

# ENPM 662 Project 2 Report

## Simulation of a 6 DoF Pick and Place Robotic Arm

*Submitted by*

Vishwanath Datla(117705928)

Denesh Nallur Narasimman(117514339)



UNIVERSITY OF  
MARYLAND

## Contents

1	Introduction	2
2	Application	3
3	Robot Type	3
4	DOFs and Dimensions	3
5	CAD models	4
6	DH Parameters	5
7	Forward Kinematics	6
8	Inverse Kinematics	7
9	Forward and Inverse Kinematics Validation	8
10	Workspace Study	9
11	Assumptions	9
12	Control Method	9
13	Gazebo Visualisation	10
14	Rviz Visualisation	10
15	Problems	11
16	Lessons Learned	11
17	Conclusion	11
18	Future Works	12

# 1 Introduction

Robotic arms are accurate and reliable tools that can be used in a variety of different ways and industries, be it factory automation to move objects around, warehouse automation to select or sort goods or in the agricultural industry to remove weeds from the farmlands. Advancements in the fields of artificial intelligence and machine vision makes the use of robots to perform tasks even more convenient, efficient and viable.

Some of the advantages of using robotics arms are as follows:

**Precision:**

Robotic arms can be more precise in performing tasks by eliminating the possible human error. They can also be designed to operate at levels of precision that humans cannot operate at.

**Efficiency:**

Robotic arms are capable of working at the same level of efficiency for extended periods of time. There is no fatigue to be factored in. This enhances the amount of production.

**Safety:**

Robotic arms help keep workers safe by operating in environments that are hazardous and executing tasks that present high risk of injury to humans.[2]



Figure 1: Robotic arm. [3]

We intend to simulate a robotic arm that is capable of grabbing regular shaped objects of different sizes and moving them to a specified location. This robot can have applications in the automated manufacturing sector.

## 2 Application

Automated manufacturing is taking over every sector due to the efficiency with which the task can be implemented both in terms of accuracy and finance. Having to assemble product parts is a long, repetitive job that can be easily disrupted by a lapse in judgement, a bored mind, and many other factors that are uncontrollable. The reduced requirement of man hours makes it ideal for mass manufacturing industries. The speed of manufacturing also improves the efficiency of the process. Typically pick and place robots are mounted on a stable stand and positioned to reach different areas. For objects to be picked and placed in a different location in the same plane, a 5 degree of freedom robotic arm is used whereas to place the objects in a different plane, the robot must be able to re-orient the object. For this, the robotic arm must have 6 degrees of freedom.[4]

## 3 Robot Type

Our robot is a manipulator arm with 6 Degrees of Freedom. Currently, there are many manipulators that are available in the industry for various applications. Some of the basic applications are to use the manipulator as an aid to hold things or hang things; some of the advanced applications would be to pick and place some stuff from one place to another. Among these applications, we first chose to perform a pick and place function with the manipulator initially.

## 4 DOFs and Dimensions

The manipulator arm that we chose has 6 Degrees of Freedom. The manipulator arm, similar to a human arm, consists of a forearm, a shoulder joint, an elbow, a wrist and a gripper with two jaws. Each of these components shall work in unison to work like a human arm and achieve the required goal with the help of controllers in each joint.

The dimensions of the bot are:

Base = 260x300 mm All the joints (including the joint between the base and the shoulder joint): Height = 46mm Diameter = 60mm Shoulder inlet Diameter = 34mm

Arm:

Length = 105mm

Total Diameter = 40mm

Diameter at inlet of the Shoulder joint = 34mm

Gripper:

Length = 100mm

Diameter = 34mm

Total length of the claws = 81mm

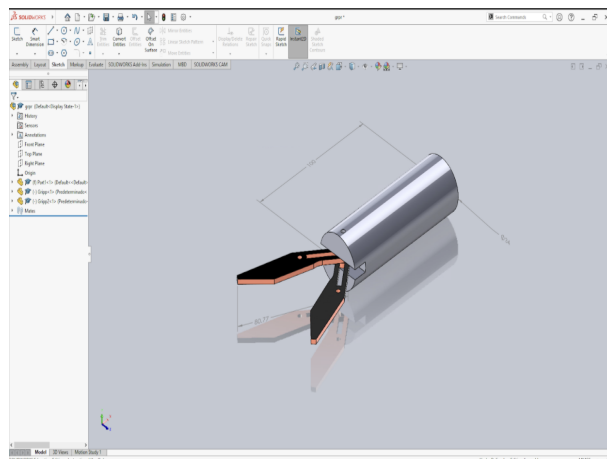


Figure 2: Gripper

## 5 CAD models

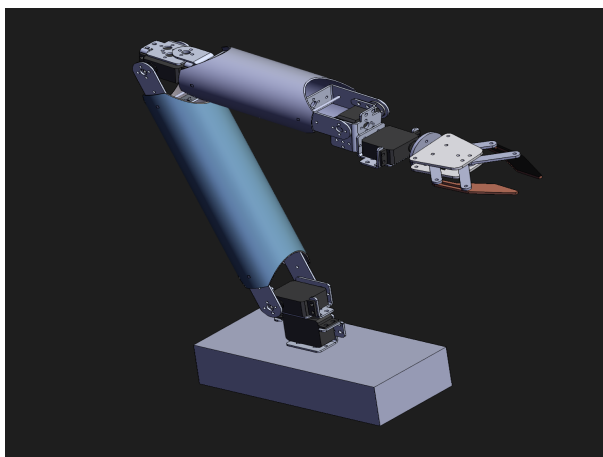


Figure 3: Original CAD model[5]

We planned to use the above CAD model as our manipulator for the pick and place application. This particular model turned out to have too many mechanical parts making the motion mechanics complex. Creating robots with

Closed Kinematics structures in URDF is not supported. To avoid this problem we used the following CAD model:

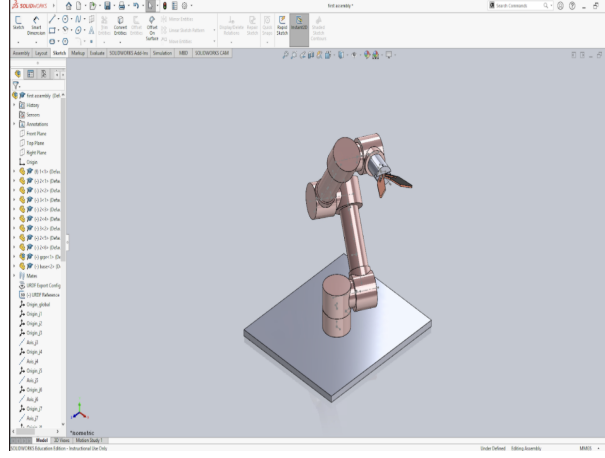


Figure 4: Implemented CAD model[6]

This model was picked due to it's simple mechanics. The gripper however was redesigned as demonstrated by the picture above because the gripper in the original design had a parallel link.

## 6 DH Parameters

i	$d_i$	$a_i$	$\alpha_i$
0	0	0	0
1	0	$d_1$	$\pi/2$
2	0	$a_2$	0
3	0	$a_3$	0
4	0	$d_4$	$\pi/2$
5	$d_5$	0	$-\pi/2$
6	$d_6$	0	0

Table 1: DH Parameters

The DH parameters of a manipulator can be derived from the kinematic diagram of the robot. The kinetic diagram of our manipulator is as follows: The values for the DH table can be found using the kinetic diagram and the following rules.

- 1) Draw the kinematic diagram.
- 2) Figure out the axes (Z axis is the axis of rotation, X axis is the one that lies among the common normal of previous z axis and current z axis and Y axis

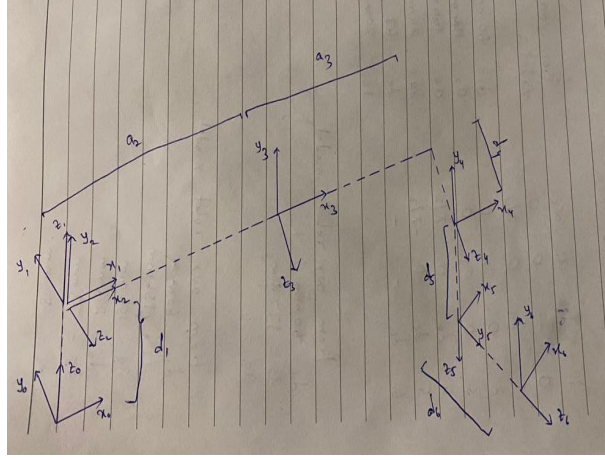


Figure 5: Kinematic diagram

can be determined using the right hand rule)

3) Calculate the DH parameters.[7]

For our manipulator, the values are:

$d_1 = 60mm, a_2 = a_3 = 156mm, d_4 = 23mm, d_5 = 73mm, d_6 = 80mm$

## 7 Forward Kinematics

Forward kinematics is the use of kinematic equations of a robot to determine the end-effector position using joint parameters. The forward kinematics of our manipulator can be determined using the DH parameters listed above.

Transformation matrices for each frame can be computed using the A matrices.

$$A_i = Rot_{z,\theta_i} \cdot Trans_{z,\theta_i} \cdot Trans_{x,\theta_i} \cdot Rot_{x,\alpha_i} \quad (1)$$

The final transformation matrix can be calculated by multiplying all the individual transformation matrices of all the frames.

Forward kinematics

Final transformation matrix:

$$T_6^0 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} R & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow \text{Derived from the A matrices}$$

Multiplying the 6 transformation matrices.

$$n_x = c_1(s_5s_6 + ((c_1c_{234} - s_1s_{234})c_5)/2 + ((c_1c_{234} + s_1s_{234})c_6)/2 - (c_1((c_1c_{234} + c_1s_{234}) - (s_1s_{234} - s_1s_{234}))) / 2$$

$$n_y = c_1((c_1c_{234} + c_1s_{234})c_5)/2 - c_1s_5 + ((c_1c_{234} - c_1s_{234})c_6)/2 + s_1((c_1c_{234} - s_1s_{234})c_5)/2 - (c_1c_{234} + c_1s_{234})/2$$

$$n_z = (s_{234}c_5 + c_{234}s_5)/2 + s_{234}c_6 - (s_{234}c_5 - c_{234}s_5)/2$$

$$o_x = -((c_1((c_1c_{234} + c_1s_{234}) - (s_1s_{234} - c_1s_{234}))) / 2 - s_1c_5s_6 + ((c_1c_{234} - s_1s_{234})c_5)/2 + ((c_1c_{234} + c_1s_{234})c_6)/2$$

$$o_y = -c_1c_5 - ((c_1c_{234} + c_1s_{234})s_5)/2 + ((c_1c_{234} - c_1s_{234})s_6)/2$$

$$o_z = (c_{234}c_5 - s_{234}s_5)/2 - (c_{234}c_6 + s_{234}s_6)/2$$

$$p_x = -(d_5(s_1c_{234} - c_1s_{234})/2 + (d_5(c_1c_{234} + c_1s_{234}))/2 + d_4s_1 - (d_5(c_1c_{234} - s_1s_{234})s_5)/2 - a_5c_1c_5 + a_5c_1c_2c_3 - a_5c_1s_2s_3)$$

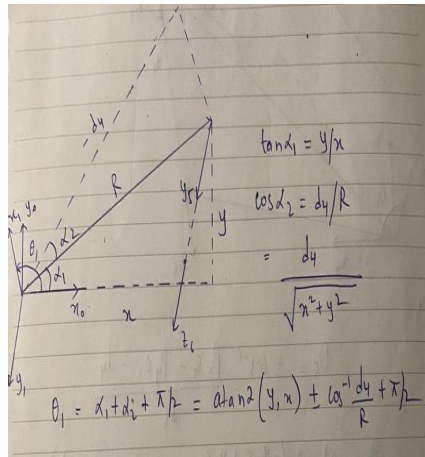
$$p_y = -(d_5(c_1c_{234} - s_1s_{234}))/2 + d_5(c_1c_{234} + c_1s_{234})/2 - d_4c_1 - (d_5(s_1c_{234} + c_1s_{234})s_5)/2 - (d_5(c_1c_{234} - c_1s_{234})s_6)/2 - d_5c_1c_5 + a_5c_2s_1 + a_5c_2c_3s_1 - a_5s_2s_3s_1$$

$$p_z = d_1 + (d_5(c_{234}c_5 - s_{234}s_5))/2 + a_5(s_5c_5 + c_5s_5) + a_5s_5 - (d_5(c_{234}c_6 + s_{234}s_6))/2 - d_5c_{234}$$

Figure 6: Forward kinematics

## 8 Inverse Kinematics

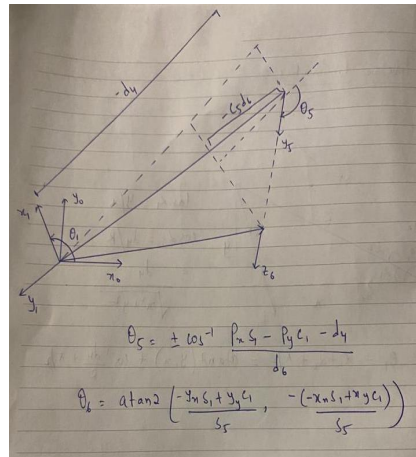
Inverse kinematics helps us calculate the joint angles given the desired position of the end effector. One of the approaches is the analytical approach. This uses trigonometric equations to determine the joint angles. The inverse kinematics for our manipulator is as follows: float



$$\tan \theta_1 = y/x$$

$$\cos \theta_1 = d_4/R$$

$$= \frac{d_4}{\sqrt{x^2 + y^2}}$$

$$\theta_1 = \arctan^2(y, x) + \cos^{-1} \frac{d_4}{R}$$


$$\theta_2 = \pm \cos^{-1} \frac{p_x c_1 - p_y c_1 - d_4}{d_6}$$

$$\theta_2 = \arctan^2 \left( \frac{-y n_1 + y_1 c_1}{s_5}, \frac{-(x n_1 + x_1 c_1)}{s_5} \right)$$

Figure 7: Inverse kinematics



## 9 Forward and Inverse Kinematics Validation

We have written a Python code to simulate the trajectory of our manipulator. This code was written using the DH parameters of our robot to simulate a circular trajectory. The code and result is as follows:

```

// ACTION FOR 4x4 MATRIX

// Function to get a transformation matrix from input parameters
// @: A1 = sym.symbols('a1')
1  def getA1(A1):
2      sym.Matrix([
3          [0, 0, 0, 0],
4          [0, 0, 0, 0],
5          [0, 0, 0, 0],
6          [0, 0, 0, 0]
7      ])
8      return A1

9  # Create a 4x4 transformation matrix
10  A1 = getA1(A1)

11  # Create a 4x4 transformation matrix
12  A2 = sym.Matrix([
13      [0, 0, 0, 0],
14      [0, 0, 0, 0],
15      [0, 0, 0, 0],
16      [0, 0, 0, 0]
17  ])

18  # Create a 4x4 transformation matrix
19  A3 = sym.Matrix([
20      [0, 0, 0, 0],
21      [0, 0, 0, 0],
22      [0, 0, 0, 0],
23      [0, 0, 0, 0]
24  ])

25  # Create a 4x4 transformation matrix
26  A4 = sym.Matrix([
27      [0, 0, 0, 0],
28      [0, 0, 0, 0],
29      [0, 0, 0, 0],
30      [0, 0, 0, 0]
31  ])

32  # Create a 4x4 transformation matrix
33  A5 = sym.Matrix([
34      [0, 0, 0, 0],
35      [0, 0, 0, 0],
36      [0, 0, 0, 0],
37      [0, 0, 0, 0]
38  ])

39  # Create a 4x4 transformation matrix
40  A6 = sym.Matrix([
41      [0, 0, 0, 0],
42      [0, 0, 0, 0],
43      [0, 0, 0, 0],
44      [0, 0, 0, 0]
45  ])

46  # Create a 4x4 transformation matrix
47  A7 = sym.Matrix([
48      [0, 0, 0, 0],
49      [0, 0, 0, 0],
50      [0, 0, 0, 0],
51      [0, 0, 0, 0]
52  ])

53  # Create a 4x4 transformation matrix
54  A8 = sym.Matrix([
55      [0, 0, 0, 0],
56      [0, 0, 0, 0],
57      [0, 0, 0, 0],
58      [0, 0, 0, 0]
59  ])

60  # Create a 4x4 transformation matrix
61  A9 = sym.Matrix([
62      [0, 0, 0, 0],
63      [0, 0, 0, 0],
64      [0, 0, 0, 0],
65      [0, 0, 0, 0]
66  ])

67  # Create a 4x4 transformation matrix
68  A10 = sym.Matrix([
69      [0, 0, 0, 0],
70      [0, 0, 0, 0],
71      [0, 0, 0, 0],
72      [0, 0, 0, 0]
73  ])

74  # Create a 4x4 transformation matrix
75  A11 = sym.Matrix([
76      [0, 0, 0, 0],
77      [0, 0, 0, 0],
78      [0, 0, 0, 0],
79      [0, 0, 0, 0]
80  ])

81  # Create a 4x4 transformation matrix
82  A12 = sym.Matrix([
83      [0, 0, 0, 0],
84      [0, 0, 0, 0],
85      [0, 0, 0, 0],
86      [0, 0, 0, 0]
87  ])

88  # Create a 4x4 transformation matrix
89  A13 = sym.Matrix([
90      [0, 0, 0, 0],
91      [0, 0, 0, 0],
92      [0, 0, 0, 0],
93      [0, 0, 0, 0]
94  ])

95  # Create a 4x4 transformation matrix
96  A14 = sym.Matrix([
97      [0, 0, 0, 0],
98      [0, 0, 0, 0],
99      [0, 0, 0, 0],
100     [0, 0, 0, 0]
101 ])

102 # Create a 4x4 transformation matrix
103 A15 = sym.Matrix([
104     [0, 0, 0, 0],
105     [0, 0, 0, 0],
106     [0, 0, 0, 0],
107     [0, 0, 0, 0]
108 ])

109 # Create a 4x4 transformation matrix
110 A16 = sym.Matrix([
111     [0, 0, 0, 0],
112     [0, 0, 0, 0],
113     [0, 0, 0, 0],
114     [0, 0, 0, 0]
115 ])

116 # Create a 4x4 transformation matrix
117 A17 = sym.Matrix([
118     [0, 0, 0, 0],
119     [0, 0, 0, 0],
120     [0, 0, 0, 0],
121     [0, 0, 0, 0]
122 ])

123 # Create a 4x4 transformation matrix
124 A18 = sym.Matrix([
125     [0, 0, 0, 0],
126     [0, 0, 0, 0],
127     [0, 0, 0, 0],
128     [0, 0, 0, 0]
129 ])

130 # Create a 4x4 transformation matrix
131 A19 = sym.Matrix([
132     [0, 0, 0, 0],
133     [0, 0, 0, 0],
134     [0, 0, 0, 0],
135     [0, 0, 0, 0]
136 ])

137 # Create a 4x4 transformation matrix
138 A20 = sym.Matrix([
139     [0, 0, 0, 0],
140     [0, 0, 0, 0],
141     [0, 0, 0, 0],
142     [0, 0, 0, 0]
143 ])

144 # Create a 4x4 transformation matrix
145 A21 = sym.Matrix([
146     [0, 0, 0, 0],
147     [0, 0, 0, 0],
148     [0, 0, 0, 0],
149     [0, 0, 0, 0]
150 ])

151 # Create a 4x4 transformation matrix
152 A22 = sym.Matrix([
153     [0, 0, 0, 0],
154     [0, 0, 0, 0],
155     [0, 0, 0, 0],
156     [0, 0, 0, 0]
157 ])

158 # Create a 4x4 transformation matrix
159 A23 = sym.Matrix([
160     [0, 0, 0, 0],
161     [0, 0, 0, 0],
162     [0, 0, 0, 0],
163     [0, 0, 0, 0]
164 ])

165 # Create a 4x4 transformation matrix
166 A24 = sym.Matrix([
167     [0, 0, 0, 0],
168     [0, 0, 0, 0],
169     [0, 0, 0, 0],
170     [0, 0, 0, 0]
171 ])

172 # Create a 4x4 transformation matrix
173 A25 = sym.Matrix([
174     [0, 0, 0, 0],
175     [0, 0, 0, 0],
176     [0, 0, 0, 0],
177     [0, 0, 0, 0]
178 ])

179 # Create a 4x4 transformation matrix
180 A26 = sym.Matrix([
181     [0, 0, 0, 0],
182     [0, 0, 0, 0],
183     [0, 0, 0, 0],
184     [0, 0, 0, 0]
185 ])

186 # Create a 4x4 transformation matrix
187 A27 = sym.Matrix([
188     [0, 0, 0, 0],
189     [0, 0, 0, 0],
190     [0, 0, 0, 0],
191     [0, 0, 0, 0]
192 ])

193 # Create a 4x4 transformation matrix
194 A28 = sym.Matrix([
195     [0, 0, 0, 0],
196     [0, 0, 0, 0],
197     [0, 0, 0, 0],
198     [0, 0, 0, 0]
199 ])

200 # Create a 4x4 transformation matrix
201 A29 = sym.Matrix([
202     [0, 0, 0, 0],
203     [0, 0, 0, 0],
204     [0, 0, 0, 0],
205     [0, 0, 0, 0]
206 ])

207 # Create a 4x4 transformation matrix
208 A30 = sym.Matrix([
209     [0, 0, 0, 0],
210     [0, 0, 0, 0],
211     [0, 0, 0, 0],
212     [0, 0, 0, 0]
213 ])

214 # Create a 4x4 transformation matrix
215 A31 = sym.Matrix([
216     [0, 0, 0, 0],
217     [0, 0, 0, 0],
218     [0, 0, 0, 0],
219     [0, 0, 0, 0]
220 ])

221 # Create a 4x4 transformation matrix
222 A32 = sym.Matrix([
223     [0, 0, 0, 0],
224     [0, 0, 0, 0],
225     [0, 0, 0, 0],
226     [0, 0, 0, 0]
227 ])

228 # Create a 4x4 transformation matrix
229 A33 = sym.Matrix([
230     [0, 0, 0, 0],
231     [0, 0, 0, 0],
232     [0, 0, 0, 0],
233     [0, 0, 0, 0]
234 ])

235 # Create a 4x4 transformation matrix
236 A34 = sym.Matrix([
237     [0, 0, 0, 0],
238     [0, 0, 0, 0],
239     [0, 0, 0, 0],
240     [0, 0, 0, 0]
241 ])

242 # Create a 4x4 transformation matrix
243 A35 = sym.Matrix([
244     [0, 0, 0, 0],
245     [0, 0, 0, 0],
246     [0, 0, 0, 0],
247     [0, 0, 0, 0]
248 ])

249 # Create a 4x4 transformation matrix
250 A36 = sym.Matrix([
251     [0, 0, 0, 0],
252     [0, 0, 0, 0],
253     [0, 0, 0, 0],
254     [0, 0, 0, 0]
255 ])

256 # Create a 4x4 transformation matrix
257 A37 = sym.Matrix([
258     [0, 0, 0, 0],
259     [0, 0, 0, 0],
260     [0, 0, 0, 0],
261     [0, 0, 0, 0]
262 ])

263 # Create a 4x4 transformation matrix
264 A38 = sym.Matrix([
265     [0, 0, 0, 0],
266     [0, 0, 0, 0],
267     [0, 0, 0, 0],
268     [0, 0, 0, 0]
269 ])

270 # Create a 4x4 transformation matrix
271 A39 = sym.Matrix([
272     [0, 0, 0, 0],
273     [0, 0, 0, 0],
274     [0, 0, 0, 0],
275     [0, 0, 0, 0]
276 ])

277 # Create a 4x4 transformation matrix
278 A40 = sym.Matrix([
279     [0, 0, 0, 0],
280     [0, 0, 0, 0],
281     [0, 0, 0, 0],
282     [0, 0, 0, 0]
283 ])

284 # Create a 4x4 transformation matrix
285 A41 = sym.Matrix([
286     [0, 0, 0, 0],
287     [0, 0, 0, 0],
288     [0, 0, 0, 0],
289     [0, 0, 0, 0]
290 ])

291 # Create a 4x4 transformation matrix
292 A42 = sym.Matrix([
293     [0, 0, 0, 0],
294     [0, 0, 0, 0],
295     [0, 0, 0, 0],
296     [0, 0, 0, 0]
297 ])

298 # Create a 4x4 transformation matrix
299 A43 = sym.Matrix([
300     [0
```

Figure 8: Code for trajectory simulation

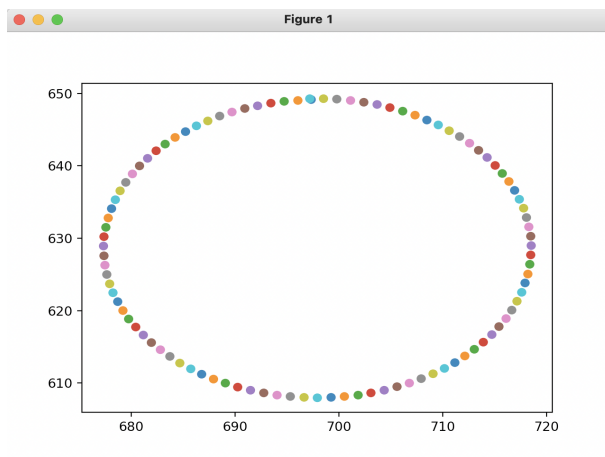


Figure 9: Simulation result

## 10 Workspace Study

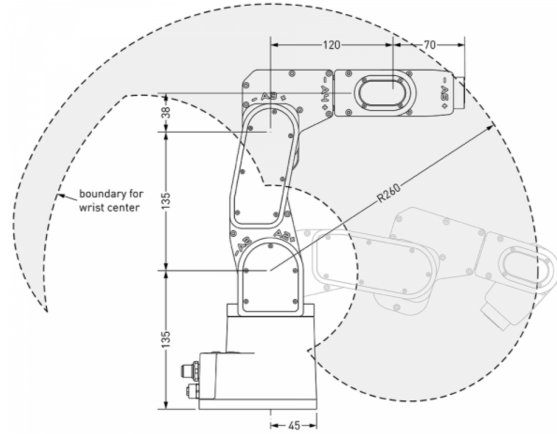


Figure 10: Workspace study

The above figure shows the workspace graphic of a similar robotic arm manipulator in an environment. Our arm will have a maximum reach of 280mm. The manipulator arm can be found in various workspaces especially, in the assembly line in an automobile manufacturing company. These types of manipulators are mainly used where there is a need for human intervention but the conditions of operation can be harmful to us like in research laboratories to handle harmful chemicals and substances and robots that are used to diffuse bombs. The workspace wherever this kind of manipulator is used must be very organised as there is always a chance of any object coming in the way of the manipulator which can cause hindrance to the function of the said manipulator.

## 11 Assumptions

All the links and joints have good structural integrity.

The friction and other external vibrations are not considered.

The objects that are picked and placed will have well defined edges and shapes.

The path of the robot arm is defined but may not be the ideal path.

The objects are within the maximum reach of the arm.

## 12 Control Method

We have opted to go with PID controllers for our manipulator. PID stands for proportional, integral, derivative and is a feedback controller. Changing

the proportional, integral and derivative gains of the controller will change the impact the controller will have. We can vary these gains depending on our needs. The controller tries to reduce the error in the current position. The proportional gain as the name suggests varies the error proportionally, the integral gain impacts the summation of errors over time and is used to correct the steady state errors, the derivative gain impacts the rate of change of the error[8]. The PID gains we used for our controllers are:

## 13 Gazebo Visualisation

Gazebo is a simulation tool that is widely used to test or simulate robots in an environment which can be created by us. Since, initially, the bot was supposed to carry out the function of pick and place, we placed a coke-o-cola can in the environment for it to pick and place it in a different place. But, unfortunately, due to an issue with the controllers, the spawned bot in Gazebo does not seem to move and fall upon itself. Also, the initial plan was to place a 3D LiDAR sensor on the manipulator arm so that it can visualize 3D shapes in order to pick and place which now has become the future scope of the robot.

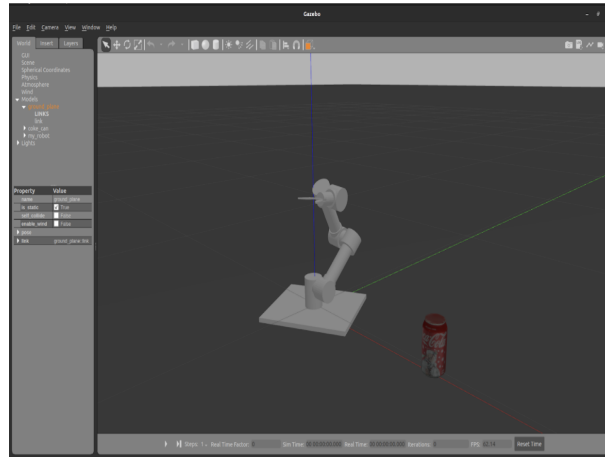


Figure 11: Gazebo visualisation

## 14 Rviz Visualisation

RViZ is a tool that is used to visualize the data that is being collected from a sensor (for example: LiDAR, Camera etc.). Here, since we have not used any sensor, there will not be any RViZ visualization.

## 15 Problems

- 1) As mentioned above, the original CAD model picked for the application had complicated mechanics and also had a parallel link due to which it could not be exported as URDF. A different model was then picked.
- 2) The gripper of the new model also had a parallel link and thus we designed our own gripper.
- 3) When we initially did the gazebo visualisation of the robot, we found that it was collapsing and not standing upright. This problem was solved by
- 4) We could not implement the python code for pick and place application of the robot.

## 16 Lessons Learned

Could have attended more TA office hours to seek help with technical issues regarding the project

Could have managed time and other assignments better

## 17 Conclusion

It was a great learning experience for us while completing this project. We got to learn a lot from designing, to exporting the URDF of the model, to adding the controllers and transmission and simulating it in Gazebo. We successfully spawned the robotic arm manipulator in the gazebo simulator. Unfortunately, the robot was not able to move because there was an issue with the controller. The initial plan for the project was to program it to pick and place circular or cylindrical objects. We did, although, try to implement moveit function but could not implement it so instead we decided to do a similar teleop file to project 1. But, that did not work either. We would like to acknowledge the fact that because of all the errors along the duration of the project and the unfortunate situation of submitting the project incomplete, after trying our best to make it work. we got the opportunity to learn more. Our future plan with this project is to spawn the bot properly and with the help of LiDAR, the arm will estimate the 3D object and move towards it to pick it and place it in another place. The future plan also includes providing different end effectors for various applications.

This project has been a learning experience for us and we would like to thank Dr. Reza Monfaredi, Mr. Ajinkya Abhay Parwekar and Mr. Karan Sutradhar to help us with the project.

## 18 Future Works

In the future, we intend to get our robot to be capable of picking up the coke can shown in the gazebo visualisation. This would require further work on the python code that can be written using moveit libraries and rospy. To further advance this project, we can add a lidar sensor or a camera so the edges of the objects to be picked up can be perceived and the robot can be made capable of adapting to the object it needs to pick up. This improves the functionality of the robot as it can be used to pick and place objects of different sizes.

## References

- [1] <https://docs.ros.org>
- [2] <https://www.intel.com/content/www/us/en/robotics/robotic-arm.html>
- [3] <https://blog.ansi.org/2019/01/industrial-robots-safety-production-ria/>
- [4] <https://6river.com/what-is-a-pick-and-place-robot/>
- [5] [https://grabcad.com/library/robotic-arm-six-degree-of-freedom-1/details?folder\\_id=3657589](https://grabcad.com/library/robotic-arm-six-degree-of-freedom-1/details?folder_id=3657589)
- [6] <https://grabcad.com/library/sample-robotic-arm-manipulator-1>
- [7] <https://blog.robotiq.com/how-to-calculate-a-robots-forward-kinematics-in-5-easy-steps>
- [8] [https://en.wikipedia.org/wiki/PID\\_controller](https://en.wikipedia.org/wiki/PID_controller)