ENPM-662 Final Project Report

By:

119262689: Mudit Singal

119399545: Joseph Thomas

Table of Contents:

Introduction:

KUKA youBot is a mobile manipulator, which is controlled with open source software, has an omnidirectional base and robotic mechanics with five degrees of freedom. Just two years after its introduction to the market, it is already the reference platform for research and training in the field of mobile manipulation. Worldwide, KUKA is primarily known as a manufacturer of industrial robots. With the youBot robot, KUKA has created an open source reference platform for robotics research, enabling developers, researchers and students to write their own control and application software.

It consists of a chassis, which can be moved omnidirectionally, with either one or two robotic arms mounted on the chassis. An industrial PC and battery have been integrated into the chassis. Via EtherCAT, the industrial PC communicates in real time (1 ms cycle) with its nine drives, which can be operated with current control, velocity control, and position control. The robotic platform and arm can also be used independently from each other.

The KUKA youBot moves on four Mecanum wheels. These special wheels have rolls mounted around the circumference at a 45°angle to the wheel's plane and enable the youBot to combine any translational and rotational movements at any given time. This makes omnidirectional motion possible, including sideways and diagonal motion.

Application:

The robot has omni-directional wheels that can help in moving the robot from one point to another on an entire plane, which is perfect for warehouse management and home automation. The robot will have a jaw gripper being used as an end effector which is useful for picking and placing objects as required by the user. With a small size, the robot will be able to maneuver tight spaces that are crucial to freely moving around even in confined spaces.

Robot type:

The youBot is a mobile robotic arm developed by KUKA. Its arm has five degrees of freedom and a linear gripper. Its base has four Mecanum wheels allowing for omni-directional movement. These wheels are efficiently modeled using asymmetric friction.

DOFs and Dimensions:

The robot we designed has a total DOF of 9. Three on the base (mecanum-wheels) which allows the robot to translate on the x-y plane and rotate about the z-axis. The robotic manipulator arm on the robot has 5 degrees of freedom and one degree of freedom is present for the gripper.

CAD Model:

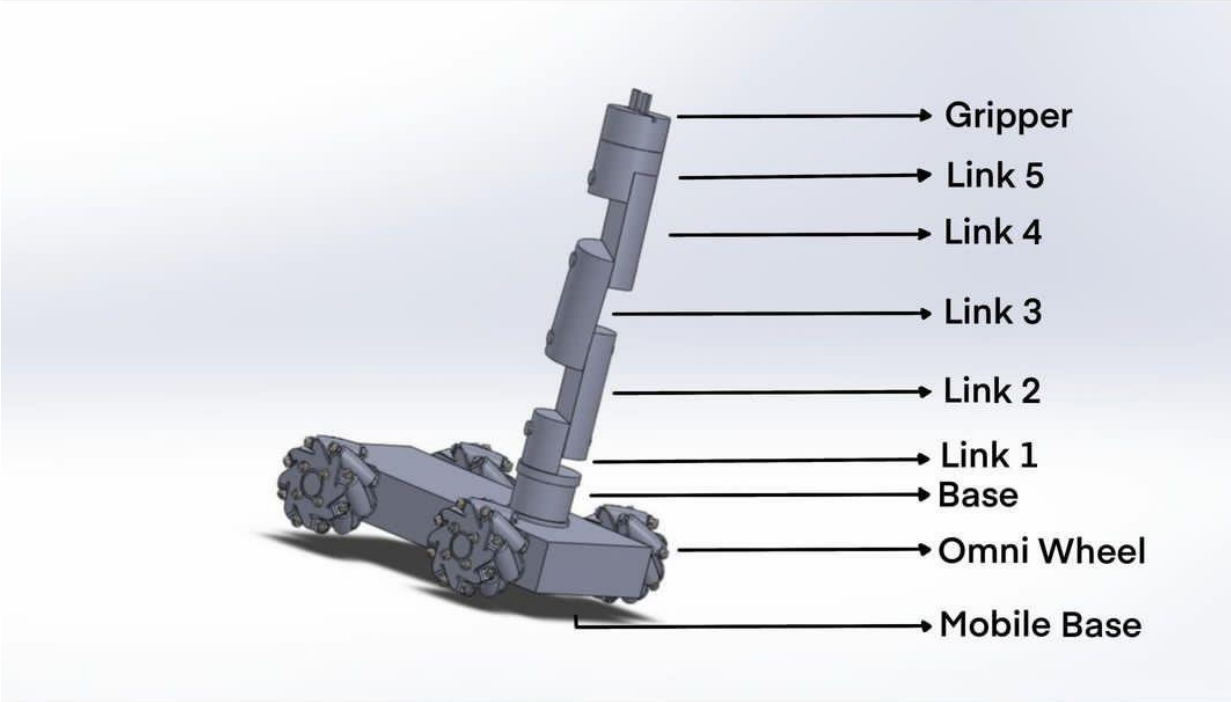

Fig. 1: CAD design in Solidworks 2022

D-H Parameters:

Joint	**a**	***α***	**d**	**θ**
1	-0.1662	0	0	-90
2	-0.0331	-90	0.2454	θ_1
3	-0.155	0	0	θ_2+90
4	-0.1349	0	0	θ_3
5	0	-90	0	θ_4+90
E.F.	0	0	0.1939	θ_5

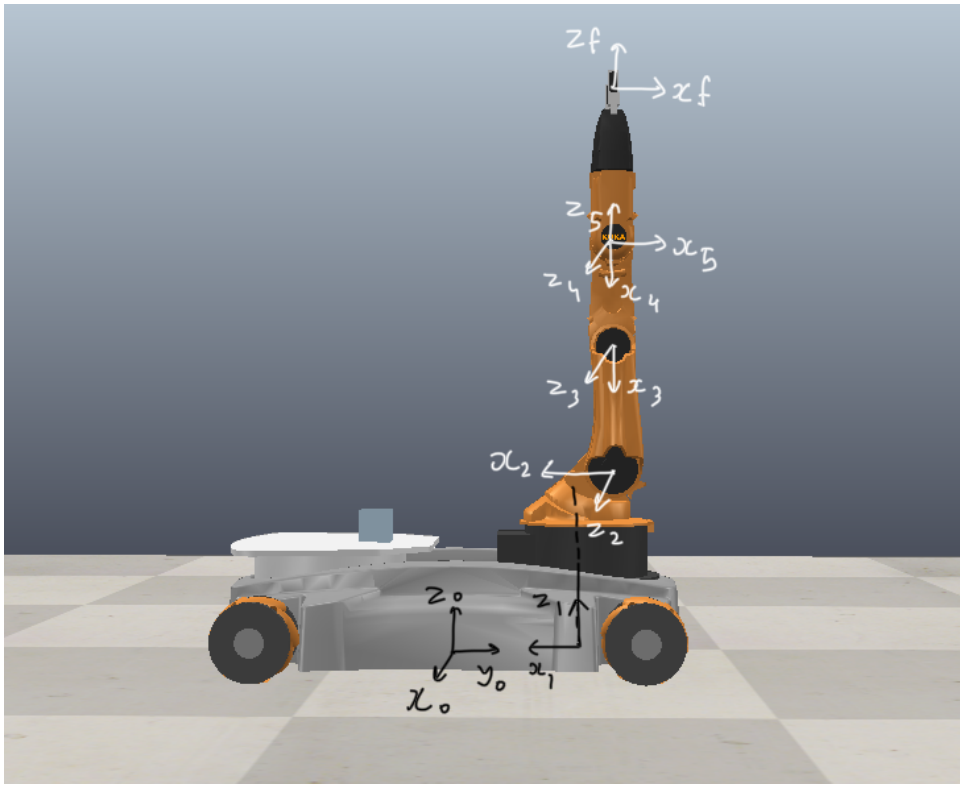

Fig.2: DH frame assignment

Forward Kinematics:

The individual transformation matrices are computed from DH parameters using the following matrix form:

$$T^{i}_{i-1} =$$

[[cos(theta_i), -cos(alpha_i)*sin(theta_i), sin(alpha_i)*sin(theta_i), a_i*cos(theta_i)],
[sin(theta_i), cos(alpha_i)*cos(theta_i), -sin(alpha_i)*cos(theta_i), a_i*sin(theta_i)],
[0, sin(alpha_i), cos(alpha_i), d_i],
[0, 0, 0, 1]]
Where, theta_i, alpha_i, a_i and d_i are the DH parameters of the i th frame.

The end-effector transformation matrix is calculated in the following manner:

$$T_{ef} = {}^{1}_{0}T\,{}^{2}_{1}T\,{}^{3}_{2}T\,{}^{4}_{3}T\,{}^{5}_{4}T\,{}^{6}_{5}T$$

The end-effector transformation matrix that we get in terms of joint angles theta 1 to theta 5 is given below:

```
Matrix([[((-((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) +
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) +
((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) -
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*sin(Theta4 +
1.570796) + (((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) +
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) +
((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) -
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*cos(Theta4 +
1.570796))*cos(Theta5) + (3.26794896538138e-7*(-((-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) + (0.999999999999947*sin(Theta1) +
3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) + ((-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) - (0.999999999999947*sin(Theta1) +
3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*cos(Theta4 + 1.570796) -
3.26794896538138e-7*(((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) +
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) +
((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) -
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*sin(Theta4 +
1.570796) + 3.26794896538103e-7*sin(Theta1) - 0.99999999999984*cos(Theta1))*sin(Theta5),
-((-((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) +
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) +
((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) -
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*sin(Theta4 +
1.570796) + (((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) +
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) +
((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) -
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*cos(Theta4 +
1.570796))*sin(Theta5) + (3.26794896538138e-7*(-((-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) + (0.999999999999947*sin(Theta1) +
3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) + ((-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) - (0.999999999999947*sin(Theta1) +
3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*cos(Theta4 + 1.570796) -
3.26794896538138e-7*(((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) +
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) +
((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) -
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*sin(Theta4 +
1.570796) + 3.26794896538103e-7*sin(Theta1) - 0.99999999999984*cos(Theta1))*cos(Theta5),
0.999999999999947*(-((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) +
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) +
((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) -
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*cos(Theta4 +
1.570796) - 0.999999999999947*(((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*sin(Theta2 +
1.570796) + (0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) +
((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) -
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*sin(Theta4 +
1.570796) - 1.06794904403367e-13*sin(Theta1) + 3.26794896538103e-7*cos(Theta1),
0.19389999999999*(-((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) +
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) +
((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) -
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*cos(Theta4 +
1.570796) - 0.19389999999999*(((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*sin(Theta2 +
1.570796) + (0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) +
((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) -
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*sin(Theta4 +
1.570796) - 0.1349*((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) +
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) -
0.1349*((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) -
(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3) -
0.155*(-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) -
0.155*(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796) -
0.0331000000000189*sin(Theta1) + 5.25486193633259e-8*cos(Theta1) - 5.43133118046386e-8],
[((-((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) +
```

(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) +
((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) -
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*sin(Theta4 +
1.570796) + (((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) +
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) +
((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) -
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*cos(Theta4 +
1.570796))*cos(Theta5) + (3.26794896538138e-7*(-((3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) + (3.26794896538138e-7*sin(Theta1) -
0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) + ((3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) - (3.26794896538138e-7*sin(Theta1) -
0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*cos(Theta4 + 1.570796) -
3.26794896538138e-7*(((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) +
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) +
((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) -
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*sin(Theta4 +
1.570796) - 0.99999999999984*sin(Theta1) - 3.26794896538103e-7*cos(Theta1))*sin(Theta5),
-((-((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) +
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) +
((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) -
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*sin(Theta4 +
1.570796) + (((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) +
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) +
((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) -
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*cos(Theta4 +
1.570796))*sin(Theta5) + (3.26794896538138e-7*(-((3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) + (3.26794896538138e-7*sin(Theta1) -
0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) + ((3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) - (3.26794896538138e-7*sin(Theta1) -
0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*cos(Theta4 + 1.570796) -
3.26794896538138e-7*(((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) +
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) +
((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) -
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*sin(Theta4 +
1.570796) - 0.99999999999984*sin(Theta1) - 3.26794896538103e-7*cos(Theta1))*cos(Theta5),
0.999999999999947*(-((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) +
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) +
((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) -
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*cos(Theta4 +
1.570796) - 0.999999999999947*(((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*sin(Theta2 +
1.570796) + (3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) +
((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) -
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*sin(Theta4 +
1.570796) + 3.26794896538103e-7*sin(Theta1) + 1.06794904403367e-13*cos(Theta1),
0.19389999999999*(-((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) +
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) +
((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) -
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*cos(Theta4 +
1.570796) - 0.19389999999999*(((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*sin(Theta2 +
1.570796) + (3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) +
((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) -
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*sin(Theta4 +
1.570796) - 0.1349*((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) +
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) -
0.1349*((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) -
(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3) -
0.155*(3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) -
0.155*(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796) +
5.25486193633259e-8*sin(Theta1) + 0.0331000000000189*cos(Theta1) + 0.166199999999991],
[((0.999999999999947*sin(Theta3)*sin(Theta2 + 1.570796) - 0.999999999999947*cos(Theta3)*cos(Theta2 +
1.570796))*sin(Theta4 + 1.570796) + (-0.999999999999947*sin(Theta3)*cos(Theta2 + 1.570796) - 0.999999999999947*sin(Theta2

+ 1.570796)*cos(Theta3))*cos(Theta4 + 1.570796))*cos(Theta5) +
(3.26794896538138e-7*(0.999999999999947*sin(Theta3)*sin(Theta2 + 1.570796) - 0.999999999999947*cos(Theta3)*cos(Theta2 +
1.570796))*cos(Theta4 + 1.570796) - 3.26794896538138e-7*(-0.999999999999947*sin(Theta3)*cos(Theta2 + 1.570796) -
0.999999999999947*sin(Theta2 + 1.570796)*cos(Theta3))*sin(Theta4 + 1.570796) - 3.26794896538121e-7)*sin(Theta5),
-((0.999999999999947*sin(Theta3)*sin(Theta2 + 1.570796) - 0.999999999999947*cos(Theta3)*cos(Theta2 +
1.570796))*sin(Theta4 + 1.570796) + (-0.999999999999947*sin(Theta3)*cos(Theta2 + 1.570796) - 0.999999999999947*sin(Theta2
+ 1.570796)*cos(Theta3))*cos(Theta4 + 1.570796))*sin(Theta5) +
(3.26794896538138e-7*(0.999999999999947*sin(Theta3)*sin(Theta2 + 1.570796) - 0.999999999999947*cos(Theta3)*cos(Theta2 +
1.570796))*cos(Theta4 + 1.570796) - 3.26794896538138e-7*(-0.999999999999947*sin(Theta3)*cos(Theta2 + 1.570796) -
0.999999999999947*sin(Theta2 + 1.570796)*cos(Theta3))*sin(Theta4 + 1.570796) - 3.26794896538121e-7)*cos(Theta5),
0.999999999999947*(0.999999999999947*sin(Theta3)*sin(Theta2 + 1.570796) - 0.999999999999947*cos(Theta3)*cos(Theta2 +
1.570796))*cos(Theta4 + 1.570796) - 0.999999999999947*(-0.999999999999947*sin(Theta3)*cos(Theta2 + 1.570796) -
0.999999999999947*sin(Theta2 + 1.570796)*cos(Theta3))*sin(Theta4 + 1.570796) + 1.06794904403373e-13,
0.19389999999999*(0.999999999999947*sin(Theta3)*sin(Theta2 + 1.570796) - 0.999999999999947*cos(Theta3)*cos(Theta2 +
1.570796))*cos(Theta4 + 1.570796) - 0.19389999999999*(-0.999999999999947*sin(Theta3)*cos(Theta2 + 1.570796) -
0.999999999999947*sin(Theta2 + 1.570796)*cos(Theta3))*sin(Theta4 + 1.570796) + 0.134899999999993*sin(Theta3)*cos(Theta2
+ 1.570796) + 0.134899999999993*sin(Theta2 + 1.570796)*cos(Theta3) + 0.154999999999992*sin(Theta2 + 1.570796) +
0.245400000000021], [0, 0, 0, 1]])

Inverse Kinematics:

We have used the second method as seen in classroom lectures, for calculating the Jacobian for the robotic arm. The formula for each column of the jacobian is:

$$J_i = [\delta o_x/\delta q_i \quad \delta o_y/\delta q_i \quad \delta o_z/\delta q_i \quad z_{i-1}^{1} \quad z_{i-1}^{2} \quad z_{i-1}^{3}]^T$$

$$J = [J_1 \quad J_2 \quad J_3 \quad J_4 \quad J_5]_{6\times5}$$

Where, o_x, o_y, o_z are the end effector translation components obtained from the 4th column of the end-effector transformation matrix. z_i^1, z_i^2, z_i^3 are the z-axis of i th joint of the robotic arm, i.e. the 3rd column of joint transformation matrices.

Following is the jacobian that we obtain using the python code:

Matrix([[(0.19389999999999*(-(-3.26794896538138e-7*sin(Theta1) +
0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796) - (-3.26794896538121e-7*sin(Theta1) -
1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3) +
0.19389999999999*((-3.26794896538121e-7*sin(Theta1) -
1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) + (3.26794896538138e-7*sin(Theta1) -
0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*cos(Theta4 + 1.570796) +
(-0.19389999999999*((-3.26794896538138e-7*sin(Theta1) +
0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796) + (-3.26794896538121e-7*sin(Theta1) -
1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3) -

0.19389999999999*((-3.26794896538121e-7*sin(Theta1) -
1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) + (3.26794896538138e-7*sin(Theta1) -
0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*sin(Theta4 + 1.570796) +
(-0.1349*(-3.26794896538138e-7*sin(Theta1) + 0.999999999999947*cos(Theta1))*cos(Theta2 +
1.570796) - 0.1349*(-3.26794896538121e-7*sin(Theta1) -
1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3) +
(-0.1349*(-3.26794896538121e-7*sin(Theta1) - 1.06794904403373e-13*cos(Theta1))*cos(Theta2 +
1.570796) - 0.1349*(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*sin(Theta2 +
1.570796))*sin(Theta3) + (5.06532089634087e-8*sin(Theta1) +
1.65532101825227e-14*cos(Theta1))*sin(Theta2 + 1.570796) + (5.06532089634114e-8*sin(Theta1) -
0.154999999999992*cos(Theta1))*cos(Theta2 + 1.570796) - 5.25486193633259e-8*sin(Theta1) -
0.0331000000000189*cos(Theta1), (-0.19389999999999*((-0.999999999999947*sin(Theta1) -
3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796) - (-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3) -
0.19389999999999*((-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) - (0.999999999999947*sin(Theta1) +
3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*sin(Theta4 + 1.570796) +
(0.19389999999999*((-0.999999999999947*sin(Theta1) -
3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796) - (-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3) +
0.19389999999999*(-(-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) + (0.999999999999947*sin(Theta1) +
3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*cos(Theta4 + 1.570796) +
(-0.1349*(-0.999999999999947*sin(Theta1) - 3.26794896538138e-7*cos(Theta1))*cos(Theta2 +
1.570796) + 0.1349*(-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3) +
(-0.1349*(-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*cos(Theta2 +
1.570796) + 0.1349*(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*sin(Theta2
+ 1.570796))*cos(Theta3) - (-0.154999999999992*sin(Theta1) -
5.06532089634114e-8*cos(Theta1))*sin(Theta2 + 1.570796) + (1.65532101825227e-14*sin(Theta1) -
5.06532089634087e-8*cos(Theta1))*cos(Theta2 + 1.570796),
(0.19389999999999*(-(-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) - (0.999999999999947*sin(Theta1) +
3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) -
0.19389999999999*((-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) - (0.999999999999947*sin(Theta1) +
3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*cos(Theta4 + 1.570796) +
(0.19389999999999*((-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) + (0.999999999999947*sin(Theta1) +
3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) -
0.19389999999999*((-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) - (0.999999999999947*sin(Theta1) +
3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*sin(Theta4 + 1.570796) -
(-0.1349*(-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*sin(Theta2 +
1.570796) - 0.1349*(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*cos(Theta2
+ 1.570796))*sin(Theta3) + (-0.1349*(-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) + 0.1349*(0.999999999999947*sin(Theta1)
+ 3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3),
-(-0.19389999999999*((-1.06794904403373e-13*sin(Theta1) +

3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) + (0.999999999999947*sin(Theta1) +
3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) +
0.19389999999999*((-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) - (0.999999999999947*sin(Theta1) +
3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*sin(Theta4 + 1.570796) +
(-0.19389999999999*((-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) + (0.999999999999947*sin(Theta1) +
3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) -
0.19389999999999*((-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) - (0.999999999999947*sin(Theta1) +
3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*cos(Theta4 + 1.570796), 0],
[(-0.19389999999999*((-0.999999999999947*sin(Theta1) -
3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796) + (-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) -
0.19389999999999*((-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) + (0.999999999999947*sin(Theta1) +
3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3))*sin(Theta4 + 1.570796) +
(0.19389999999999*((-0.999999999999947*sin(Theta1) -
3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796) + (-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) +
0.19389999999999*(-(-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) - (0.999999999999947*sin(Theta1) +
3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3))*cos(Theta4 + 1.570796) +
(-0.1349*(-0.999999999999947*sin(Theta1) - 3.26794896538138e-7*cos(Theta1))*sin(Theta2 +
1.570796) - 0.1349*(-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) +
(-0.1349*(-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*sin(Theta2 +
1.570796) - 0.1349*(0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*cos(Theta2
+ 1.570796))*cos(Theta3) + (-0.154999999999992*sin(Theta1) -
5.06532089634114e-8*cos(Theta1))*cos(Theta2 + 1.570796) + (1.65532101825227e-14*sin(Theta1) -
5.06532089634087e-8*cos(Theta1))*sin(Theta2 + 1.570796) - 0.0331000000000189*sin(Theta1) +
5.25486193633259e-8*cos(Theta1), (-0.19389999999999*((-3.26794896538138e-7*sin(Theta1) +
0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796) - (3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3) -
0.19389999999999*((3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) - (3.26794896538138e-7*sin(Theta1) -
0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*sin(Theta4 + 1.570796) +
(0.19389999999999*((-3.26794896538138e-7*sin(Theta1) +
0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796) - (3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3) +
0.19389999999999*(-(3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) + (3.26794896538138e-7*sin(Theta1) -
0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*cos(Theta4 + 1.570796) +
(-0.1349*(-3.26794896538138e-7*sin(Theta1) + 0.999999999999947*cos(Theta1))*cos(Theta2 +
1.570796) + 0.1349*(3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3) +
(-0.1349*(3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*cos(Theta2 +
1.570796) + 0.1349*(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*sin(Theta2
+ 1.570796))*cos(Theta3) - (-5.06532089634114e-8*sin(Theta1) +

0.154999999999992*cos(Theta1))*sin(Theta2 + 1.570796) + (-5.06532089634087e-8*sin(Theta1) -
1.65532101825227e-14*cos(Theta1))*cos(Theta2 + 1.570796),
(0.19389999999999*(-(3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) - (3.26794896538138e-7*sin(Theta1) -
0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) -
0.19389999999999*((3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) - (3.26794896538138e-7*sin(Theta1) -
0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*cos(Theta4 + 1.570796) +
(0.19389999999999*((3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) + (3.26794896538138e-7*sin(Theta1) -
0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) -
0.19389999999999*((3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) - (3.26794896538138e-7*sin(Theta1) -
0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*sin(Theta4 + 1.570796) -
(-0.1349*(3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*sin(Theta2 +
1.570796) - 0.1349*(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*cos(Theta2
+ 1.570796))*sin(Theta3) + (-0.1349*(3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) +
0.1349*(3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*sin(Theta2 +
1.570796))*cos(Theta3), -(-0.19389999999999*((3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) + (3.26794896538138e-7*sin(Theta1) -
0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) +
0.19389999999999*((3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) - (3.26794896538138e-7*sin(Theta1) -
0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*sin(Theta4 + 1.570796) +
(-0.19389999999999*((3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) + (3.26794896538138e-7*sin(Theta1) -
0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) -
0.19389999999999*((3.26794896538121e-7*sin(Theta1) +
1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) - (3.26794896538138e-7*sin(Theta1) -
0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*cos(Theta4 + 1.570796), 0], [0,
(-0.193899999999979*sin(Theta3)*sin(Theta2 + 1.570796) +
0.193899999999979*cos(Theta3)*cos(Theta2 + 1.570796))*sin(Theta4 + 1.570796) +
(0.193899999999979*sin(Theta3)*cos(Theta2 + 1.570796) + 0.193899999999979*sin(Theta2 +
1.570796)*cos(Theta3))*cos(Theta4 + 1.570796) - 0.134899999999993*sin(Theta3)*sin(Theta2 +
1.570796) + 0.134899999999993*cos(Theta3)*cos(Theta2 + 1.570796) +
0.154999999999992*cos(Theta2 + 1.570796), (-0.193899999999979*sin(Theta3)*sin(Theta2 + 1.570796)
+ 0.193899999999979*cos(Theta3)*cos(Theta2 + 1.570796))*sin(Theta4 + 1.570796) +
(0.193899999999979*sin(Theta3)*cos(Theta2 + 1.570796) + 0.193899999999979*sin(Theta2 +
1.570796)*cos(Theta3))*cos(Theta4 + 1.570796) - 0.134899999999993*sin(Theta3)*sin(Theta2 +
1.570796) + 0.134899999999993*cos(Theta3)*cos(Theta2 + 1.570796),
-(0.193899999999979*sin(Theta3)*sin(Theta2 + 1.570796) -
0.193899999999979*cos(Theta3)*cos(Theta2 + 1.570796))*sin(Theta4 + 1.570796) +
(0.193899999999979*sin(Theta3)*cos(Theta2 + 1.570796) + 0.193899999999979*sin(Theta2 +
1.570796)*cos(Theta3))*cos(Theta4 + 1.570796), 0], [0, 0, -3.26794896538121e-7*sin(Theta1) +
0.999999999999893*cos(Theta1), -3.26794896538121e-7*sin(Theta1) +
0.999999999999893*cos(Theta1), 0.999999999999947*(-((-1.06794904403373e-13*sin(Theta1) +
3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) + (0.999999999999947*sin(Theta1) +
3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) +

((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) - (0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*cos(Theta4 + 1.570796) - 0.999999999999947*(((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*sin(Theta2 + 1.570796) + (0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) + ((-1.06794904403373e-13*sin(Theta1) + 3.26794896538121e-7*cos(Theta1))*cos(Theta2 + 1.570796) - (0.999999999999947*sin(Theta1) + 3.26794896538138e-7*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*sin(Theta4 + 1.570796) - 1.06794904403367e-13*sin(Theta1) + 3.26794896538103e-7*cos(Theta1)], [0, 0, 0.999999999999893*sin(Theta1) + 3.26794896538121e-7*cos(Theta1), 0.999999999999893*sin(Theta1) + 3.26794896538121e-7*cos(Theta1), 0.999999999999947*(-((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) + (3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*sin(Theta3) + ((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) - (3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*cos(Theta3))*cos(Theta4 + 1.570796) - 0.999999999999947*(((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*sin(Theta2 + 1.570796) + (3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*cos(Theta2 + 1.570796))*cos(Theta3) + ((3.26794896538121e-7*sin(Theta1) + 1.06794904403373e-13*cos(Theta1))*cos(Theta2 + 1.570796) - (3.26794896538138e-7*sin(Theta1) - 0.999999999999947*cos(Theta1))*sin(Theta2 + 1.570796))*sin(Theta3))*sin(Theta4 + 1.570796) + 3.26794896538103e-7*sin(Theta1) + 1.06794904403367e-13*cos(Theta1)], [1, 1, 3.26794896538138e-7, 3.26794896538138e-7, 0.999999999999947*(0.999999999999947*sin(Theta3)*sin(Theta2 + 1.570796) - 0.999999999999947*cos(Theta3)*cos(Theta2 + 1.570796))*cos(Theta4 + 1.570796) - 0.999999999999947*(-0.999999999999947*sin(Theta3)*cos(Theta2 + 1.570796) - 0.999999999999947*sin(Theta2 + 1.570796)*cos(Theta3))*sin(Theta4 + 1.570796) + 1.06794904403373e-13]])

Forward Kinematics Validation:

The visualization for forward kinematics validation is present here. The following variations of joint angles were conducted to verify the forward kinematics:

- Joint 5 rotated by 90°
- Joint 4 rotated by -90°
- Joint 3 rotated by -90°
- Joint 2 rotated by -90°
- Joint 1 rotated by -90°

Following are the transformation matrices obtained for respective joint rotations:

The end effector transformation matrix when joint 5 is rotated by 90 degrees is:

```
[[-9.99999995e-01  1.00000093e-04  3.00324800e-08 -3.45979238e-06]
 [-1.00000098e-04 -9.99999950e-01 -2.99346404e-04  1.99199571e-01]
 [ 9.78102359e-11 -2.99346406e-04  9.99999955e-01  7.29199988e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
```

The end effector transformation matrix when joint 4 is rotated by -90 degrees is:

```
[[-9.96932370e-05 -9.99999990e-01 -1.00000063e-04 -2.28556279e-05]
 [ 1.99663235e-04 -1.00019967e-04  9.99999975e-01  3.93157610e-01]
 [-9.99999975e-01  9.96732682e-05  1.99673204e-04  5.35338713e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
```

The end effector transformation matrix when joint 3 is rotated by -90 degrees is:

```
[[-9.96932370e-05 -9.99999990e-01 -1.00000063e-04 -3.63483346e-05]
 [ 1.99663235e-04 -1.00019967e-04  9.99999975e-01  5.28084544e-01]
 [-9.99999975e-01  9.96732682e-05  1.99673204e-04  4.00452206e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
```

The end effector transformation matrix when joint 2 is rotated by -90 degrees is:

```
[[-9.96932370e-05 -9.99999990e-01 -1.00000063e-04 -5.18498846e-05]
 [ 1.99663235e-04 -1.00019967e-04  9.99999975e-01  6.83099993e-01]
 [-9.99999975e-01  9.96732682e-05  1.99673204e-04  2.45452207e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
```

The end effector transformation matrix when joint 1 is rotated by -90 degrees is:

```
[[ 9.99999950e-01 -1.00653683e-04 -2.99346406e-04  3.29995172e-02]
 [ 1.00653687e-04  9.99999995e-01 -1.95664397e-10  1.66200116e-01]
 [ 2.99346404e-04 -2.99346552e-08  9.99999955e-01  7.29199988e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
```

Inverse Kinematics Validation:

The visualization for inverse velocity kinematics validation is present here. The following trajectory has been executed:

Z = a, where a varies w.r.t. time. The values of x, y, and θ_x, θ_y, θ_z remain constant throughout the trajectory execution. The following plot of end-effector is obtained:

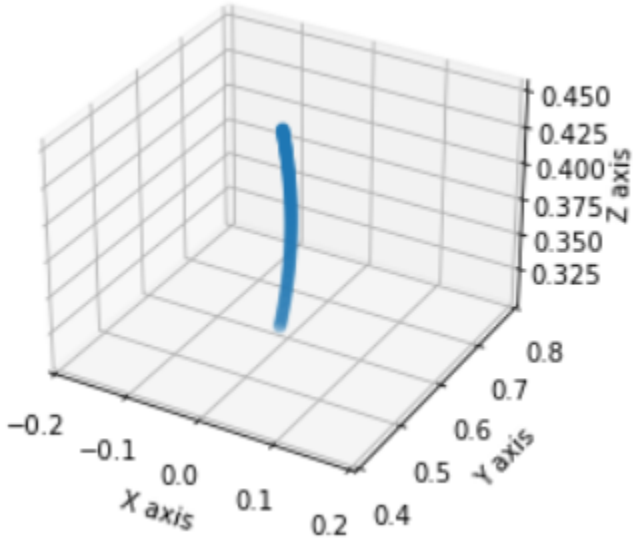

Fig. 2: Plot of trajectory for inverse kinematics validation

Final Task Execution:

Execution of the final task of picking and placing a block is present here. The robot picks a cube of 5 cm x 5 cm x 5 cm from a known point in the plane and places the block at another known point in the plane. This task integrates the robot chassis kinematics, trajectory generation, arm velocity inverse kinematics and gripper actuation.

Kinematics of Chassis:

The kinematics of chassis is calculated using the following matrix multiplication:

$$\begin{pmatrix} wheel\ 1\ vel \\ wheel\ 2\ vel \\ wheel\ 3\ vel \\ wheel\ 4\ vel \end{pmatrix} = \begin{pmatrix} -l-w & 1 & -1 \\ l+w & 1 & 1 \\ l+w & 1 & -1 \\ -l-w & 1 & 1 \end{pmatrix} \begin{pmatrix} w_z \\ v_x \\ v_y \end{pmatrix}$$

$2l = \ distance\ between\ front\ and\ rear\ wheels$

$2w = distance\ between\ left\ and\ right\ wheels$

$w_z = angular\ velocity\ of\ the\ robotic\ chassis$

$v_x = linear\ velocity\ of\ the\ robotic\ chassis\ along\ x - axis$

$v_y = linear\ velocity\ of\ the\ robotic\ chassis\ along\ y - axis$

The below image shows the wheel assignment corresponding to this matrix operation:

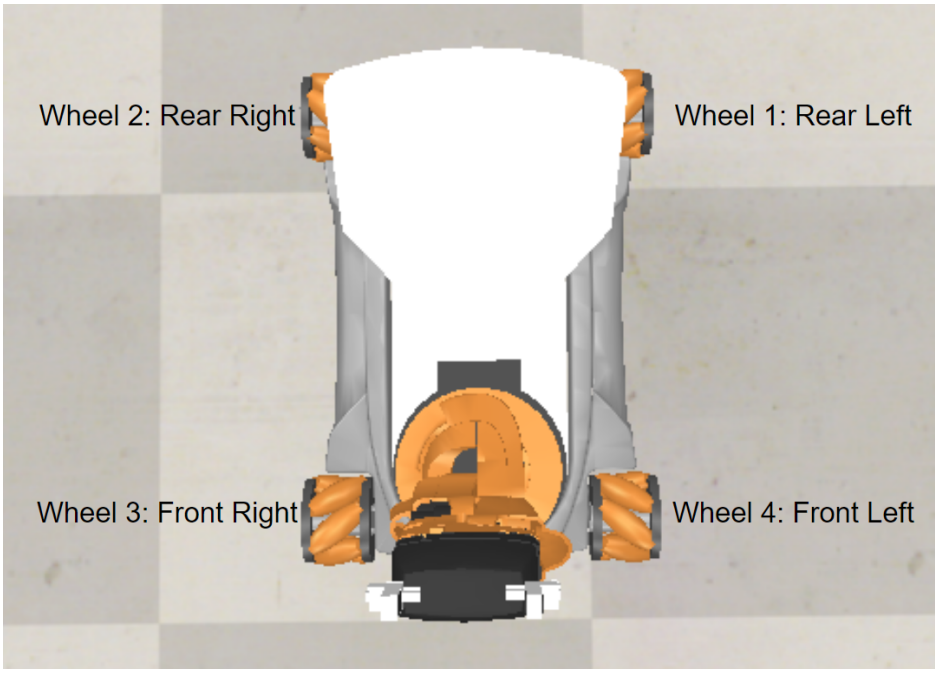

Fig. 3: Wheel assignment for kinematics of robot chassis

Workspace study:

The set of all reachable end points:

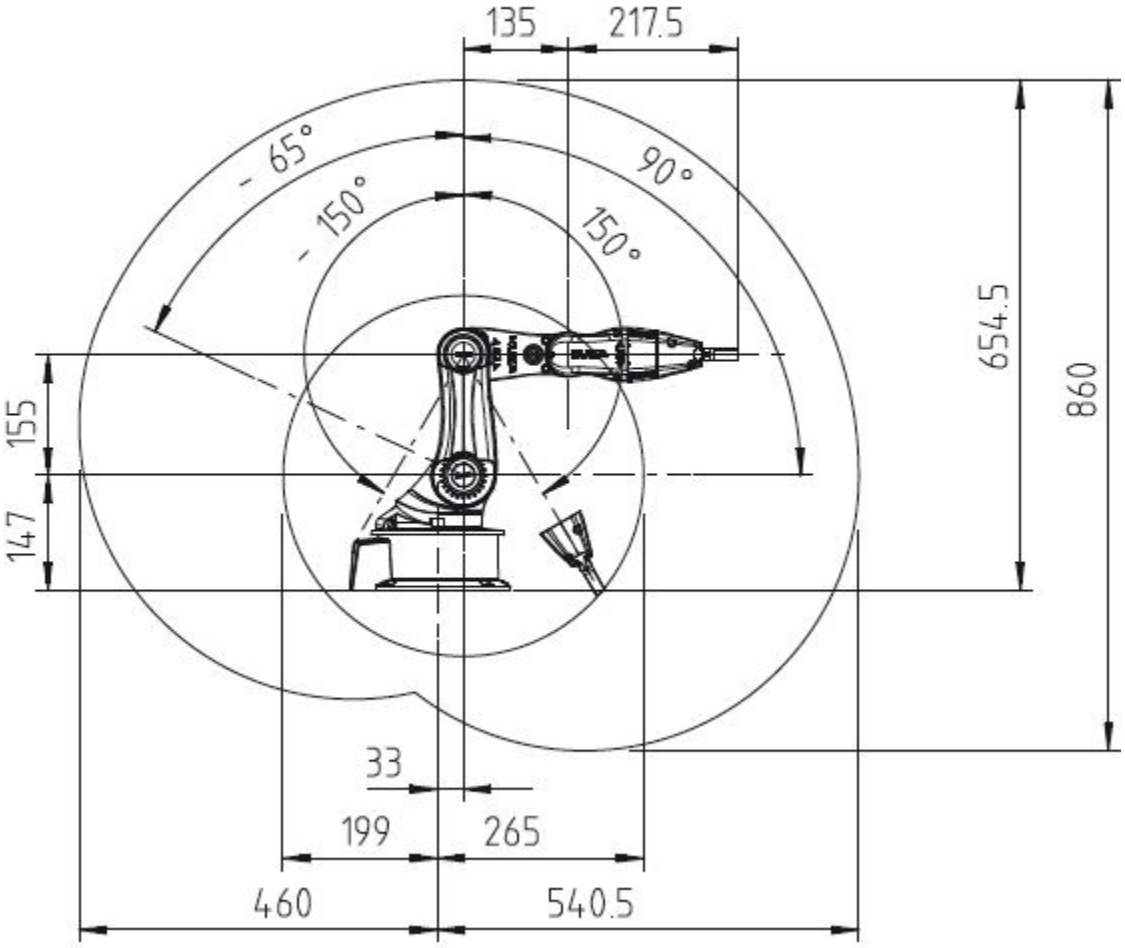

Fig. 4: Workspace study in y-z plane

As we can see in the above diagram, the robot has a small range of motion when the arm is fully extended, with some limitations due to the robot chassis. In the Y-Z plane the robot arm can extend upto 540.5mm in front of the arm and 460mm behind the arm in the Y direction and 860mm in the Z direction. When the first joint is fixed, the range reduces to a total of 464mm with 150 degrees of motion in both directions.

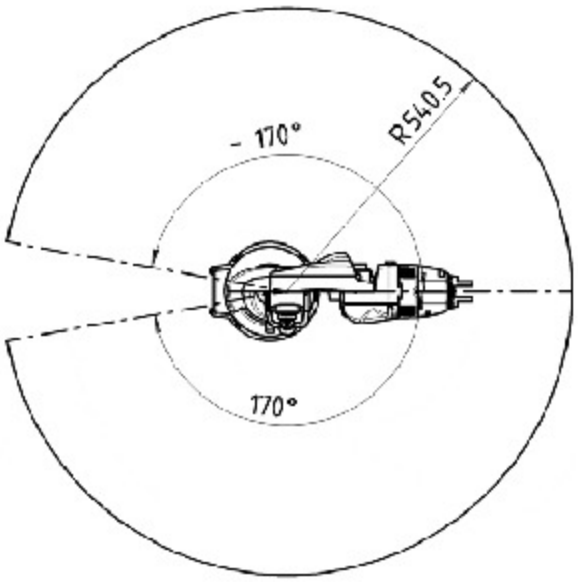

Fig. 5: Workspace study in x-y plane

In the X-Y plane however, the robot can move in 170 degrees in either direction and extends to 540.5 mm.

Assumptions:

- There is no noise in getting the position of the block, which is the goal position for the robot.
- All the links and objects are rigid bodies.
- All the joints have sufficient torque to counter gravity and execute any desired trajectory.
- There is no friction in the joints. We do not model friction as part of this project due to the complexity of friction models and limited time-frame of the project.
- The gripper is rigid, in the sense that when we command the gripper to grip an object, there will not be any motion in any direction other than the gripper's joint.
- It is assumed that there is no slip in the wheels and the ground. Hence, the desired motion will be performed provided that the controller provides the desired input to the wheel motors.

Control Method:

The robot chassis is controlled by a basic P controller (Integral and differential parts of PID were not needed). The arm joints are controlled in open loop mode. The wheels are controlled by a velocity controller and manipulator joints are controlled by a position

controller. CoppeliaSim provides the following function for controlling the velocity of the wheels:

setJointTargetVelocity(int objectHandle, float targetVelocity, float[] motionParams=[])

Here, objectHandle is the handle for the controller joint. targetVelocity is the desired velocity to be given to the robot's wheels. motionParams is an array to specify acceleration, constraints or other parameters if they are to be varied during executing the motion. We do not use this argument for our project.

CoppeliaSim provides the following function for controlling the position of manipulator joints:

setJointTargetPosition(int objectHandle, float targetPosition, float[] motionParams=[])

Here, objectHandle is the handle for the controller joint. targetPosition is the desired position to be given to the robot's manipulator joints. motionParams is an array to specify acceleration, constraints or other parameters if they are to be varied during executing the motion. We do not use this argument for our project.

Visualization:

Visualization was done in CoppeliaSim. The robot - Kuka YouBot was spawned in CoppeliaSim and the necessary python code was added to simulate the desired motion.

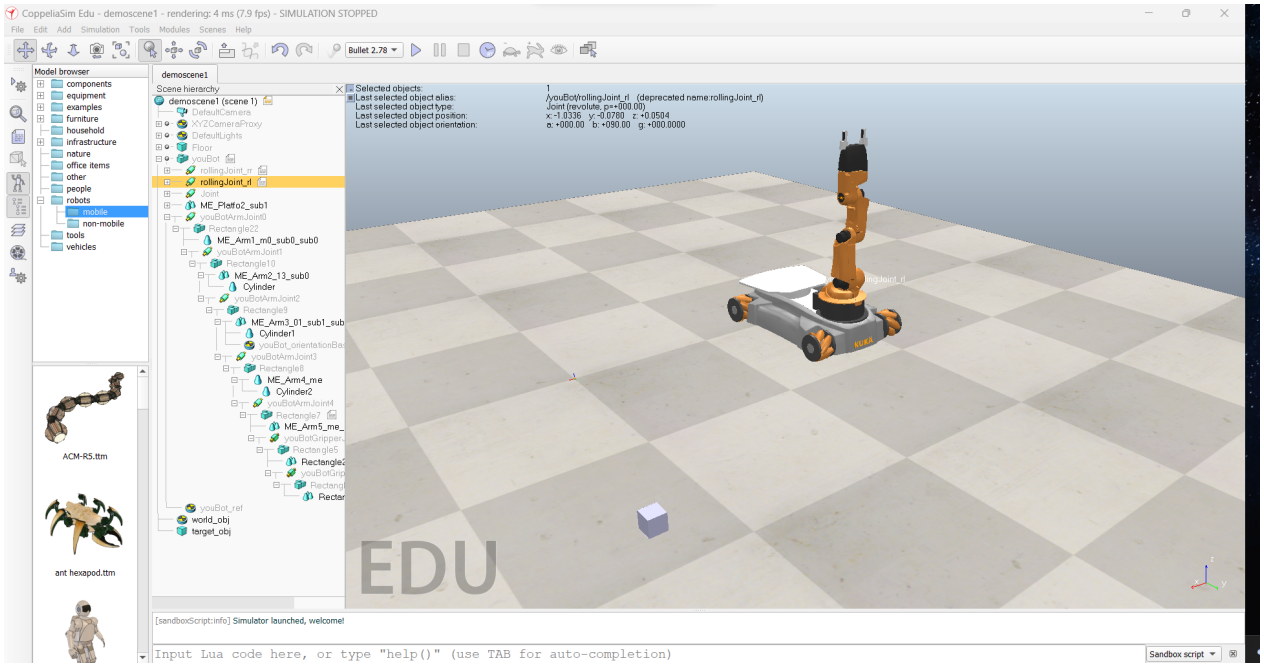

Fig. 4: The robot spawned in CoppeliaSim world with the block to be picked up

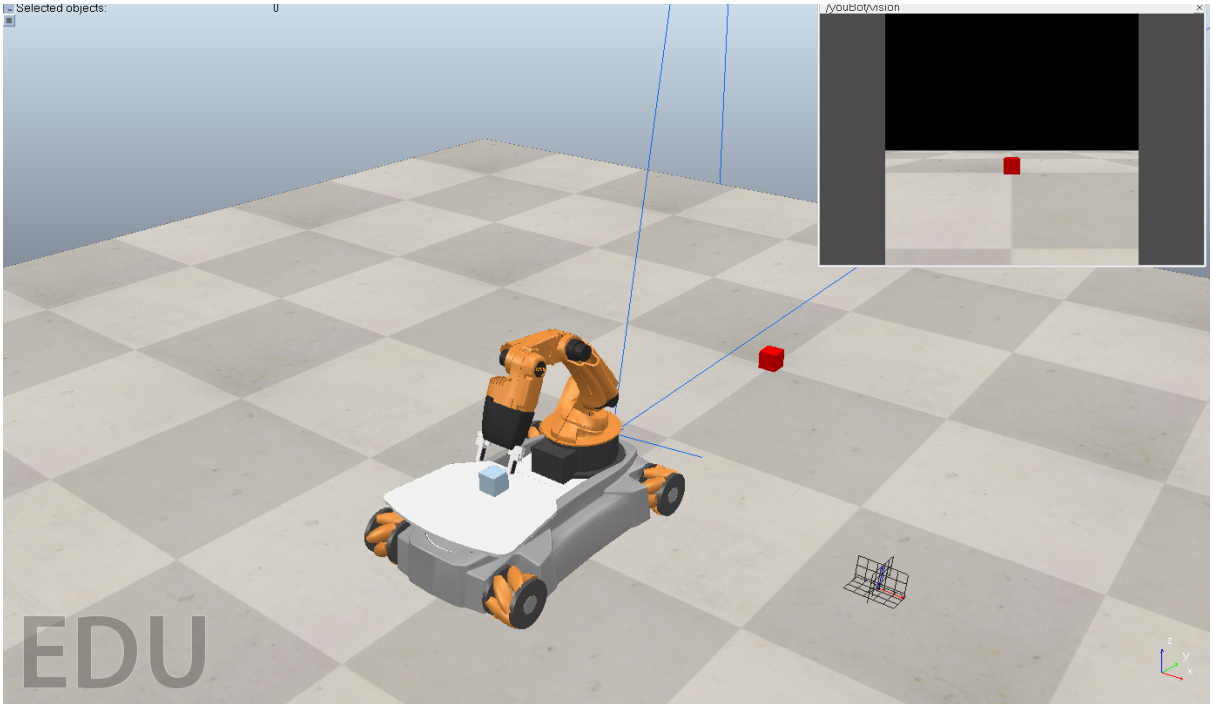

Fig 5: The visualization of robot's camera view

Problems Faced:

- Circular pattern and mate copying during wheel design.
- The selection of faces during the copying of mates and setting the right dimensions for the circular pattern for the mecanum wheel design proved to be one of the major challenges we faced.
- Mating multiple subassemblies during CAD assembly.
- Setting the subassemblies to be flexible and scaling the subassemblies to match the dimensions of the main body were also one of the challenges defaced during assembly.
- Setting the python environment in CoppeliaSim. Python is not natively supported on CoppeliaSim and we had to use a python wrapper just in order to write the script in python.
- Synchronizing python code with CoppeliaSim. We had to enable threading in the config file and use setThreadAutomaticSwitch(True) to enable that python code updates were in sync with the robot's simulated environment.
- Generating the trajectory to the block. We had to split the trajectory into 2 parts to properly align the gripper w.r.t. the block to be picked up.

Lessons Learned:

- Designing models in a CAD environment such as SolidWorks
- Finding out the DOFs of the robot.
- Assigning DH frames and calculating the DH parameters.
- Calculating forward and inverse kinematics of the robot.
- Using the simulation software CoppeliaSim
- Integrating python scripts in CoppeliaSim
- Finding the reference trajectory and joint velocities from the trajectory
- Using CoppeliaSim to run the simulation
- Integrating a vision sensor onto the robot to get a vision feedback and the relative transformation matrix

Conclusions:

We successfully designed and simulated the robot in accordance with the task outline. The DH table, forward and inverse kinematics and the jacobian were calculated. We have validated the forward and inverse kinematics of the robot following which, we generated and executed a trajectory to pick and place a cube. We also integrated a vision sensor into the robot that gives the position of an object in front of it as output.

Future Work:

- Implementing the vision sensor with object detection algorithms into the simulated model of the robot to give the position of the block as the output.
- Further enhancements in vision capabilities of the robot can be made with the help of obstacle avoidance algorithms.
- Optimization of the robot's trajectory time can be done by implementing advanced control algorithms taking into account the motions of the mobile chassis and the manipulator arm.

References:

- http://www.youbot-store.com/wiki/index.php/YouBot_Detailed_Specifications
- https://cyberbotics.com/doc/guide/youbot?version=R2021b

- https://www.researchgate.net/publication/261074766_Unified_Closed_Form_Inverse_Kinematics_for_the_KUKA_youBot
- https://www.academia.edu/23871359/Derivation_of_Kinematic_Equations_for_the_KUKA_youBot_and_implementation_of_those_in_Simulink
- https://www.coppeliarobotics.com/helpFiles/en/regularApi/simGetObjectMatrix.htm
- https://www.coppeliarobotics.com/helpFiles/en/apiFunctions.htm
- https://www.coppeliarobotics.com/helpFiles/en/threadedAndNonThreadedCode.htm
- https://www.researchgate.net/figure/DH-table-for-KUKA-Youbot-manipulator_tbl2_330852442
- https://www.coppeliarobotics.com/helpFiles/en/luaPythonDifferences.html