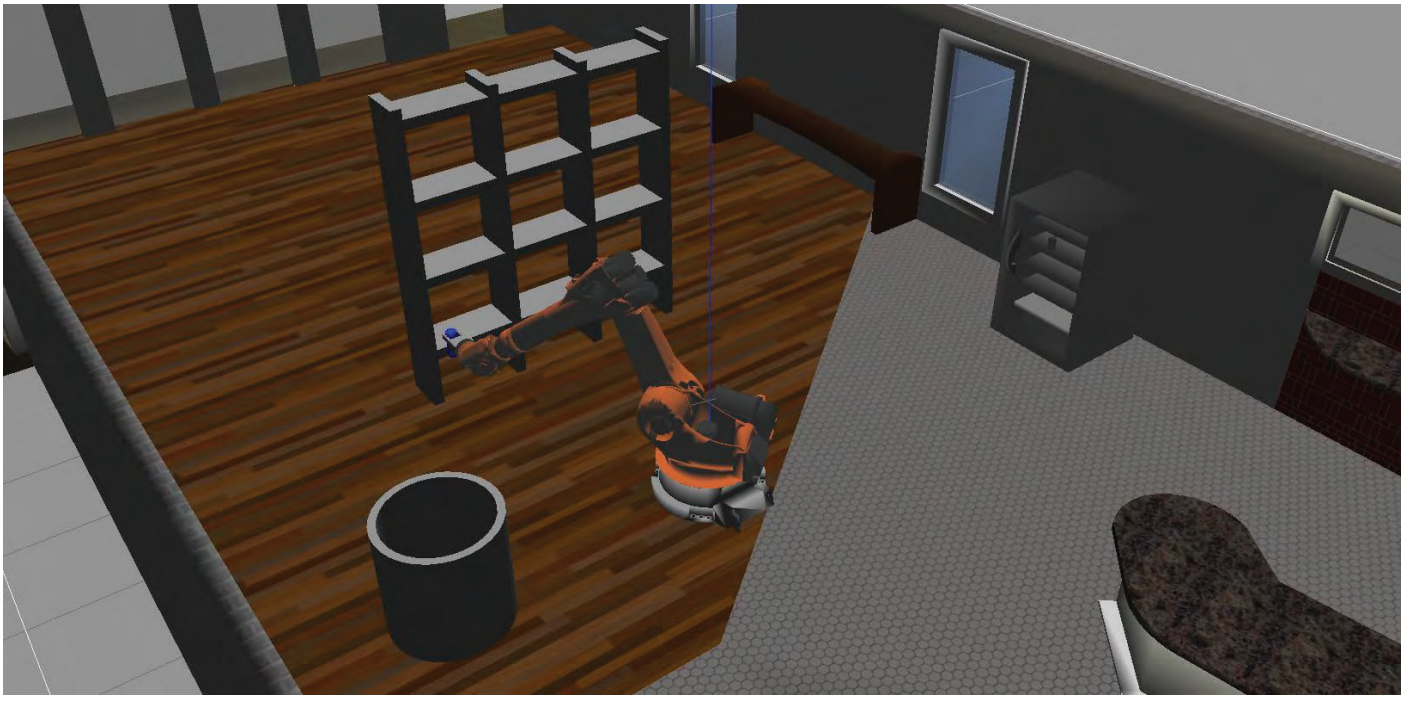
Yang Ding
Project 2

Yang Ding
Project 2

Yang Ding
Project 2



Yang Ding
Project 2

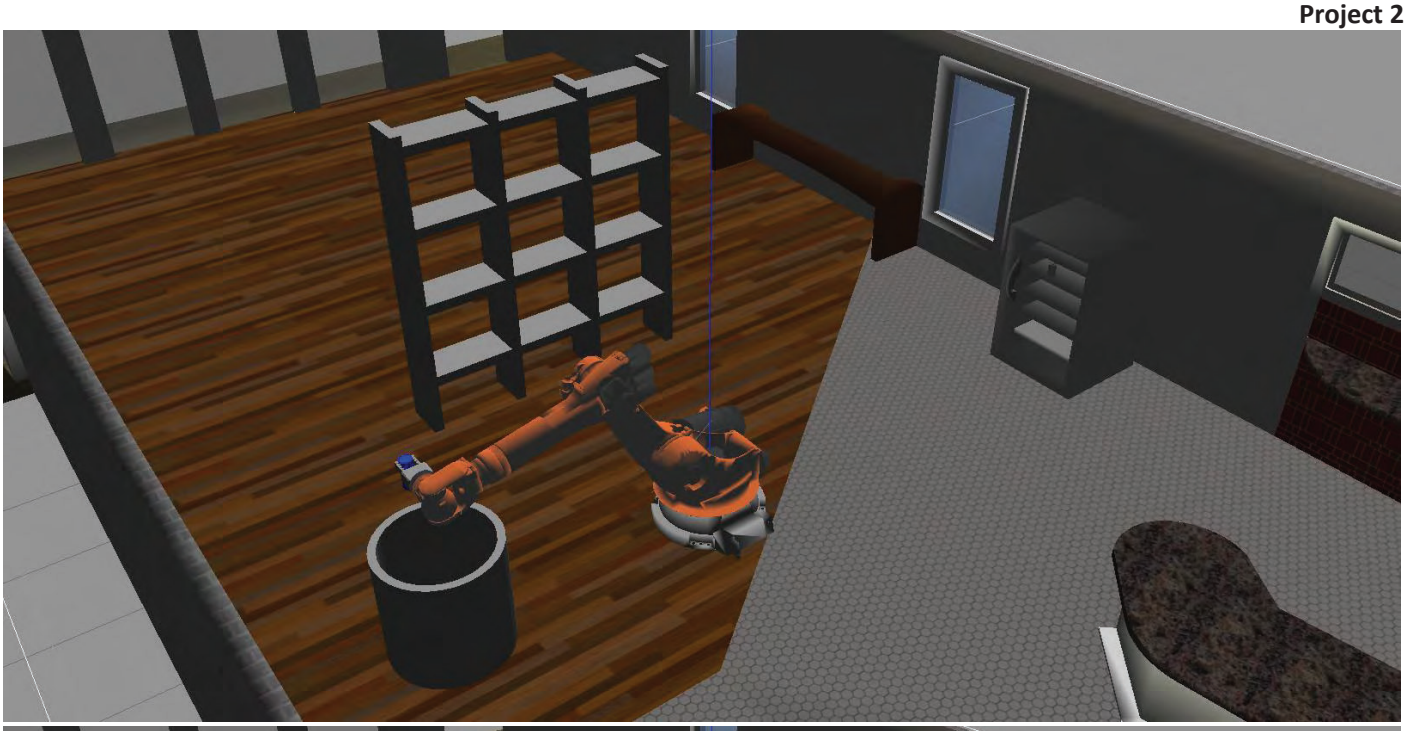


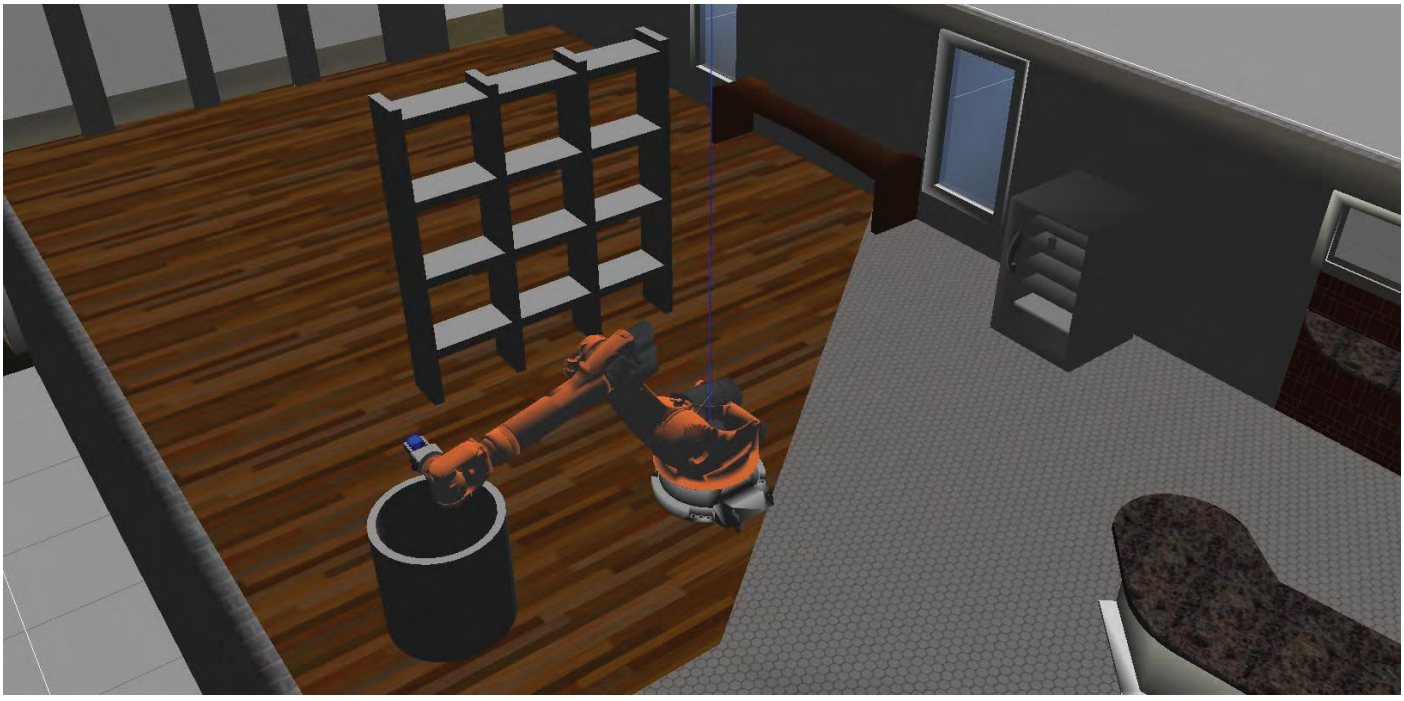
Yang Ding
Project 2

Yang Ding
Project 2

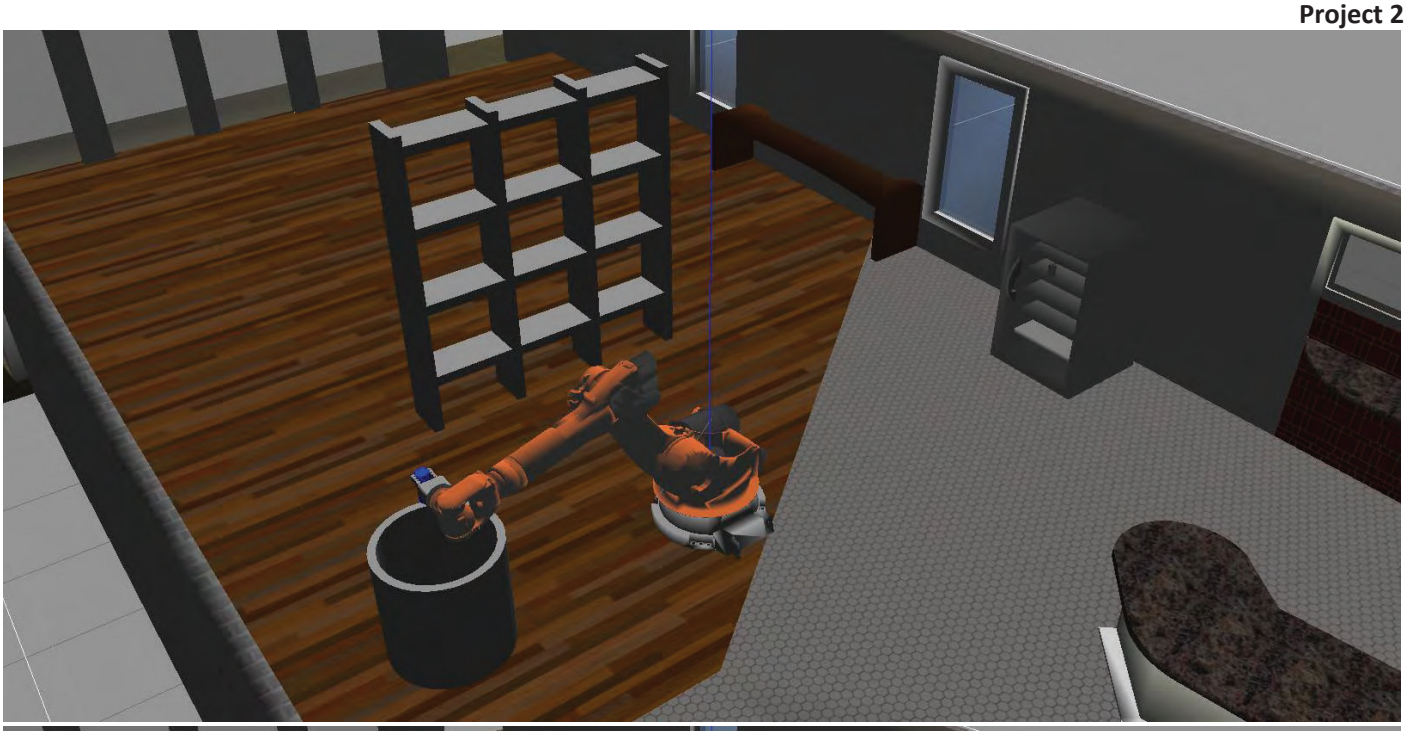
Yang Ding
Project 2

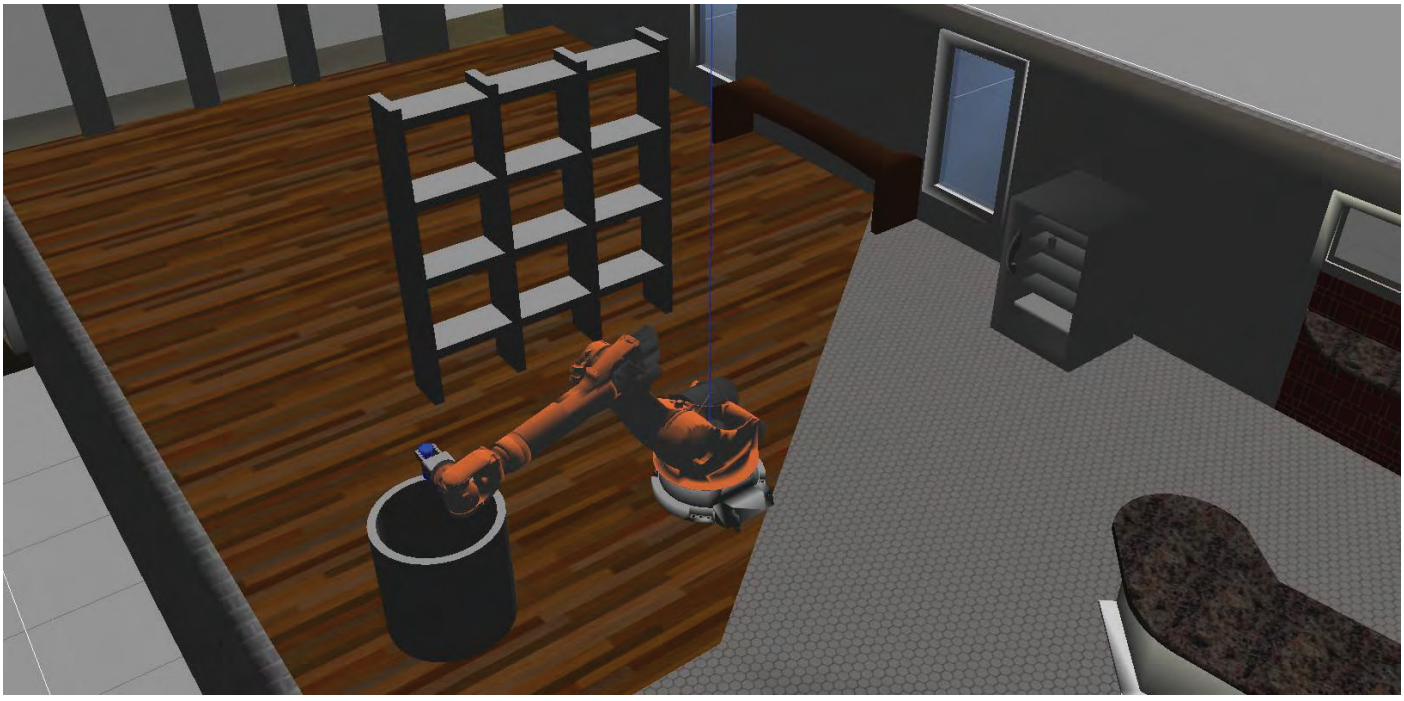
Yang Ding
Project 2



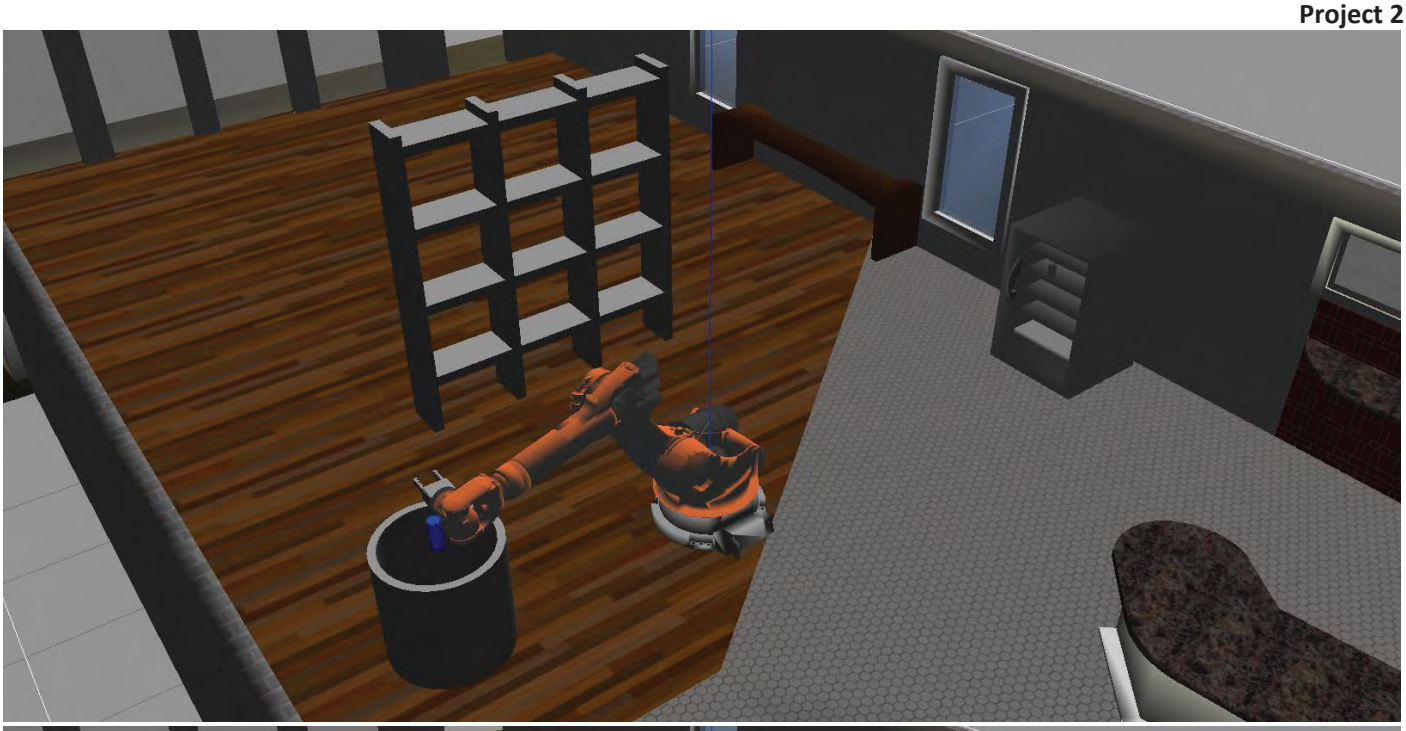


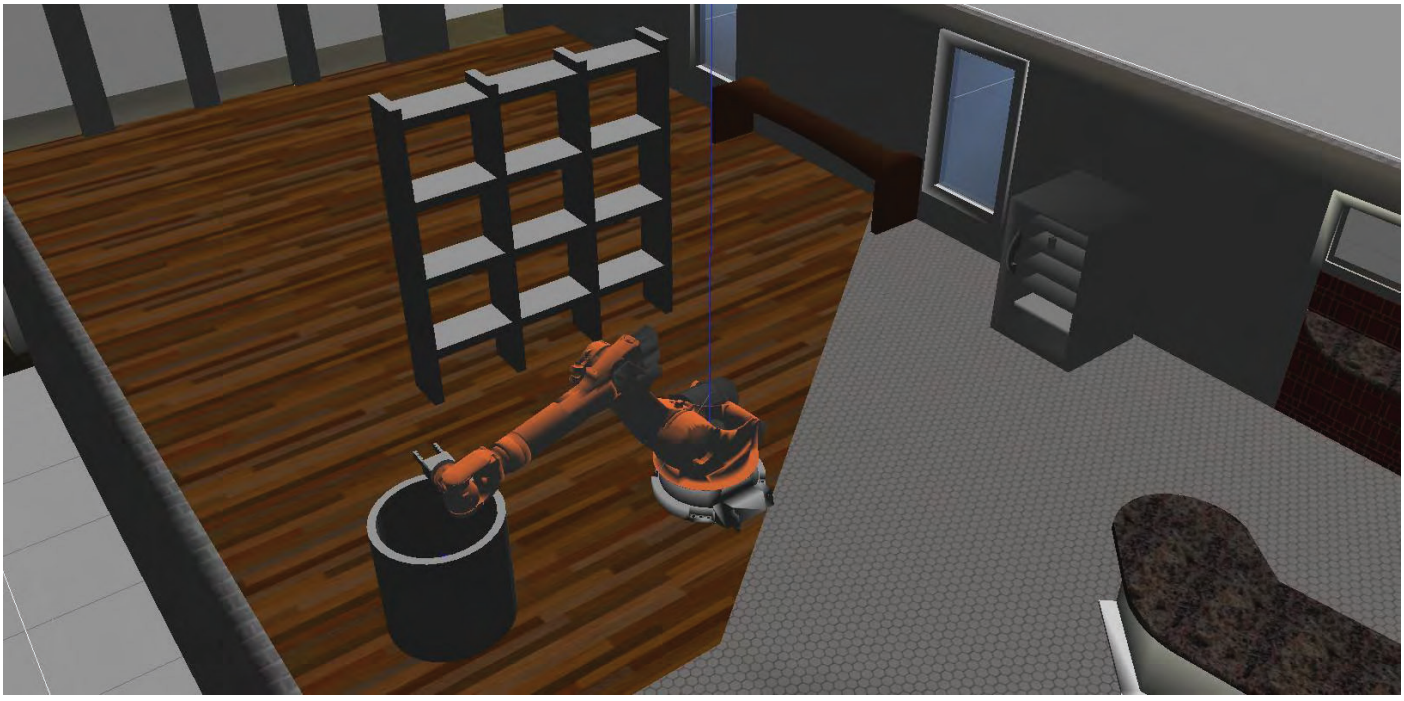
Yang Ding
Project 2





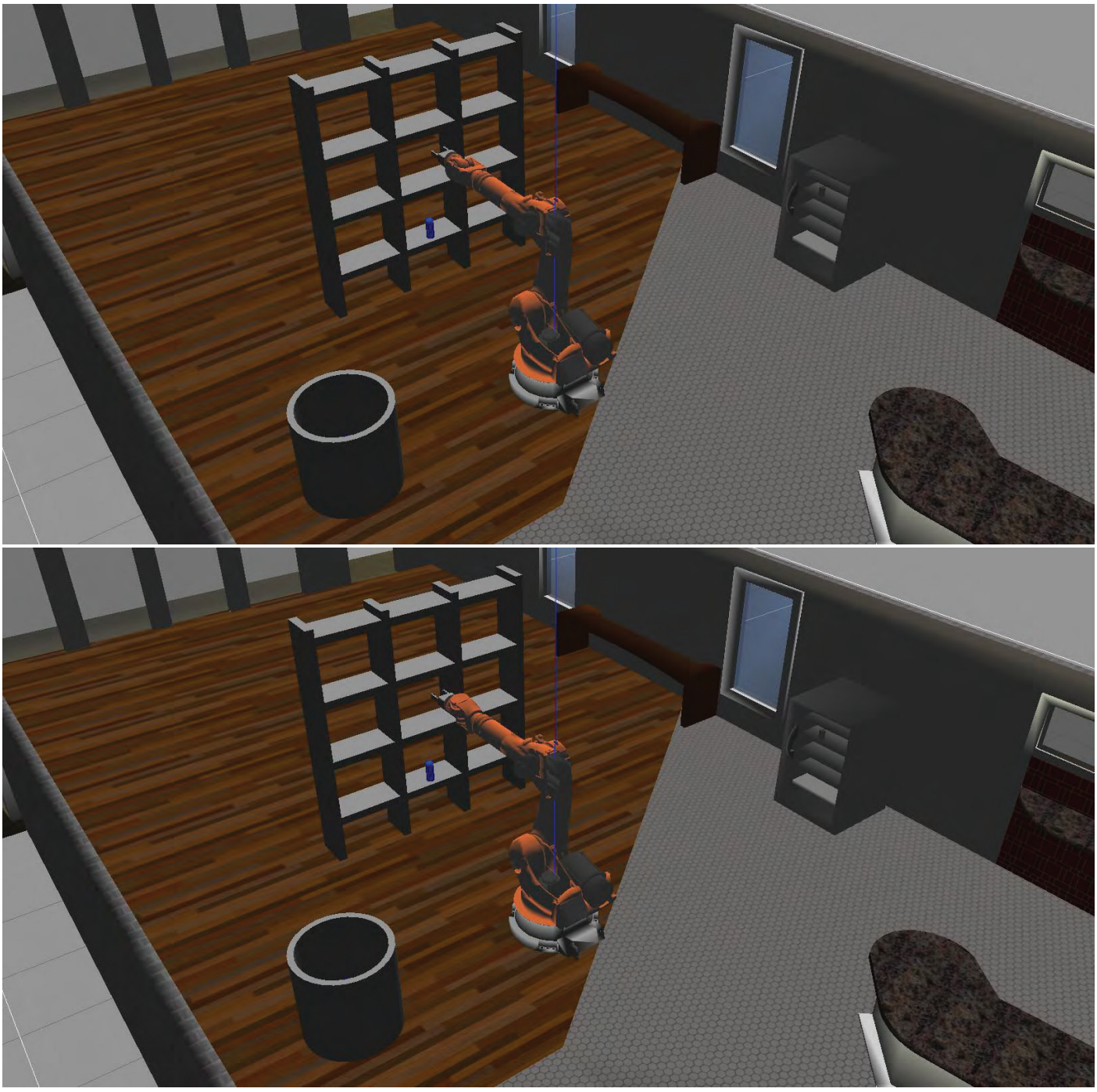
Yang Ding
Project 2



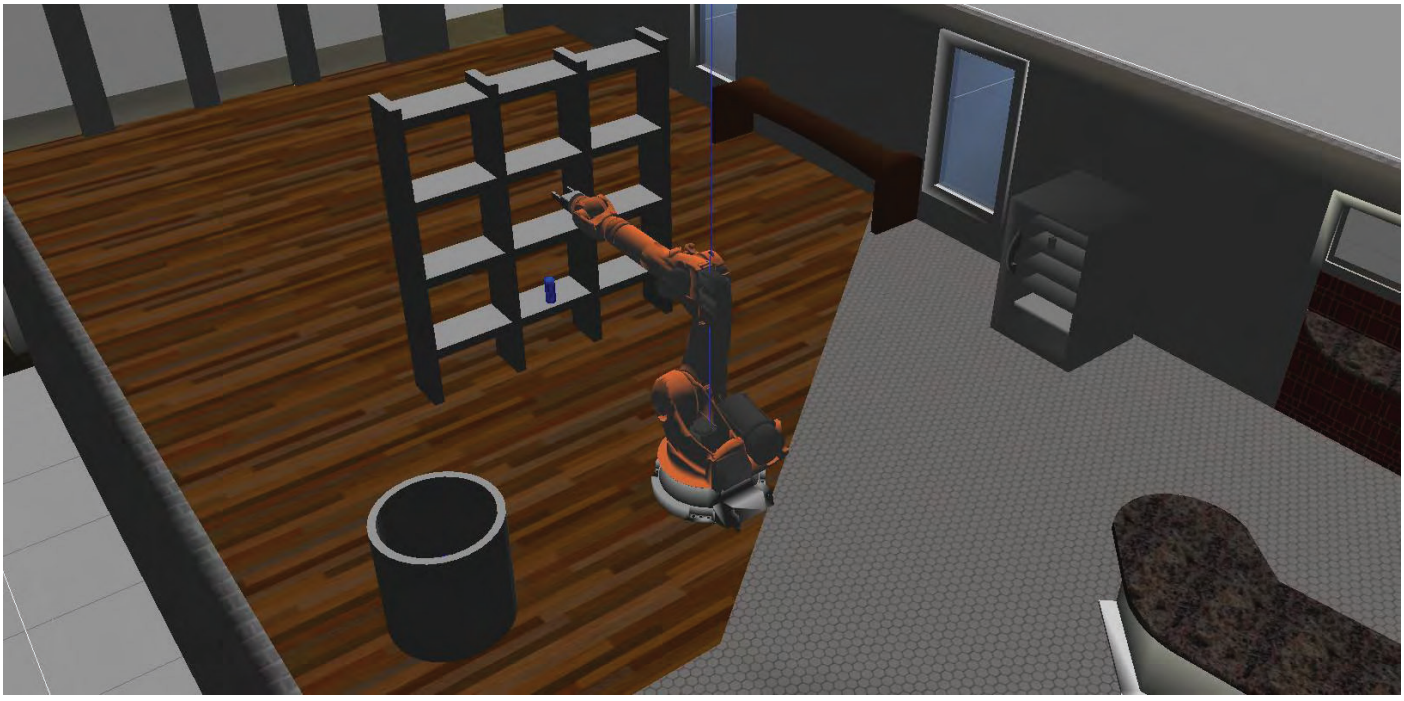


Yang Ding
Project 2

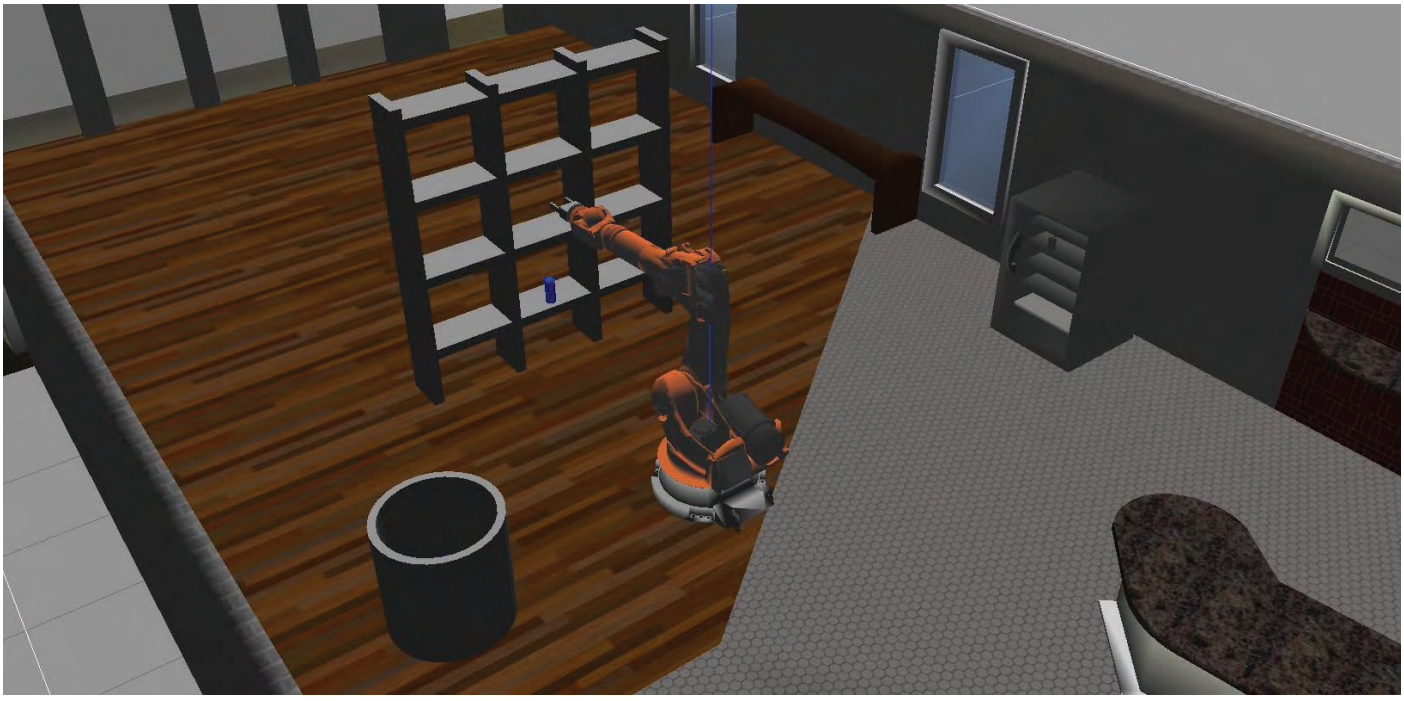
Slot 2



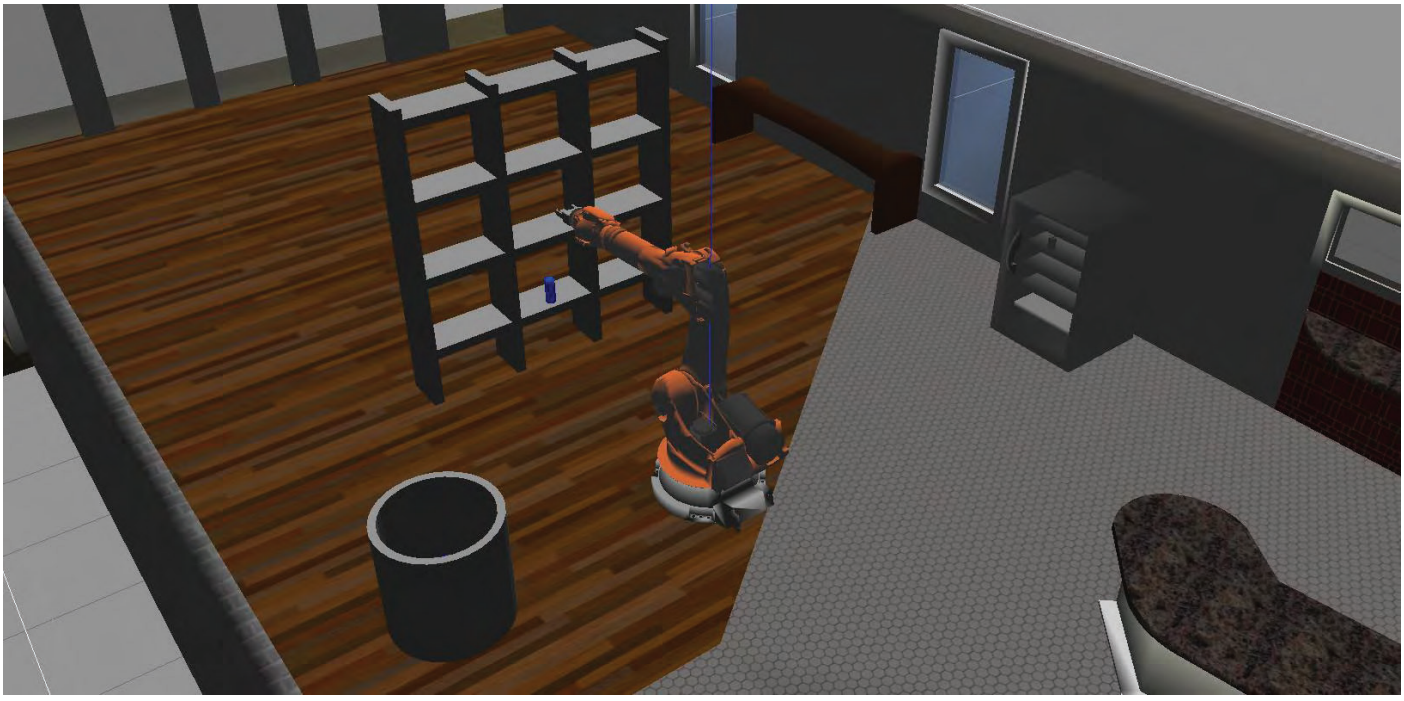
Yang Ding
Project 2



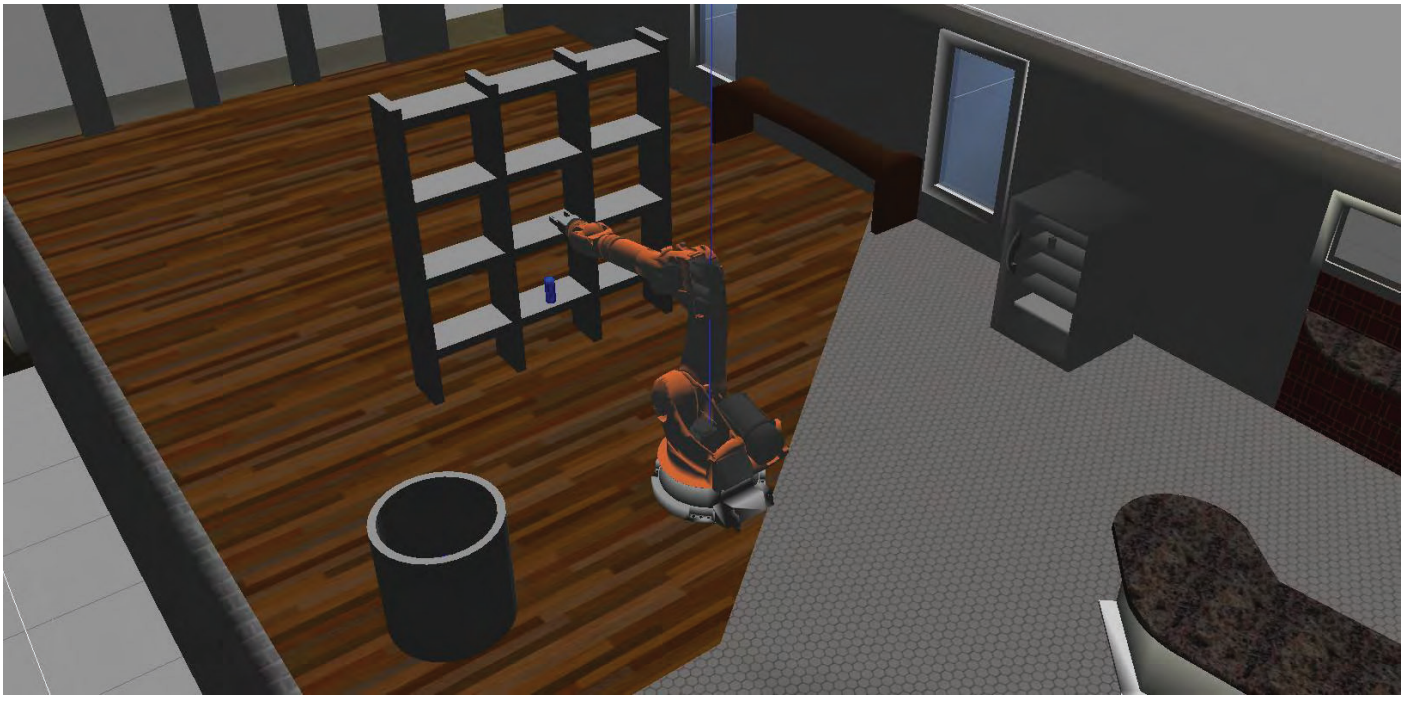
Yang Ding
Project 2



Yang Ding
Project 2

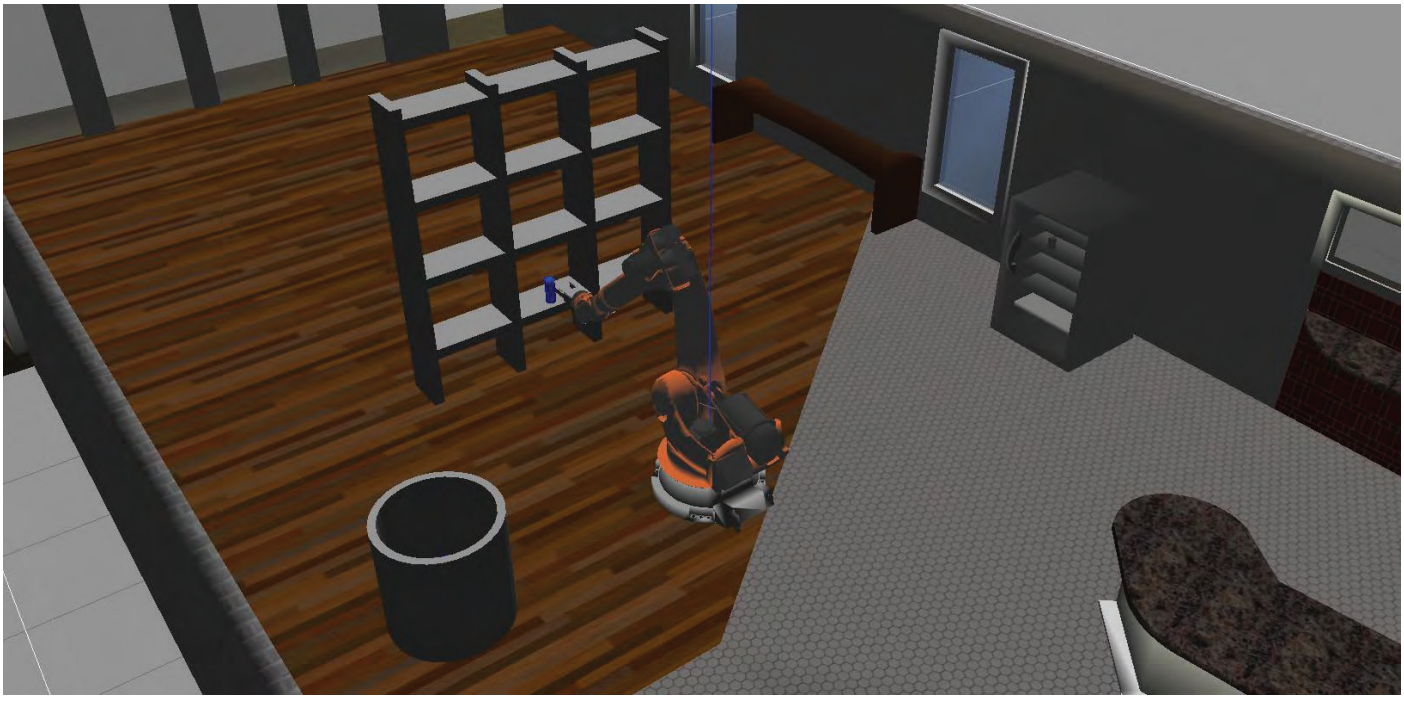


Yang Ding
Project 2



Yang Ding
Project 2

Yang Ding
Project 2



Yang Ding
Project 2

Yang Ding
Project 2