ENPM662: Introduction to Robot Modelling

PROJECT-2

FINAL REPORT

Modelling & Gazebo Simulation of AgroBot

Raghu Dharahas Reddy Kotla

UID:120378935

Email: raghu17@umd.edu

Sai Dinesh Gelam

UID:120167140

Email: sgelam@umd.edu

Contents

1.Introduction:

In the realm of agriculture, where the industry grapples with persistent challenges like labour shortages, heightened demand for food, and inefficient resource utilization, innovative solutions such as Agrobot have emerged as indispensable assets for farmers. The agricultural sector often faces significant setbacks due to human labour shortages during crucial harvesting periods, impeding the timely gathering of crops. Agrobot addresses this challenge by serving as a mobile fruit-picking robot, offering a reliable and tireless alternative to human labour. This not only ensures a more efficient and timely harvesting process but also mitigates the impact of labour constraints on overall agricultural productivity.

Furthermore, Agrobot contributes significantly to the modernization of agriculture by enhancing productivity, scalability, and precision farming practices. Beyond its role in automating fruit picking, the robot provides real-time data that enables farmers to optimize resource use and reduce environmental impact. The incorporation of robotics in agriculture, exemplified by Agrobot, showcases the transformative potential of technology in revolutionizing traditional farming methods. By alleviating labour constraints, enhancing efficiency, and promoting sustainable practices, Agrobot symbolizes a shift towards a more resilient, productive, and environmentally conscious future for the agricultural sector.

2.Application:

Our project aims to design and implement a robot capable of identifying and selectively picking ripe fruits from trees and collecting them in a bin, streamlining the harvesting process and reducing the reliance on manual labour. This done by integrating manipulator arm (UR-10) to a mobile robot.

3.Robot Type:

The Agrobot is a mobile robot designed for various applications, equipped with specific features that enhance its functionality. Here's an elaboration on its key components and capabilities:

Mobile Platform: The robot is equipped with a 4-wheel drive system, providing stability and manoeuvrability on different surfaces. This configuration allows the robot to navigate through various environments with ease.

Steering System: The Agrobot features front two steering wheels, which enable precise control over its movement. This design facilitates better navigation and enables the robot to navigate around obstacles or tight spaces.

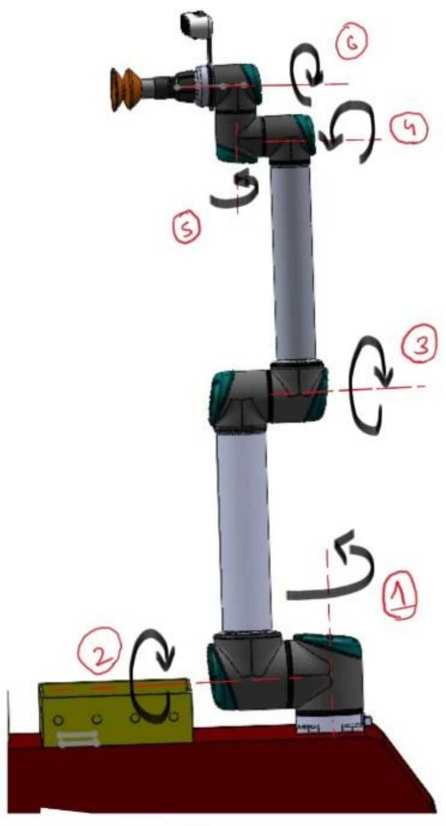
Robotic Arm - UR10: The robot is equipped with a Universal Robots UR10 robotic arm. This robotic arm adds a high degree of flexibility and versatility to the robot's capabilities. The UR10 arm is known for its precision and ability to perform a wide range of tasks.

Degrees of Freedom (DOF): The UR10 robotic arm provides six degrees of freedom, allowing the robot to move its arm in six different directions. This high level of freedom enables the robot to reach and manipulate objects in a three-dimensional space with great precision.

End Effector - Vacuum Gripper: The end effector of the robotic arm is a vacuum gripper. A vacuum gripper is a specialized tool designed for gripping and handling objects using suction. This type of gripper is particularly useful for picking up items of varying shapes, sizes, and materials, making it suitable for a wide range of applications.

Applications: The combination of the mobile platform, steering system, UR10 robotic arm, and vacuum gripper makes the Agrobot suitable for diverse applications. It can be used in industries such as manufacturing, logistics, agriculture, and more. The robot's ability to navigate, manipulate objects, and perform tasks with the vacuum gripper makes it adaptable to different use cases.

4. DOFs and Dimensions:



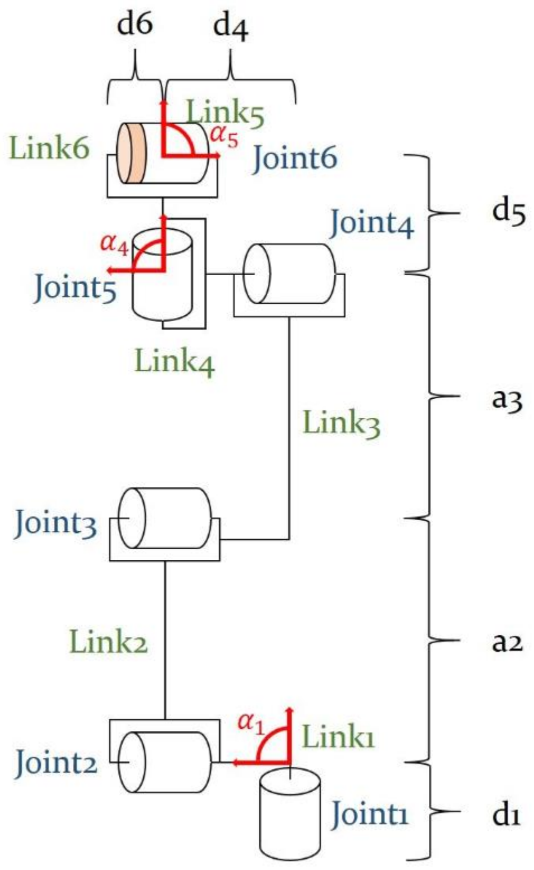
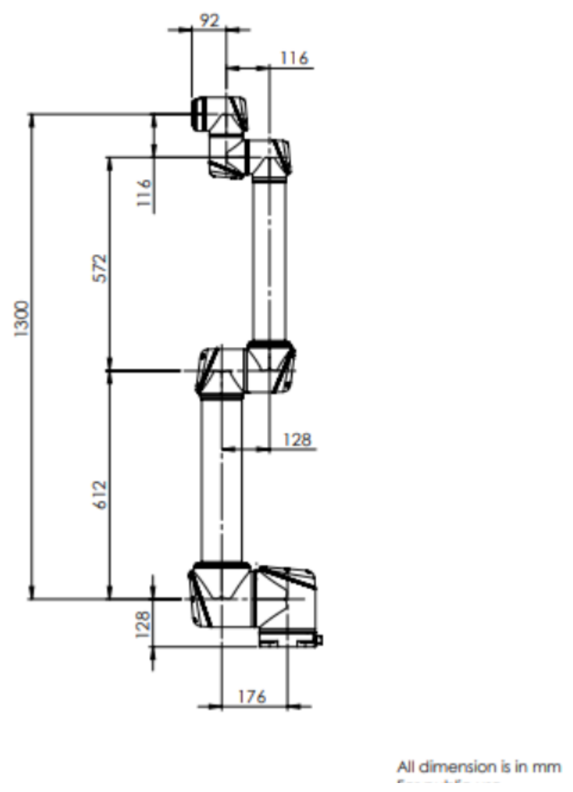


Fig 2: Degree of freedom of manipulator



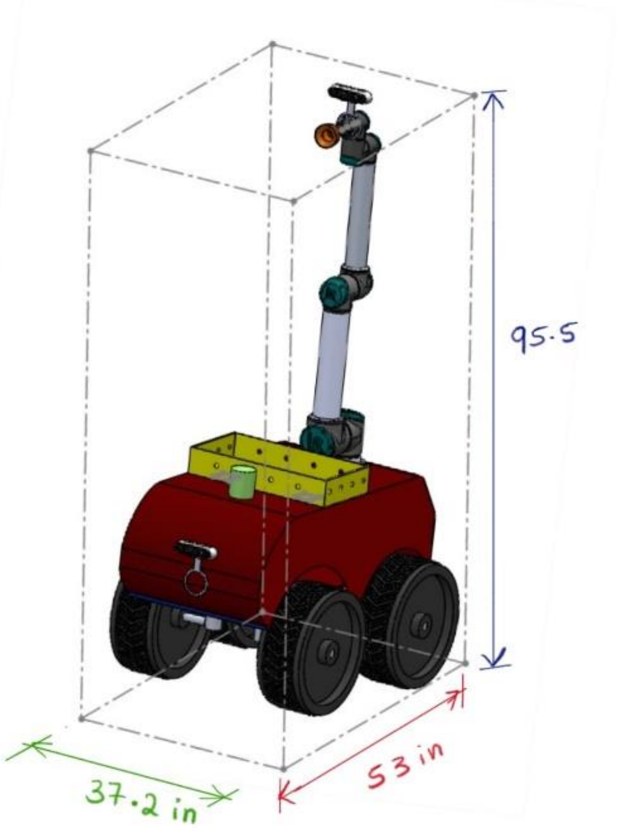


Fig 3: Dimensions of the robot

The above shown manipulator is UR-10 which has 6 Degree of freedom around the axes mentioned above, which facilitates all range of operations. The maximum (box fit) dimensions of the Agrobot are

Height: 95.5 in
Width: 37.2 in
Breath: 53 in
End Effector length: 3.97 in

4.CAD Models

The following are some snaps of Cad designs in SolidWorks (However part and assembly files will be attached separately along with the report)

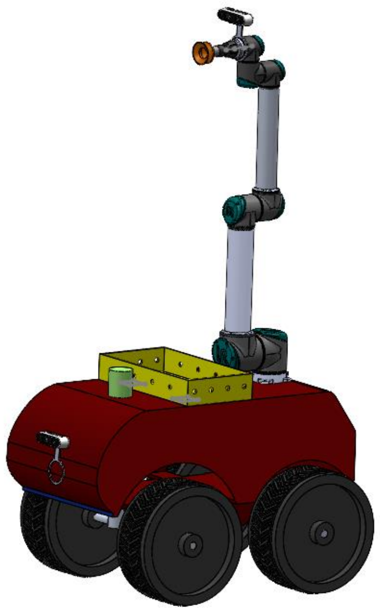


Fig 4: CAD model of the Robot

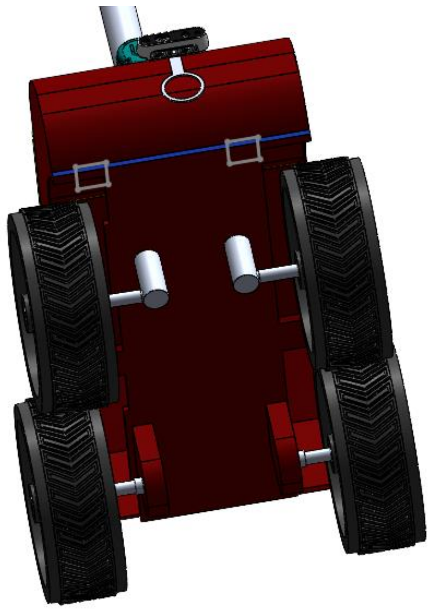
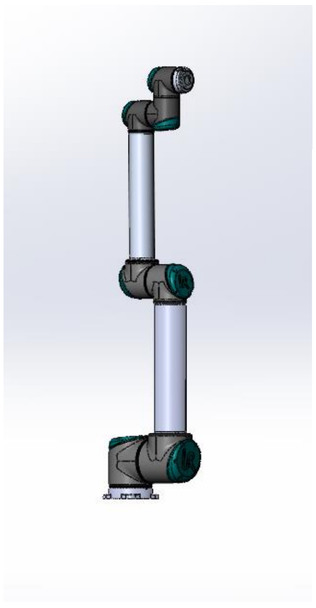


Fig 5: showing wheel joints of the Robot.



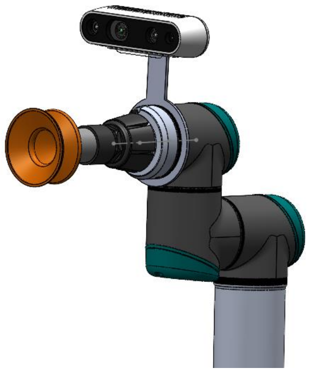


Fig 6: UR10 Manipulator with Vacuum Gripper

5.Frame Assignment:

The Frame Assignment for UR-10 is assigned as follows,

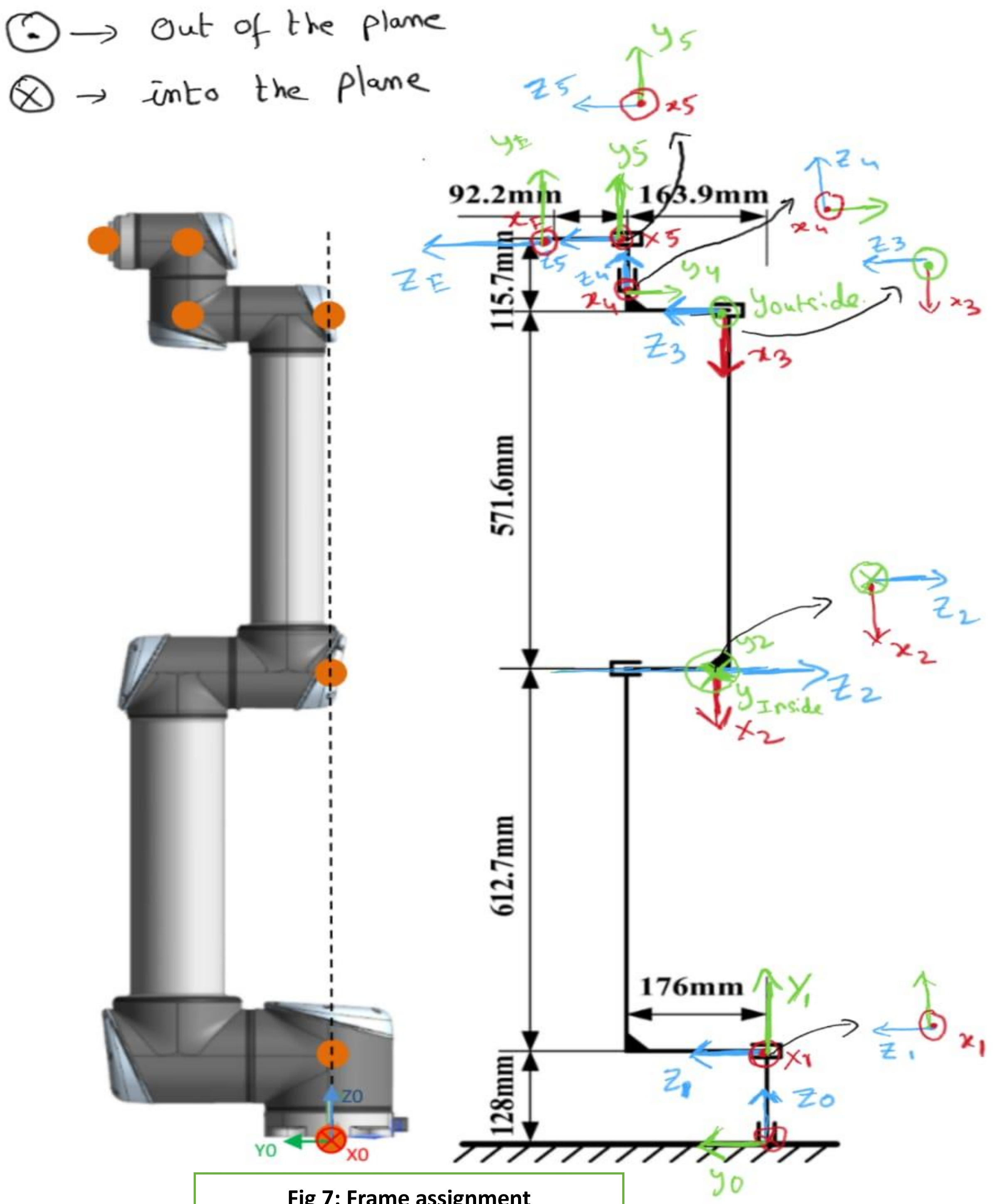


Fig 7: Frame assignment

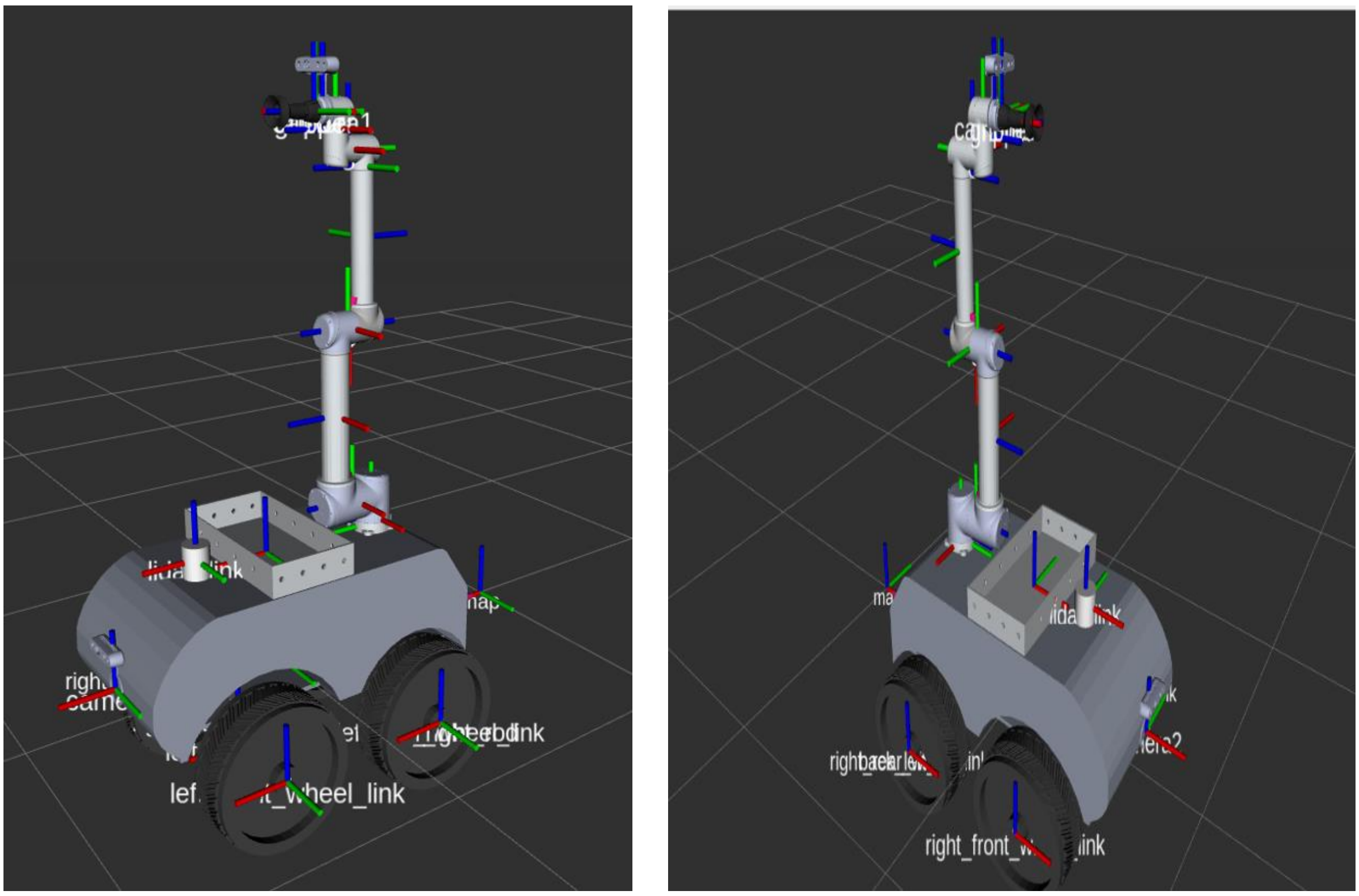


Fig 8: Frame visualisation in Rviz

6.DH parameters:

The following are the DH parameters of UR-10

links	a_i	α_i	d_i	θ_i
0-1	0	90	128	θ_0+180
1-2	-612.7	180	0	$\theta_1 - 90$
2-3	-571.6	180	0	θ_2
3-4	0	-90	163.9	$\theta_3 + 90$
4-5	0	90	115.7	θ_4
5-E	0	0	192.2	θ_5

a_i = distance along x_i from the intersection of the x_i and z_{i-1} axes to o_i.

d_i = distance along z_{i-1} from o_{i-1} to the intersection of the x_i and z_{i-1} axes. If joint i is prismatic, d_i is variable.

α_i = the angle from z_{i-1} to z_i measured about x_i.

θ_i = the angle from x_{i-1} to x_i measured about z_{i-1}. If joint i is revolute, θ_i is variable.

7.Forward Kinematics

Transformation matrix:

Transformation matrix from link to link is given by.

$$T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\propto_i & \sin\theta_i \sin\propto_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\propto_i & -\cos\theta_i \sin\propto_i & a_i \sin\theta_i \\ 0 & \sin\propto_i & \cos\propto_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Applying the above matrix to every successive link to get transformation matrix.

We get the following transformation matrices.

T01

$$\begin{bmatrix} -\cos(\theta_0) & 0 & -\sin(\theta_0) & 0 \\ -\sin(\theta_0) & 0 & \cos(\theta_0) & 0 \\ 0 & 1 & 0 & 128 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T12

$$\begin{bmatrix} \sin(\theta_1) & -\cos(\theta_1) & 0 & -612.7\cdot\sin(\theta_1) \\ -\cos(\theta_1) & -\sin(\theta_1) & 0 & 612.7\cdot\cos(\theta_1) \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T12

$$\begin{bmatrix} \sin(\theta_1) & -\cos(\theta_1) & 0 & -612.7\cdot\sin(\theta_1) \\ -\cos(\theta_1) & -\sin(\theta_1) & 0 & 612.7\cdot\cos(\theta_1) \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T23

$$\begin{bmatrix} \cos(\theta_2) & \sin(\theta_2) & 0 & -571.6\cdot\cos(\theta_2) \\ \sin(\theta_2) & -\cos(\theta_2) & 0 & -571.6\cdot\sin(\theta_2) \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T34

$$\begin{bmatrix} -\sin(\theta_3) & 0 & -\cos(\theta_3) & 0 \\ \cos(\theta_3) & 0 & -\sin(\theta_3) & 0 \\ 0 & -1 & 0 & 163.9 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T45

$$\begin{bmatrix} \cos(\theta_4) & 0 & \sin(\theta_4) & 0 \\ \sin(\theta_4) & 0 & -\cos(\theta_4) & 0 \\ 0 & 1 & 0 & 115.7 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T56

$$\begin{bmatrix} \cos(\theta_5) & -\sin(\theta_5) & 0 & 0 \\ \sin(\theta_5) & \cos(\theta_5) & 0 & 0 \\ 0 & 0 & 1 & 192.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To get the final orientation of the final end effector with respective to the base frame we need to post-multiple each and every above matrix.

$$^{0}_{E}T = {}^{0}_{1}T * {}^{1}_{2}T * {}^{2}_{3}T * {}^{3}_{4}T * {}^{4}_{5}T * {}^{5}_{E}T$$

Note: let's take all the $\boldsymbol{\theta_i}$ = 0 for simplification.

$$ {}^{0}_{E}T = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 356.1 \\ 0 & 1 & 0 & 1428 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The actual matrix will be very big and cannot be pasted here, however it will be generated in the code which will be attached to the file.

Code: https://colab.research.google.com/drive/1si9rpRyce3pMsKl74htI5bIYLOawaZx0?usp=sharing

8.Inverse Kinematics

To find Jacobian matrix I followed method one

$$
\begin{aligned}
J_v &= [J_{v_1} \ldots\ldots J_{v_n}] \\
J_{v_i} &= R_{i-1} \times (o_n - o_{i-1}) \\
J_w &= [J_{w_1} \ldots . J_{w_n}] \\
J_{w_i} &= R_{i-1}
\end{aligned}
$$

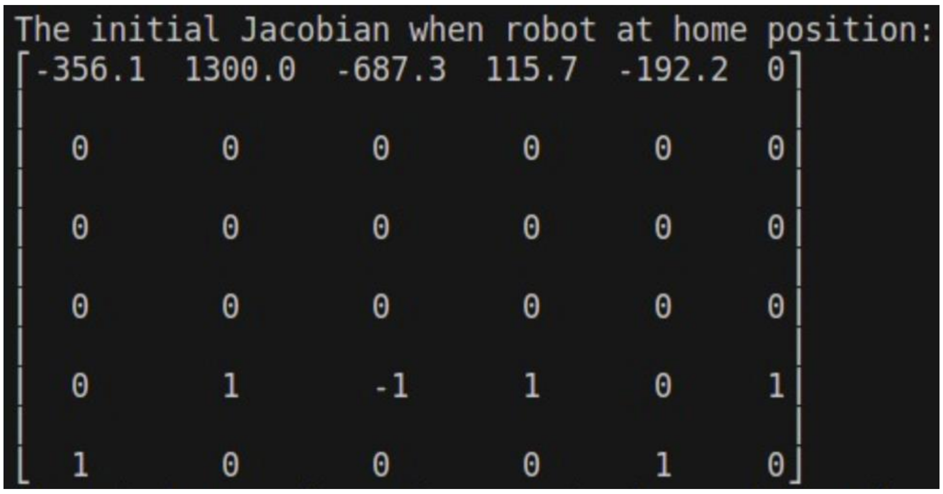
$$ J_i = \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix} $$

$$ J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} $$

So, the following will be the J matrix

$$ J = \begin{bmatrix} R_0^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} (o_6^0 - o_0^0) & R_1^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} (o_6^0 - o_1^0) & R_2^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} (o_6^0 - o_2^0) & R_3^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} (o_6^0 - o_3^0) & R_4^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} (o_6^0 - o_4^0) & R_5^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} (o_6^0 - o_5^0) \\ R_0^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & R_1^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & R_2^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & R_3^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & R_4^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & R_5^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{bmatrix} $$

Since the Jacobian matrix is very large to print here, Jacobian at initial point (**all the $\boldsymbol{\theta_i}$ = 0)** will be shown here



```
The initial Jacobian when robot at home position:
⎡-356.1  1300.0  -687.3  115.7  -192.2  0⎤
⎢                                        ⎥
⎢  0       0       0       0      0     0⎥
⎢                                        ⎥
⎢  0       0       0       0      0     0⎥
⎢                                        ⎥
⎢  0       0       0       0      0     0⎥
⎢                                        ⎥
⎢  0       1      -1       1      0     1⎥
⎢                                        ⎥
⎣  1       0       0       0      1     0⎦
```

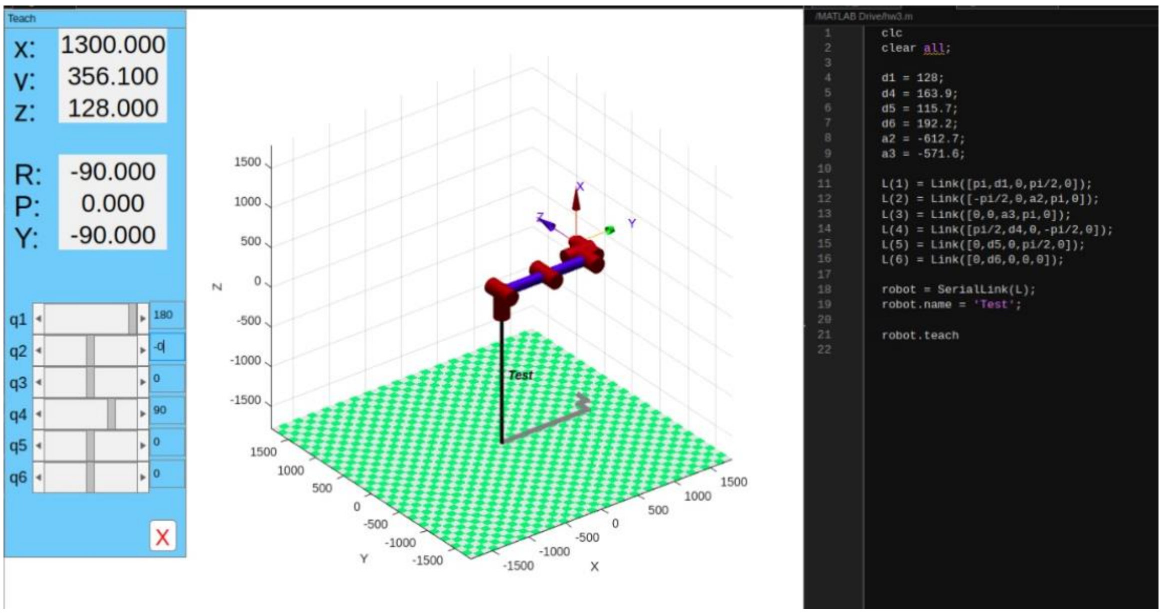
The actual Jacobian matrix will be printed the following code output:
https://colab.research.google.com/drive/1cwnwz7Ii0SKXWeUeUjYhQm3urDsmj8RB?usp=sharing

9.Forward Kinematics Validation

Forward Kinematics validation has been done considering three cases using Peter Corke Toolbox as follows:

Case1: In the initial orientation lets rotate $\boldsymbol{\theta_2 by}$ 90 degrees, we get the following configuration

$$ {}^{0}_{E}T = \begin{bmatrix} 0 & 1 & 0 & 1300 \\ 0 & 0 & 1 & 356 \\ 1 & 0 & 0 & 128 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

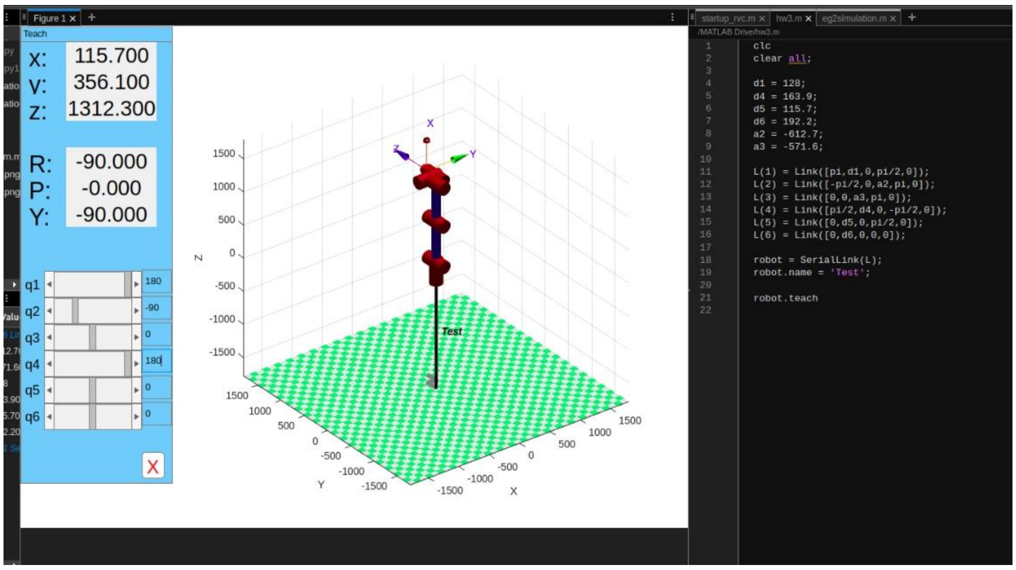


```
X: 1300.000
Y: 356.100
Z: 128.000

R: -90.000
P: 0.000
Y: -90.000

q1 180
q2 -0
q3 0
q4 90
q5 0
q6 0
```

```
/MATLAB Drive/hw3.m
clc
clear all;

d1 = 128;
d4 = 163.9;
d5 = 115.7;
d6 = 192.2;
a2 = -612.7;
a3 = -571.6;

L(1) = Link([pi,d1,0,pi/2,0]);
L(2) = Link([-pi/2,0,a2,pi,0]);
L(3) = Link([0,0,a3,pi,0]);
L(4) = Link([pi/2,d4,0,-pi/2,0]);
L(5) = Link([0,d5,0,pi/2,0]);
L(6) = Link([0,d6,0,0,0]);

robot = SerialLink(L);
robot.name = 'Test';

robot.teach
```

Case2: In the initial orientation lets rotate $\boldsymbol{\theta_5}$ ***by* 90** degrees, we get the following configuration

$$ {}^{0}_{E}T = \begin{bmatrix} 0 & 1 & 0 & 115.7 \\ 0 & 0 & 1 & 356.1 \\ 1 & 0 & 0 & 1312.3 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

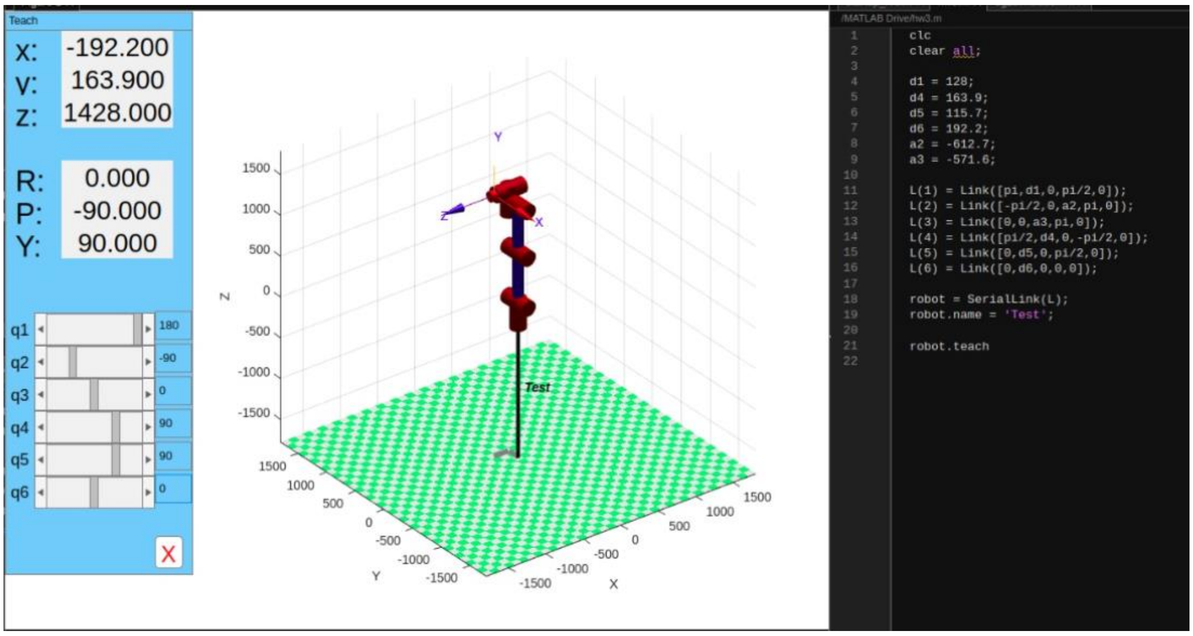


```
X: 115.700
Y: 356.100
Z: 1312.300

R: -90.000
P: -0.000
Y: -90.000

q1 180
q2 -90
q3 0
q4 180
q5 0
q6 0
```

```
startup_rvc.m   hw3.m   eg2simulation.m
/MATLAB Drive/hw3.m
clc
clear all;

d1 = 128;
d4 = 163.9;
d5 = 115.7;
d6 = 192.2;
a2 = -612.7;
a3 = -571.6;

L(1) = Link([pi,d1,0,pi/2,0]);
L(2) = Link([-pi/2,0,a2,pi,0]);
L(3) = Link([0,0,a3,pi,0]);
L(4) = Link([pi/2,d4,0,-pi/2,0]);
L(5) = Link([0,d5,0,pi/2,0]);
L(6) = Link([0,d6,0,0,0]);

robot = SerialLink(L);
robot.name = 'Test';

robot.teach
```

Case3: In the initial orientation lets rotate $\boldsymbol{\theta_4}$ ***by* 90** degrees, we get the following configuration

$$^{0}_{E}T=\begin{bmatrix} 0 & 0 & -1 & -192.2 \\ -1 & 0 & 0 & 163.9 \\ 0 & 1 & 0 & 1428 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



```
clc
clear all;

d1 = 128;
d4 = 163.9;
d5 = 115.7;
d6 = 192.2;
a2 = -612.7;
a3 = -571.6;

L(1) = Link([pi,d1,0,pi/2,0]);
L(2) = Link([-pi/2,0,a2,pi,0]);
L(3) = Link([0,0,a3,pi,0]);
L(4) = Link([pi/2,d4,0,-pi/2,0]);
L(5) = Link([0,d5,0,pi/2,0]);
L(6) = Link([0,d6,0,0,0]);

robot = SerialLink(L);
robot.name = 'Test';

robot.teach
```

10.Inverse Kinematics Validation

In this we will validate our Inverse Kinematics by drawing a circle of radius 100mm in 200 sec in Z-X plane

Defining velocity matrix:

$$\xi = \begin{bmatrix} v_n^0 \\ \omega_n^0 \end{bmatrix}$$

Where in our case Vn will be vector of Vx, Vy, Vz and Wn will be Wx, Wy, Wz

Since our end effector is moving in a single plane of z-x the remaining all the terms of Vn and Wn will be zero except Vx and Vz.

```
E = np.matrix([[x_dot], [0], [z_dot], [0], [0], [0]])
```

So in our case we need to plot circle, by the parametric equations of circle we get the following equation

$$x = r.cos\theta, z = r.sin\theta$$
$$\dot{x} = r.cos\theta.\dot{\theta}, \dot{z} = r.sin\theta.\dot{\theta}$$

Where $d\theta$/dt = angular velocity which is equal to 2*π / 200 since it is required to complete the circle in 200 seconds. So using this method we get velocity matrix as

$$E = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

Now using inverse kinematics method

$$E = J \cdot \dot{q}$$

$$\dot{q} = J^{-1} \cdot E$$

After getting angular velocities of each joint angles, we convert them into angles by numerical integration and then we feed them into final transformation matrix to plot trajectory of the robot end effector.

```
q_dot = J_inv * E
q = q + q_dot * dt
```

On doing this iteratively we get the following plot of circle in 3D and 2D plots

The video of inverse kinematics validation can be found in this link
(Drawing a circle and video is in 2X speed):
https://drive.google.com/file/d/11ZoIrYxgQfb1TJtqK_ZgYFzDIqWeSP6Q/view?usp=sharing

Plots & Graphs:

The following plots have been generated by tracking the end effector trajectory.

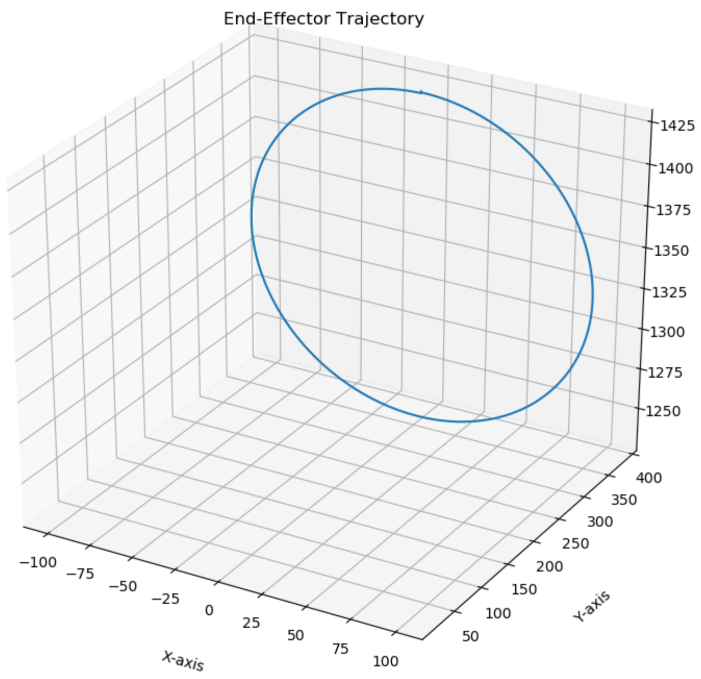


Fig 9: 3D Plot of Circle Trajectory

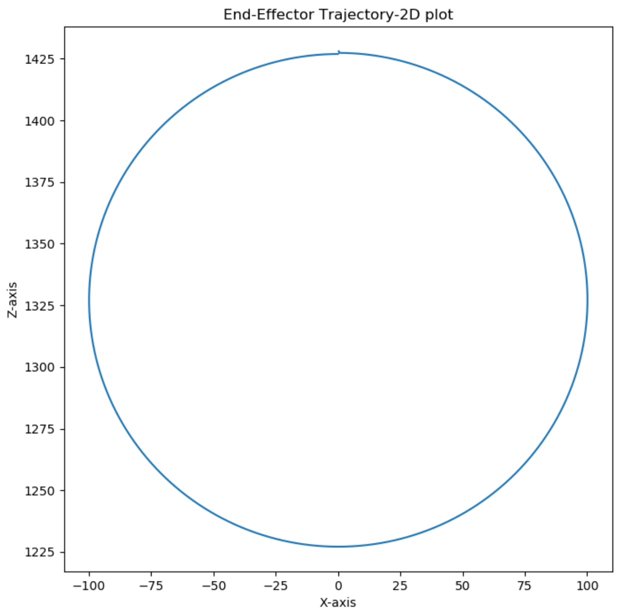


Fig 10: 2D Plot of Circular Trajectory

11.WorkSpace Study:

1. In our project we have used UR10 robotic arm developed by universal robotics. It has a 3Dimensional work space and has maximum reach of 1300mm.
2. The UR10 has 6 rotating joints and has 6 degrees of freedom, allowing it to move in six different directions including rotation and translation along X, Y and Z axes.
3. The maximum pay load of Original UR10 is 10kg (22 lbs) [1]
4. As we mounted UR10 on our mobile platform, So Limits have been imposed to avoid body collisions.

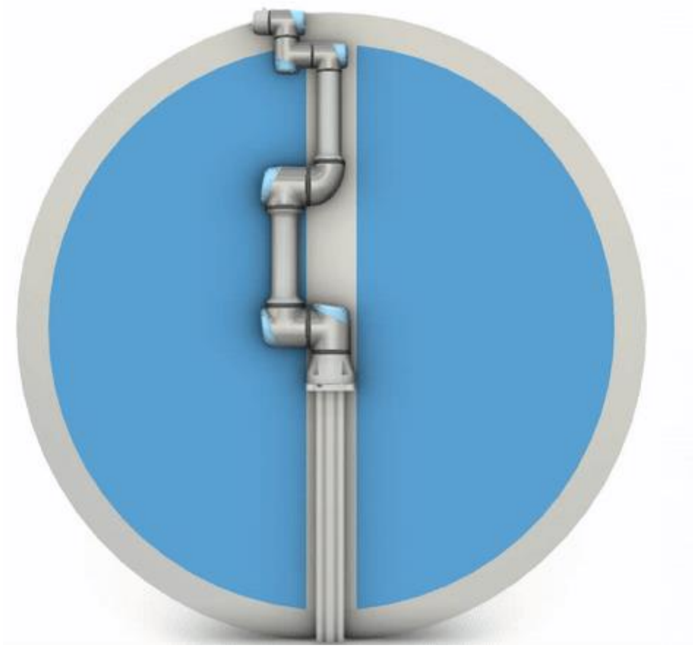
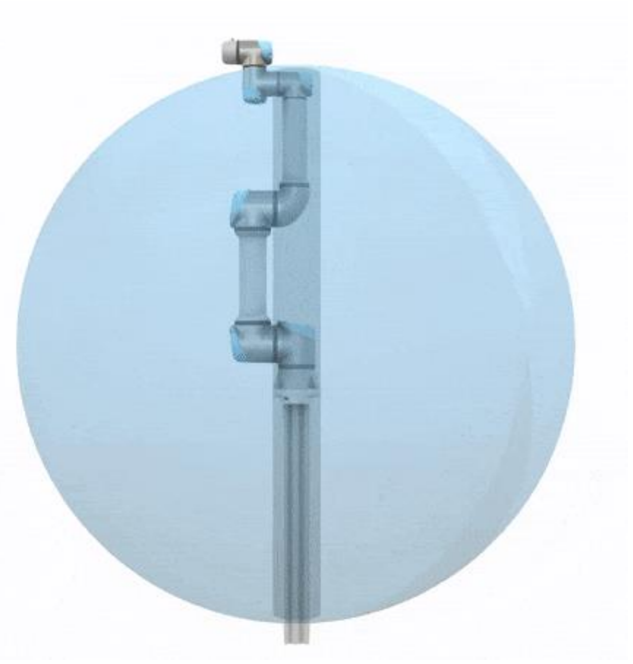



fig 11: Outer workspace limit of UR10 **Fig 12: Inner Workspace limit of UR10**

Source: https://www.universal-robots.com/articles/ur/application-installation/what-is-a-singularity/ [2]

12. Assumptions:

1. The entire Motion of Robotic arm is assumed to be quasi-static and dynamics are neglected in the environment.
2. The Entire simulation environment is stable and no external forces are acting on the system except gravity.
3. The Material of the entire robot is PVC.
4. The coordinates of the objects like Tomatoes and plants spawned in the world are assumed to be known.
5. Despite the integration of Lidar and depth cameras onto the robotic arm, these sensory components remain inactive in the current operational state.

13. Control Methods:

1. Simple Open-Loop Controller: In Open loop controller the input to the system is used to generate an output without the use of feedback. This means that the system does not adjust based on the desired output and the actual output. We have used open-loop controller for our project to control the mobile robot as well as UR10 manipulator because of the simplicity of open loop controller and does not need to compute error and no feedback is available. This leads to few abnormalities in the system.

A. **Mobile Robot Controller:** The control method used in this project is a simple open-loop control. The velocities of the wheels are set to a certain value for a certain period (8 cycles in this case), and then they are set to zero. There is no feedback mechanism in this control method to adjust the velocities based on the actual state of the system.

B. **UR10 Manipulator:** The open loop controller has been used to move the manipulator mounted on Mobile robot with vacuum gripper as its end effector to move from Initial position to Goal position. We have utilized Forward and inverse kinematics to accomplish this task. The Inverse kinematics method in the code is responsible for computing joint angles required to achieve a desired end-effector position and orientation. This method takes the pose of end effector as input and output the joint parameters that will achieve the position and the forward kinematics method takes the joint parameters and output the pose of end effector. As we are not using any feedback loop so the actual position of end effector may not be exact to the required goal position.

14.Gazebo Visualisation:

The following link provides the visualisation of the final working of robot in the gazebo environment:

https://drive.google.com/file/d/1KyryIZxrJzhwPCDGWsSMtdYno1ihYbDe/view?usp=sharing

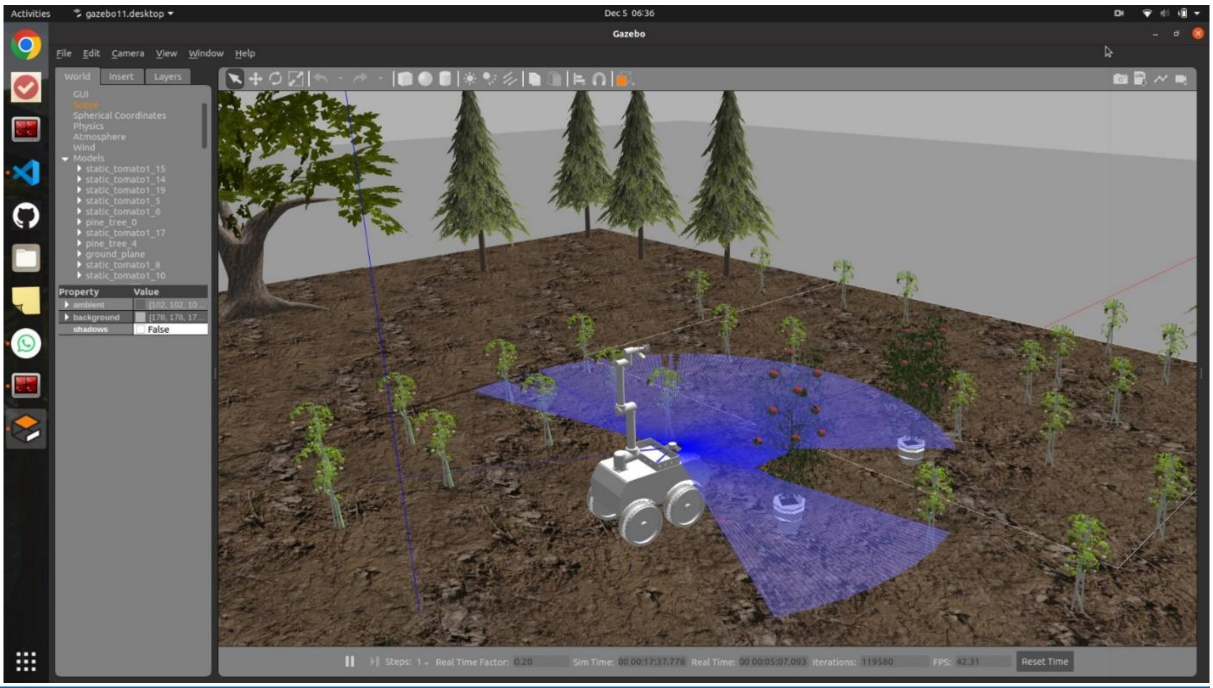


Fig 13: Robot in Gazebo world

Plots & Graphs:

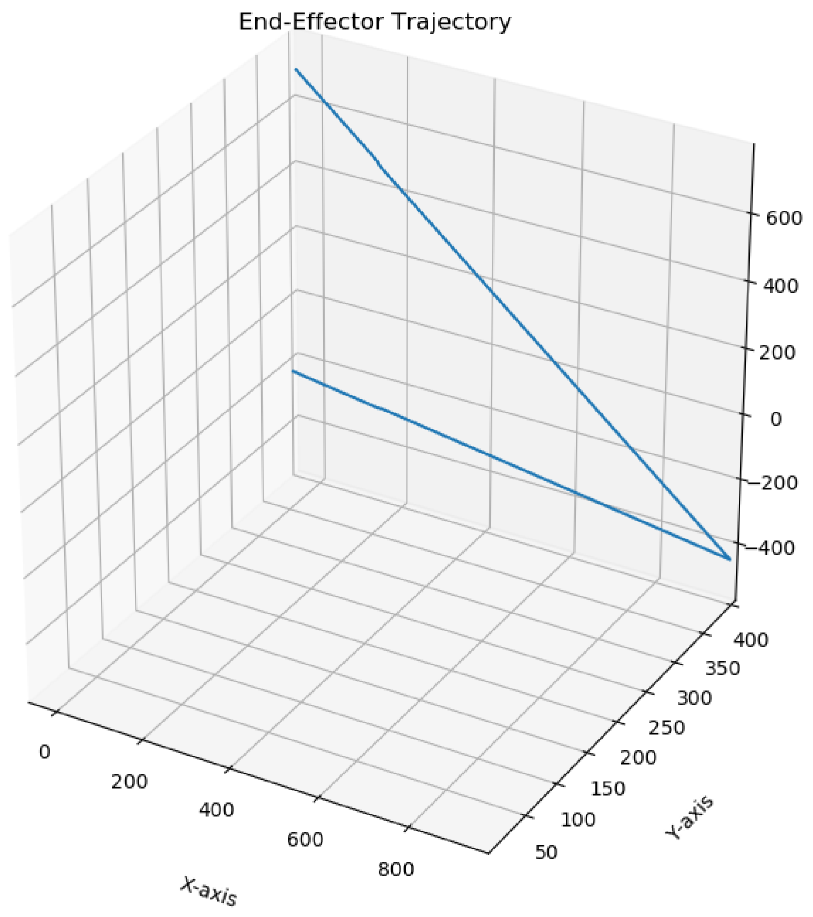


Fig 14: Final Robot Trajectory in 3D Plot

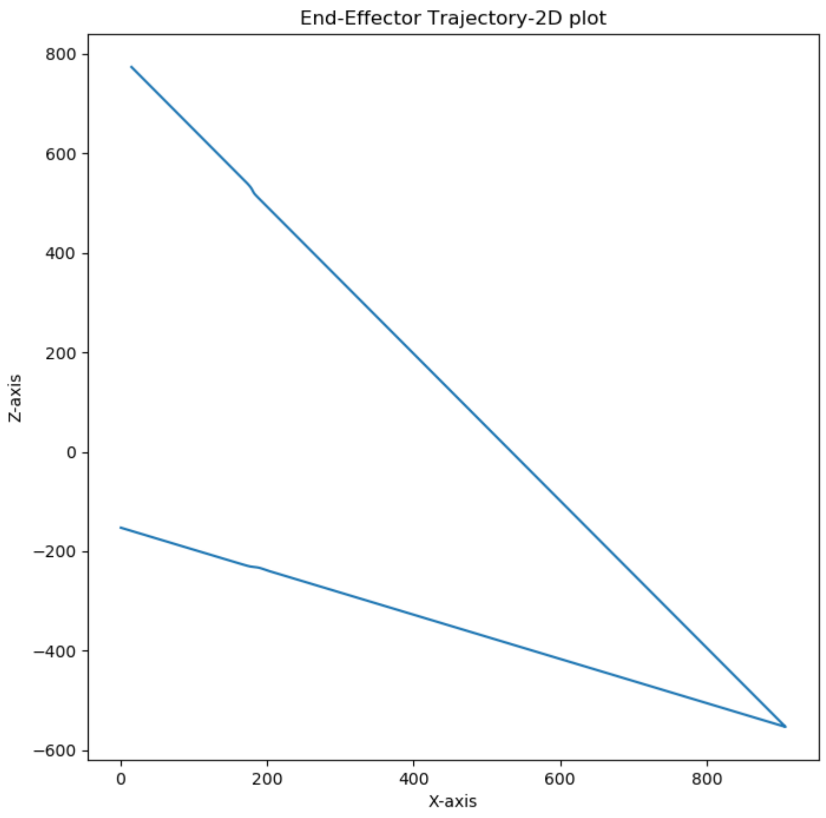


Fig 15: Final Robot Trajectory in 2D Plot

RVIZ Visualisation:

The Following video shows the robot spawned in Rviz (Note: We are not utilizing any sensors mounted in our project):
https://drive.google.com/file/d/1aLQuhBmSBPbe58lVKIE6pYWlBsCZDord/view?usp=sharing

15. Problems Faced:

1. Initially, we have assigned frames to the manipulator without following the DH table and it led to issues while manipulation and in inverse kinematics computation.
2. While controlling the manipulator using inverse kinematics, we have faced issues due to singularities.
3. Creating custom world and saving.

16. Lessons Learned:

1. Generating URDF model from SolidWorks.
2. Using forward and inverse kinematics to operate manipulator.
3. Application of open-loop controller as well as inverse kinematics.
4. Singularities in the robot manipulators and how to solve them, we have implemented one method in our code to check for singularities and solve them by incrementing the angles. But, it is not efficient enough to address all cases.

17.Conclusion:

In conclusion, The AgroBot explores the applications of robotics in agriculture, particularly focusing on labour shortages and inefficient harvesting processes. Featuring the UR10 robotic arm with vacuum gripper. We aimed to pick the fruits or vegetables and drop them in basket. However, we have not utilised any computer vision algorithms to identify the ripe fruits but, we made sure that robot moves to the point and captures the fruit using vacuum gripper and drops it in the basket. Overall, It is a great learning experience and opportunity to explore the potential of robotics.

18.Futurework:

1. Build an efficient algorithm to address the issue of singularities.
2. Implement SLAM for the Mobile robot.
3. Integrating with Depth camera and computer vision algorithm to identify the ripe fruits and vegetables.

19. References:

1. https://store.clearpathrobotics.com/products/universal-robots-ur10
2. https://www.universal-robots.com/articles/ur/application-installation/what-is-a-singularity/
3. https://www.sciencedirect.com/topics/engineering/open-loop-controller#:~:text=An%20open%2Dloop%20controller%2C%20also,where%20feedback%20is%20not%20critical.
4. UR10 Robot manipulator and its symbolic representation