ENPM 662 PROJECT 2
MOBILE FRUIT PICKER ROBOT

Jonathan Crespo (jcrespo)
Robens Cyprien (rcyprien)
Kent-Diens Joseph (kjoseph5)
Hamsaavarthan Ravichandar (rhamsaa)

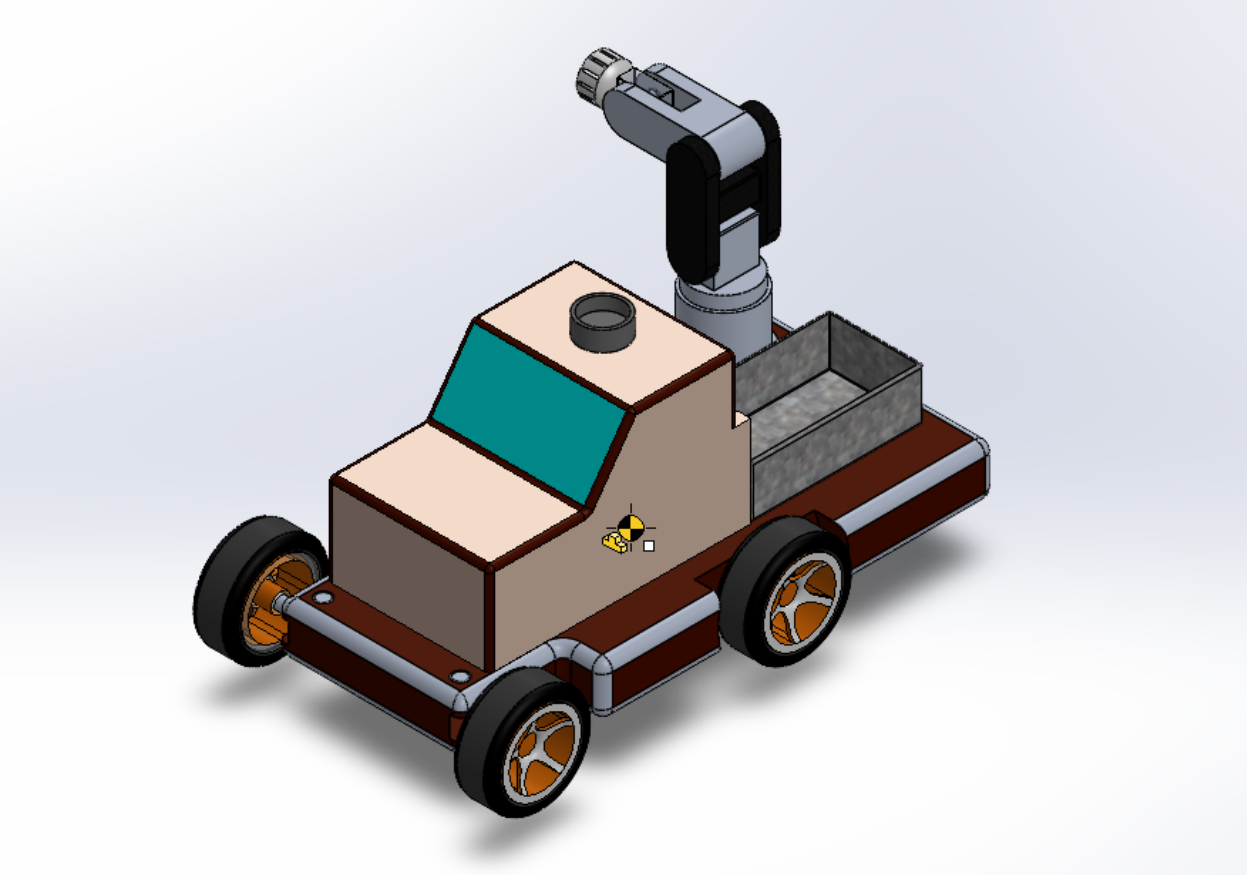


TABLE OF CONTENTS

INTRODUCTION

The agricultural industry is feeling the strain of labor shortages due to stricter policies, rising labor costs, and fewer young people pursuing careers in farming. To combat this problem, many commercial farmers are turning to robots. The automated apple (or any fruit) picking robots are currently capable of picking an apple (fruits) every 3.6 seconds in a high-density orchard and achieve an 80 percent fruit-picking success rate with minimal bruising or quality issues.[1]

This project proposes to design a mobile robot with a manipulator attached to its base that can recognize, fetch apples (fruits) from a cluttered tree setup while traversing through a predefined path in a dense orchard, and simulate it in a software with physical limitations.

ROBOT TYPE, APPLICATION, DOF

The robot consistS of a truck, a robot arm with 4 revolute joints, and a vacuum gripper as the end effector.The truck has two front wheels, each with independent steering, that are passive, and two rear wheels that are powered.The main application of this robot will be to autonomously move to apple trees and use the robot arm vacuum gripper to harvest apples and placed them into a container in the bed of the truck.The truck and robot arm assembly overall has 7 degrees of freedom:

- 1 DOF for the rear-wheels, since they are connected by a one drive shaft.
- 2 DOF (1 for each wheel) for the two front wheels since they have independent steering.
- 4 DOF for the robot arm, since it has four revolute joints.

ASSUMPTIONS

- The gazebo world is a decent approximation of the real world with reasonably certain physical limitations.
- The Gazebo controllers will be able to control all the robot components perfectly
- Trees will have specific configurations and geometries as well the apples for the proposed tasks
- The manipulator arm can pick up whatever it detects with the same gripper
- The robot will be exposed and simulated in a pre-constructed environment that is defined by us, that has trees with specific shapes and the fruits/apples set in specific locations.
- Our camera and lidar will be able to detect the trees and apples and report their location.
- The coordinates for the location of trees/apples in the gazebo are provided, given

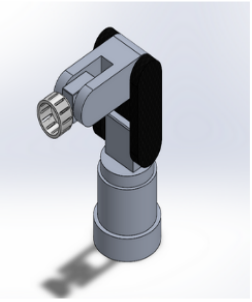
that the positions were already detected and determined using lidar/camera.

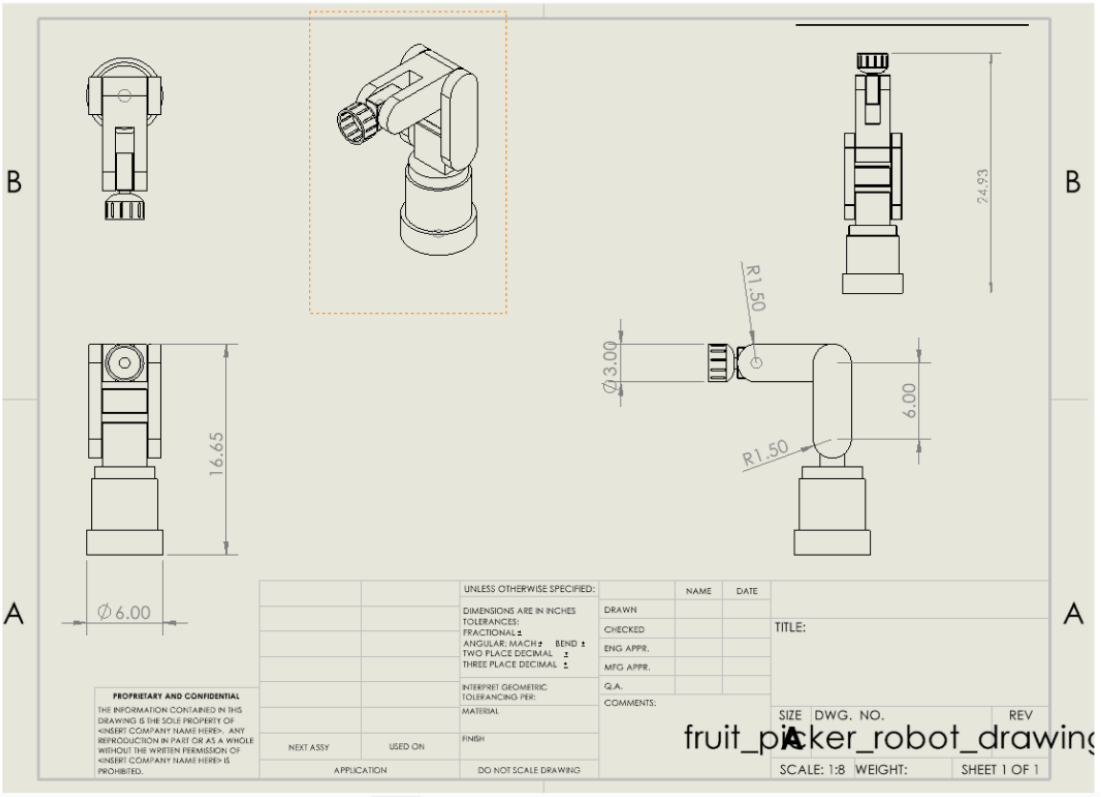
- The path for reaching the destination/target location is pre-defined and fed to the robot at the time of traversal.

CAD MODELS & DIMENSIONS

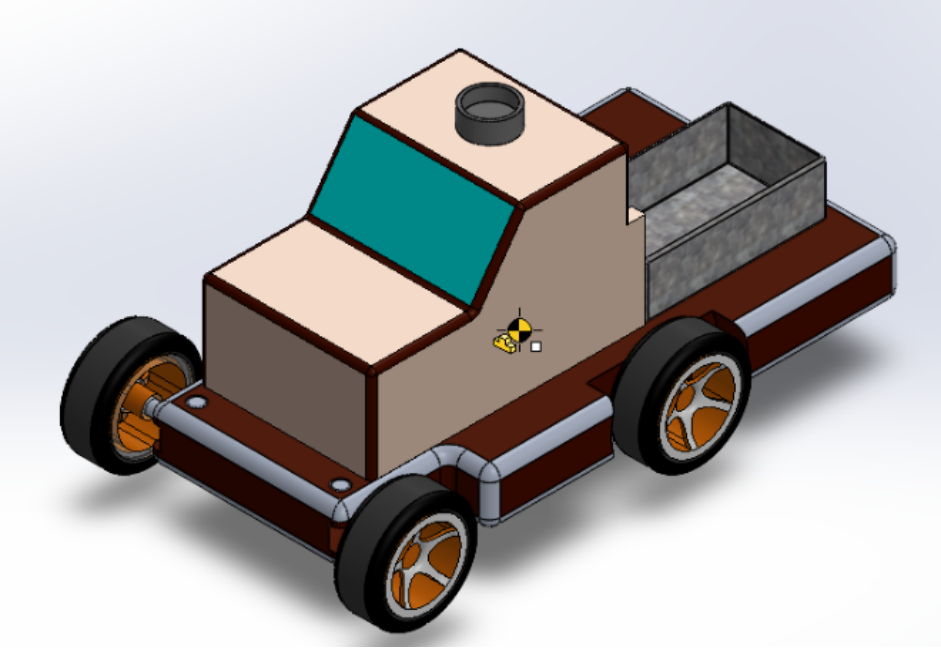
https://drive.google.com/drive/folders/1gbCOuZfvvhksql2FFSNNTEFmud0X-PkQ?usp=sharing

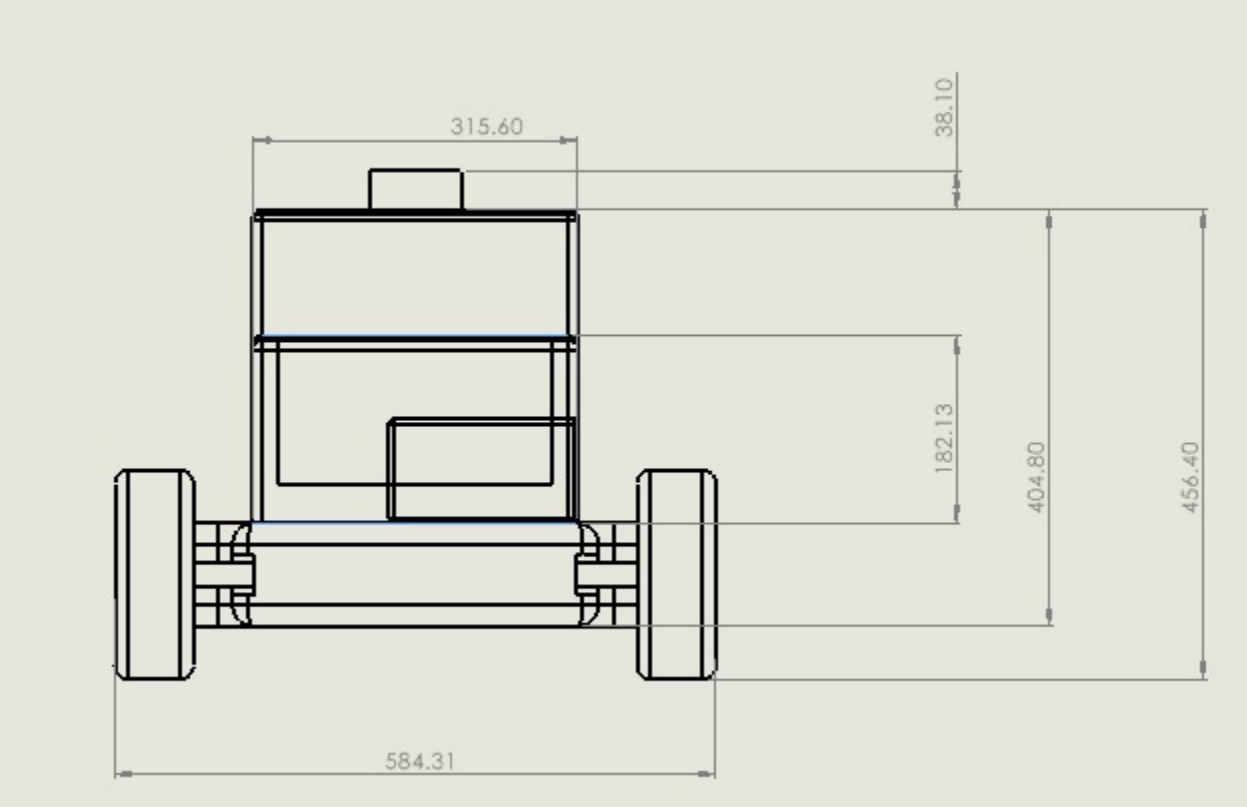
The robot arms base is cylindrical and has a 6 inch diameter. The overall height of the robot arm, when fully extended, is 24.93 inches. The images below show its additional dimensions.

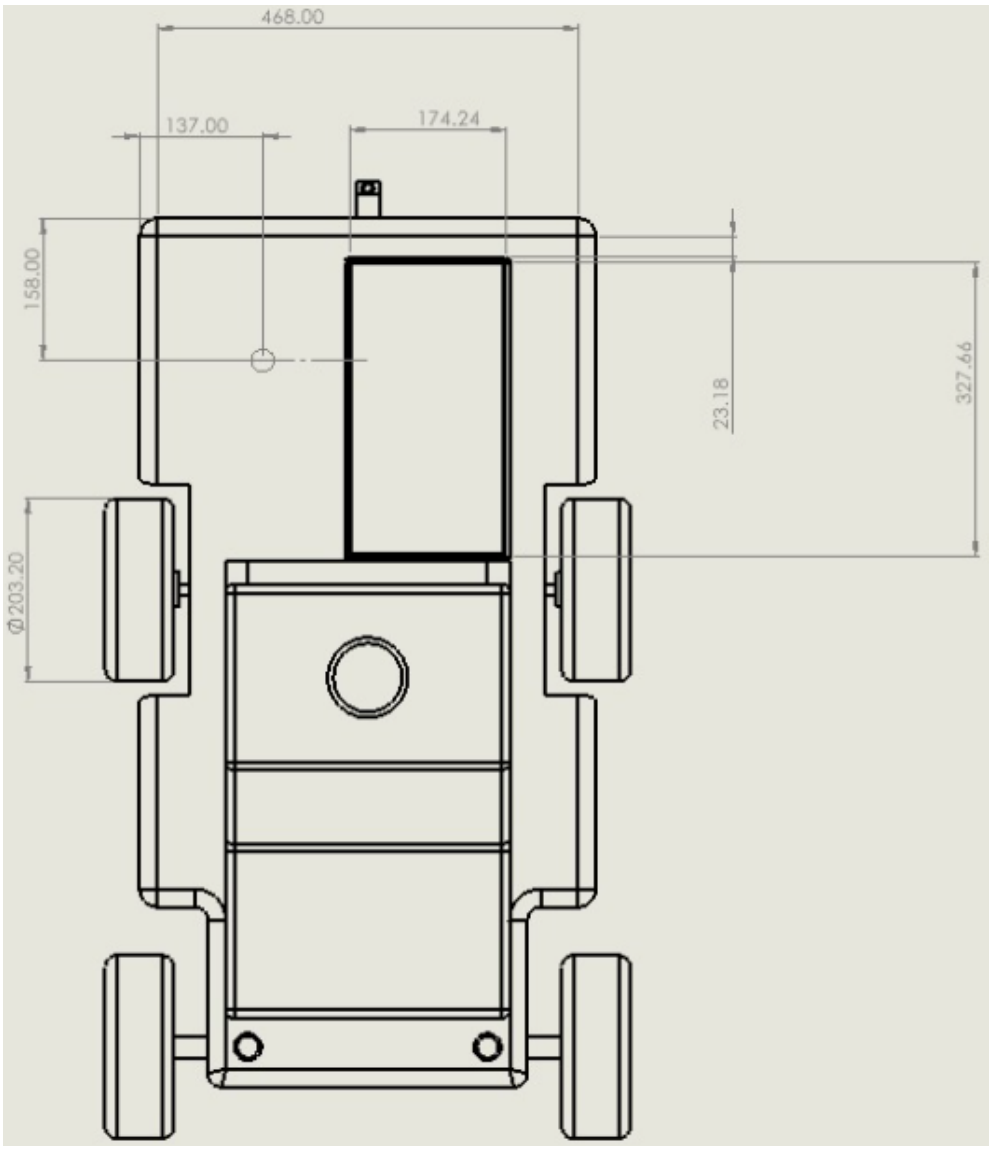


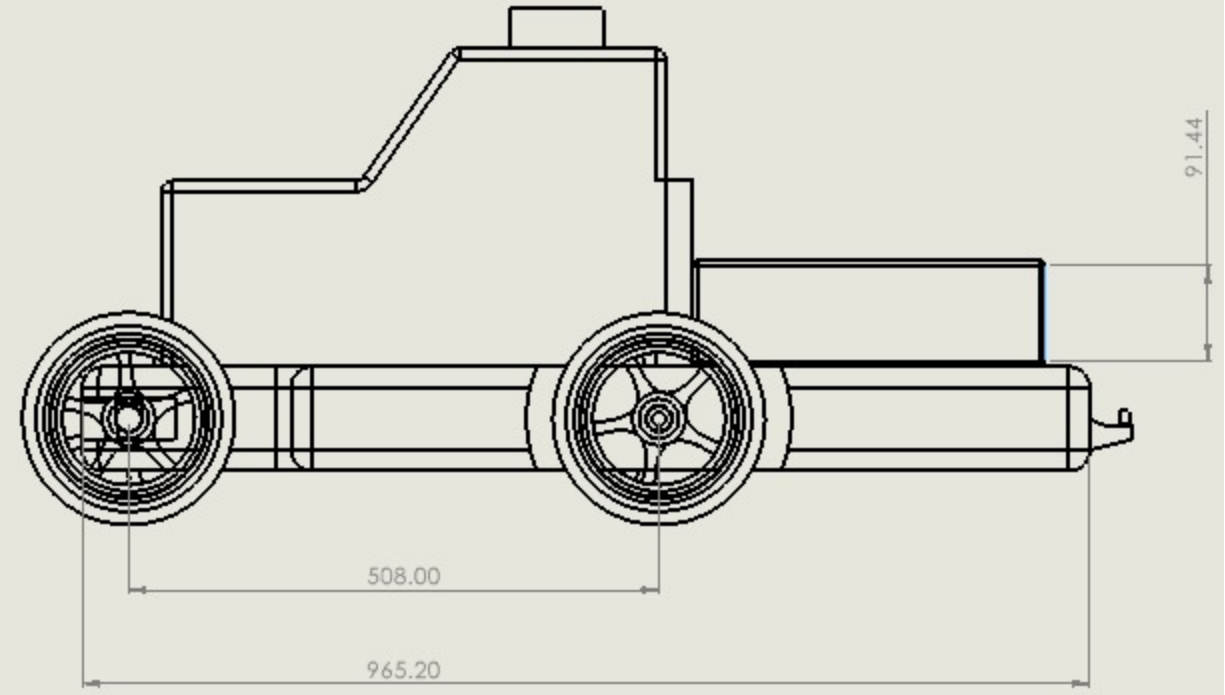


The truck's overall length is 965.2 mm (38 in), its width is 584.31 mm (23 in), and its overall height is 456.4 mm (17.97 in). The fruit container is 327.66 mm (12.9in) in length and 174.24 mm (6.86 in) in width. The following are drawings of the truck and the container.

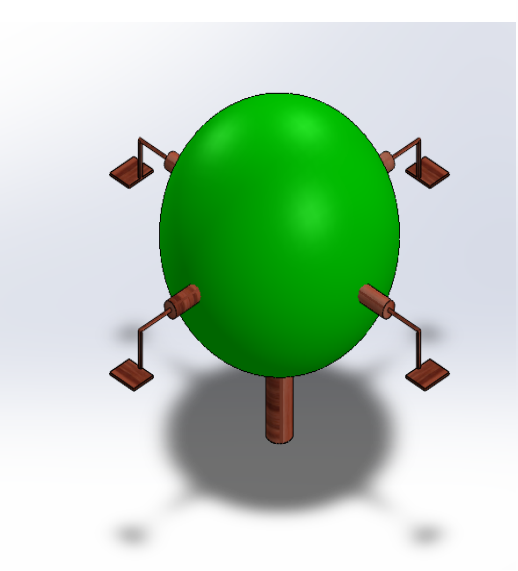


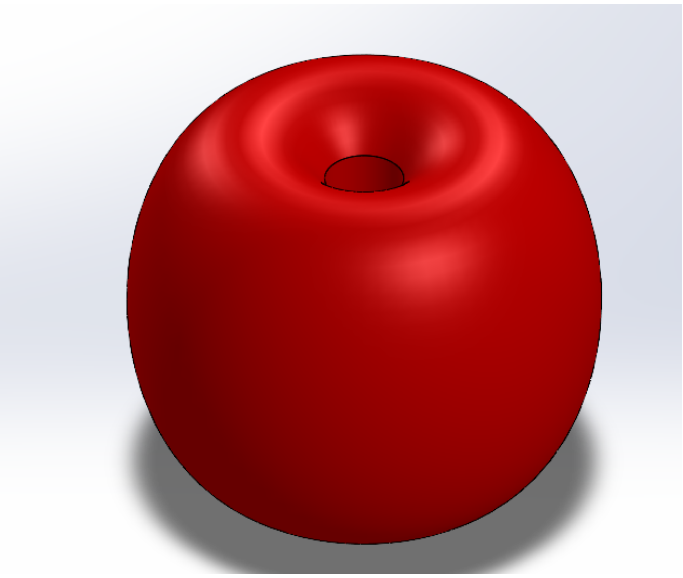

315.60
38.10
182.13
404.80
456.40
584.31

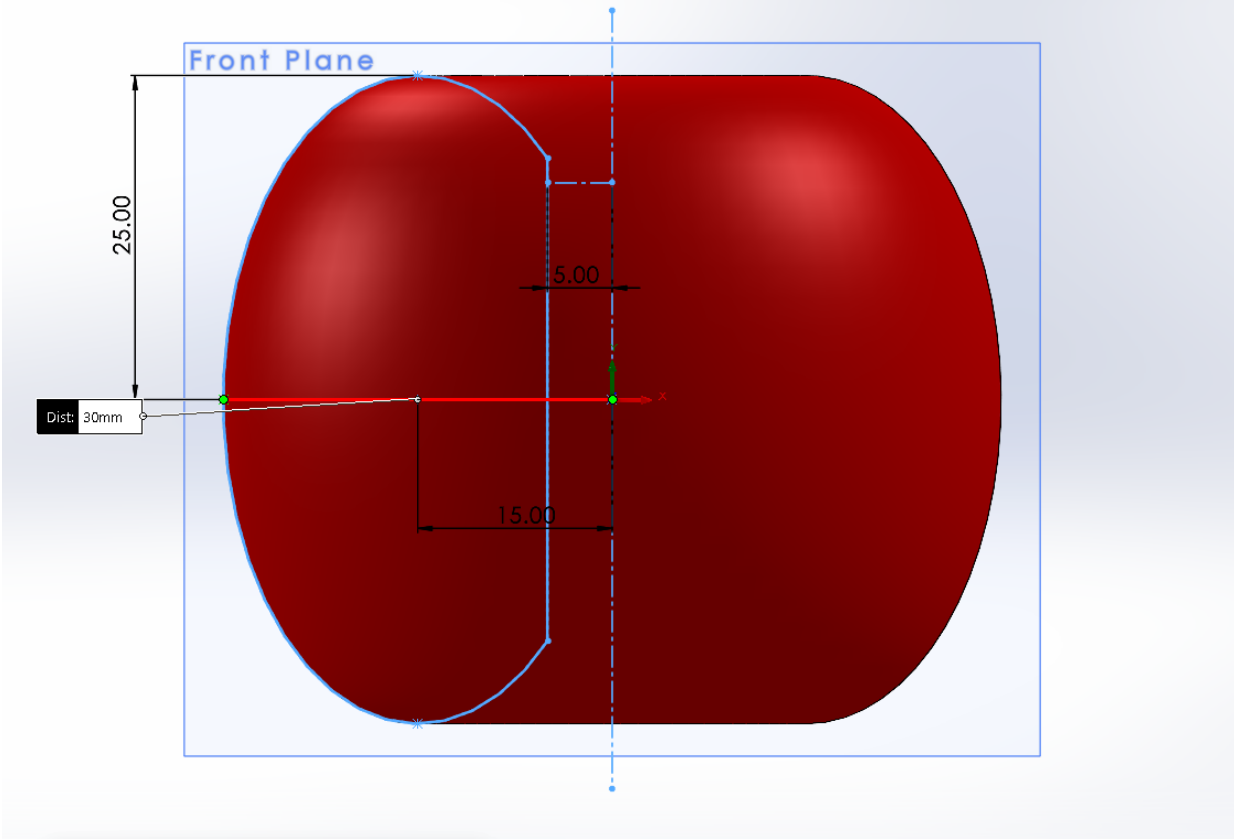

468.00
137.00
174.24
158.00
23.18
327.66
Ø203.20

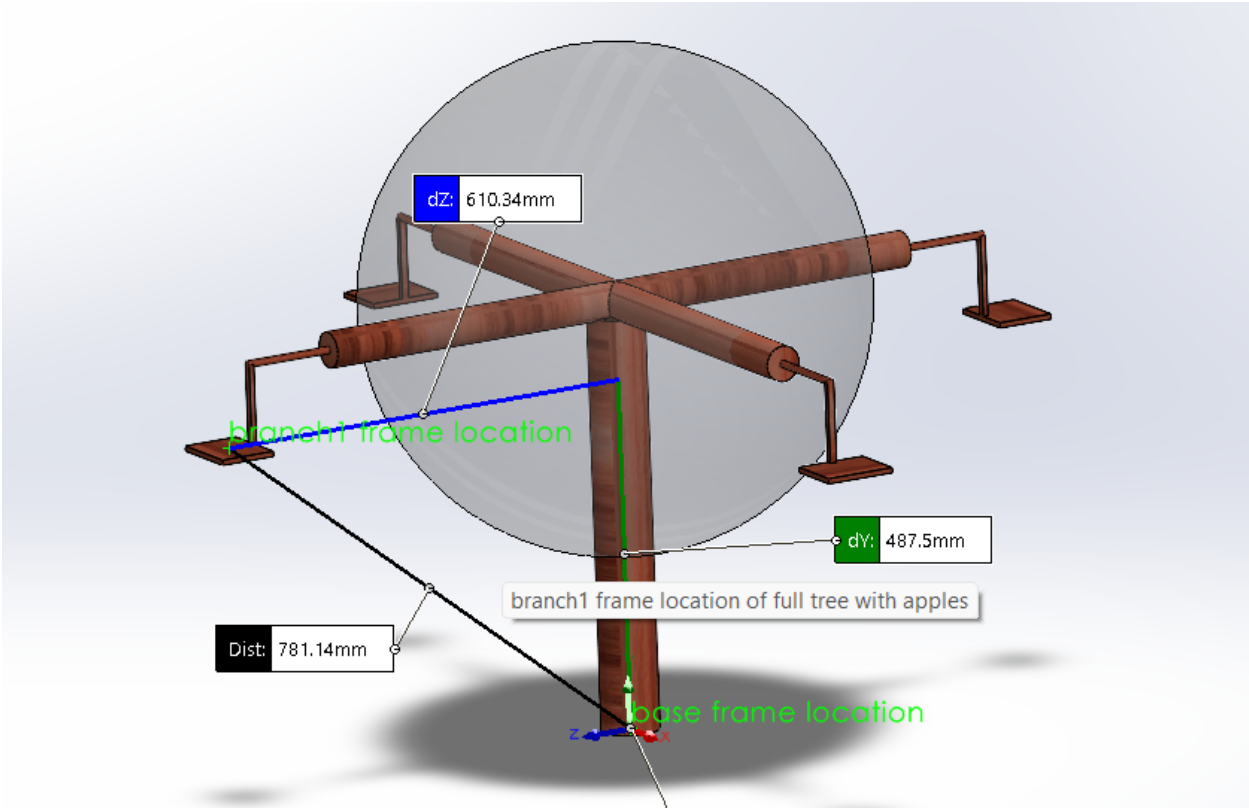


The images below are the CAD models used for the apple tree, and the apple fruit.





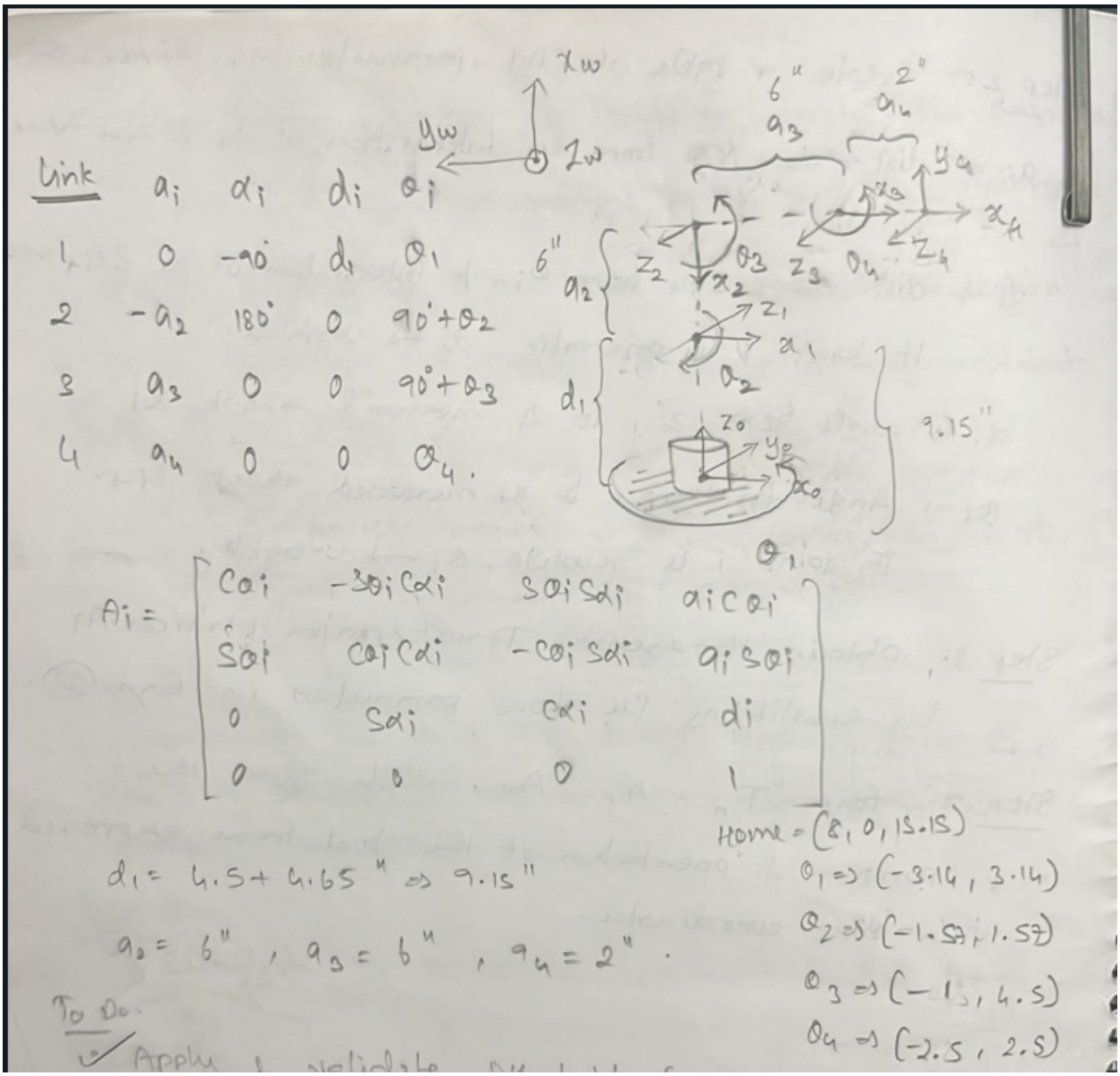

Front Plane
25.00
5.00
15.00
Dist: 30mm


dZ: 610.34mm
branch1 frame location
dY: 487.5mm
branch1 frame location of full tree with apples
Dist: 781.14mm
base frame location

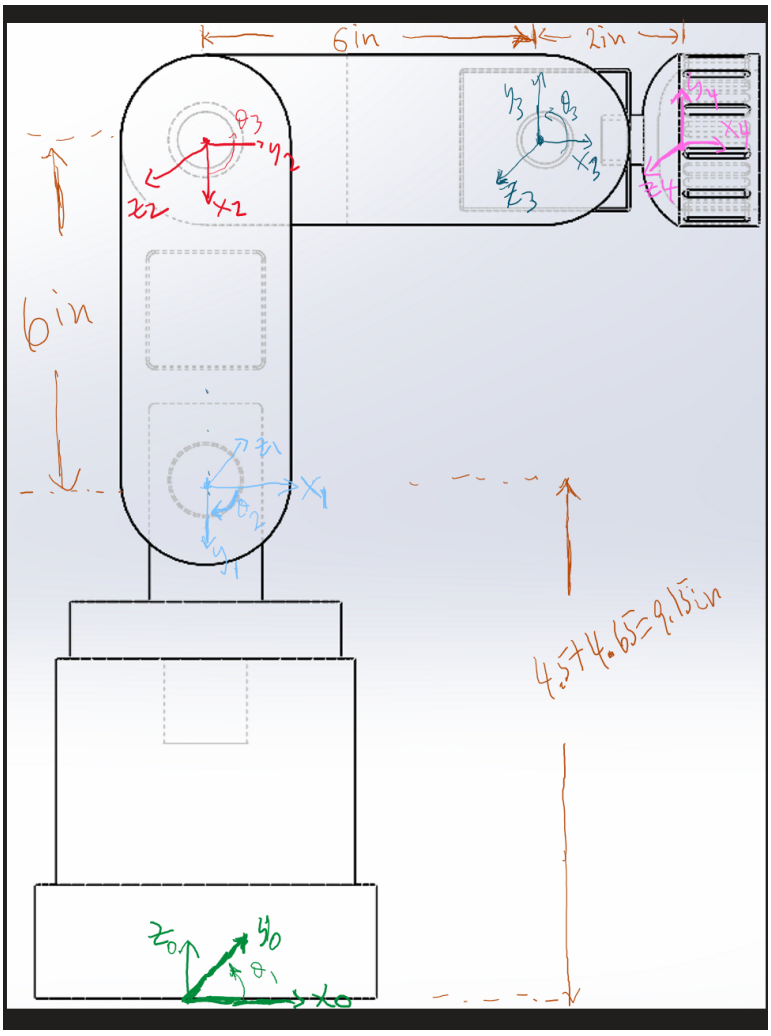
DH PARAMETERS

DH PARAMETERS - DH FRAMES

The DH frames were assigned according to Spong's method. Note that the initial DH frames and DH tables, that were submitted in the PowerPoint Presentation, were made using different DH frames than the ones below. Both methods are valid and passed the forward and inverse kinematics validation tests; the only reason for the change is that the method below was found to be more intuitive for controlling the robot arm in the Gazebo world. Also note that the joint angle limits listed on the bottom right corner of the image were found during the workspace study in Gazebo.



The image below shows the same DH frames on the solidworks robot arm model



DH- PARAMETERS - DH TABLE

The information from the DH frames was used to create the DH table below.

Theta [radians]	Alpha [radians]	a [in]	d [in]		a[m]	d[m]
theta1	-pi/2	0	9.15		0	0.23241
theat2+pi/2	pi	-6	0		-0.1524	0
theta3+pi/2	0	6	0		0.1524	0
theta4	0	2	0		0.0508	0

FORWARD POSITION KINEMATICS

The equation below from the textbook by Spong [7] was used in a python script to determine the end-effectors homogeneous transformation matrix with respect to the base frame. The screenshot of the matrix from the terminal is listed after the equation

$$A_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

the homogeneous transform for the end-effector w.r.t the base frame is:

$$
\begin{bmatrix}
(\sin(\theta_2)\cdot\sin(\theta_3)\cdot\cos(\theta_1) + \cos(\theta_1)\cdot\cos(\theta_2)\cdot\cos(\theta_3))\cdot\cos(\theta_4) + (\sin(\theta_2)\cdot\cos(\theta_1 & (\sin(\theta_1)\cdot\sin(\theta_2)\cdot\sin(\theta_3) + \sin(\theta_1)\cdot\cos(\theta_2)\cdot\cos(\theta_3))\cdot\cos(\theta_4) + (\sin(\theta_1)\cdot\sin(\theta_2 & (\sin(\theta_2)\cdot\sin(\theta_3) + \cos(\theta_2)\cdot\cos(\theta_3))\cdot\sin(\theta_4) + (-\sin(\theta_2)\cdot\cos(\theta_3 & 0 \\
)\cdot\cos(\theta_3) - \sin(\theta_3)\cdot\cos(\theta_1)\cdot\cos(\theta_2))\cdot\sin(\theta_4) & -(\sin(\theta_2)\cdot\sin(\theta_3)\cdot\cos(\theta_1) + \cos(\\
)\cdot\cos(\theta_3) - \sin(\theta_1)\cdot\sin(\theta_3)\cdot\cos(\theta_2))\cdot\sin(\theta_4) & -(\sin(\theta_1)\cdot\sin(\theta_2)\cdot\sin(\theta_3) + \sin(\\
) + \sin(\theta_3)\cdot\cos(\theta_2))\cdot\cos(\theta_4) & (\sin(\theta_2)\cdot\sin(\theta_3)
\end{bmatrix}
$$

$$
\begin{bmatrix}
\theta_1)\cdot\cos(\theta_2)\cdot\cos(\theta_3))\cdot\sin(\theta_4) + (\sin(\theta_2)\cdot\cos(\theta_1)\cdot\cos(\theta_3) - \sin(\theta_3)\cdot\cos(\theta_1)\cdot\cos(& \theta_2))\cdot\cos(\theta_4) & \sin(\theta_1) & 2\cdot(\sin(\theta_2)\cdot\sin(\theta_3)\cdot\cos(\theta_1) + \cos(\theta_1)\cdot\cos(\theta_2)\cdot\cos(\theta_3))\cdot \\
\theta_1)\cdot\cos(\theta_2)\cdot\cos(\theta_3))\cdot\sin(\theta_4) + (\sin(\theta_1)\cdot\sin(\theta_2)\cdot\cos(\theta_3) - \sin(\theta_1)\cdot\sin(\theta_3)\cdot\cos(& \theta_2))\cdot\cos(\theta_4) & -\cos(\theta_1) & 2\cdot(\sin(\theta_1)\cdot\sin(\theta_2)\cdot\sin(\theta_3) + \sin(\theta_1)\cdot\cos(\theta_2)\cdot\cos(\theta_3))\cdot \\
+ \cos(\theta_2)\cdot\cos(\theta_3))\cdot\cos(\theta_4) - (-\sin(\theta_2)\cdot\cos(\theta_3) + \sin(\theta_3)\cdot\cos(\theta_2))\cdot\sin(\theta_4) & 0 & & 2\cdot(\sin(\theta_2)\cdot\sin(\theta_3) + \cos(\theta_2)\cdot c \\
0 & 0 & & \\
\cos(\theta_4) + 2\cdot(\sin(\theta_2)\cdot\cos(\theta_1)\cdot\cos(\theta_3) - \sin(\theta_3)\cdot\cos(\theta_1)\cdot\cos(\theta_2))\cdot\sin(\theta_4) + 6\cdot\sin & & & \\
\cos(\theta_4) + 2\cdot(\sin(\theta_1)\cdot\sin(\theta_2)\cdot\cos(\theta_3) - \sin(\theta_1)\cdot\sin(\theta_3)\cdot\cos(\theta_2))\cdot\sin(\theta_4) + 6\cdot\sin & & & \\
\cos(\theta_3))\cdot\sin(\theta_4) + 2\cdot(-\sin(\theta_2)\cdot\cos(\theta_3) + \sin(\theta_3)\cdot\cos(\theta_2))\cdot\cos(\theta_4) - 6\cdot\sin(\theta_2)\cdot c & & & \\
1 & & &
\end{bmatrix}
$$

$$
\begin{bmatrix}
n(\theta_2)\cdot\sin(\theta_3)\cdot\cos(\theta_1) + 6\cdot\sin(\theta_2)\cdot\cos(\theta_1) + 6\cdot\cos(\theta_1)\cdot\cos(\theta_2)\cdot\cos(\theta_3) \\
n(\theta_1)\cdot\sin(\theta_2)\cdot\sin(\theta_3) + 6\cdot\sin(\theta_1)\cdot\sin(\theta_2) + 6\cdot\sin(\theta_1)\cdot\cos(\theta_2)\cdot\cos(\theta_3) \\
\cos(\theta_3) + 6\cdot\sin(\theta_3)\cdot\cos(\theta_2) + 6\cdot\cos(\theta_2) + 9.15
\end{bmatrix}
$$

(note “a” & “d” dimensions in inches from the DH table were used to compute the above matrix)

FORWARD POSITION KINEMATICS VALIDATION

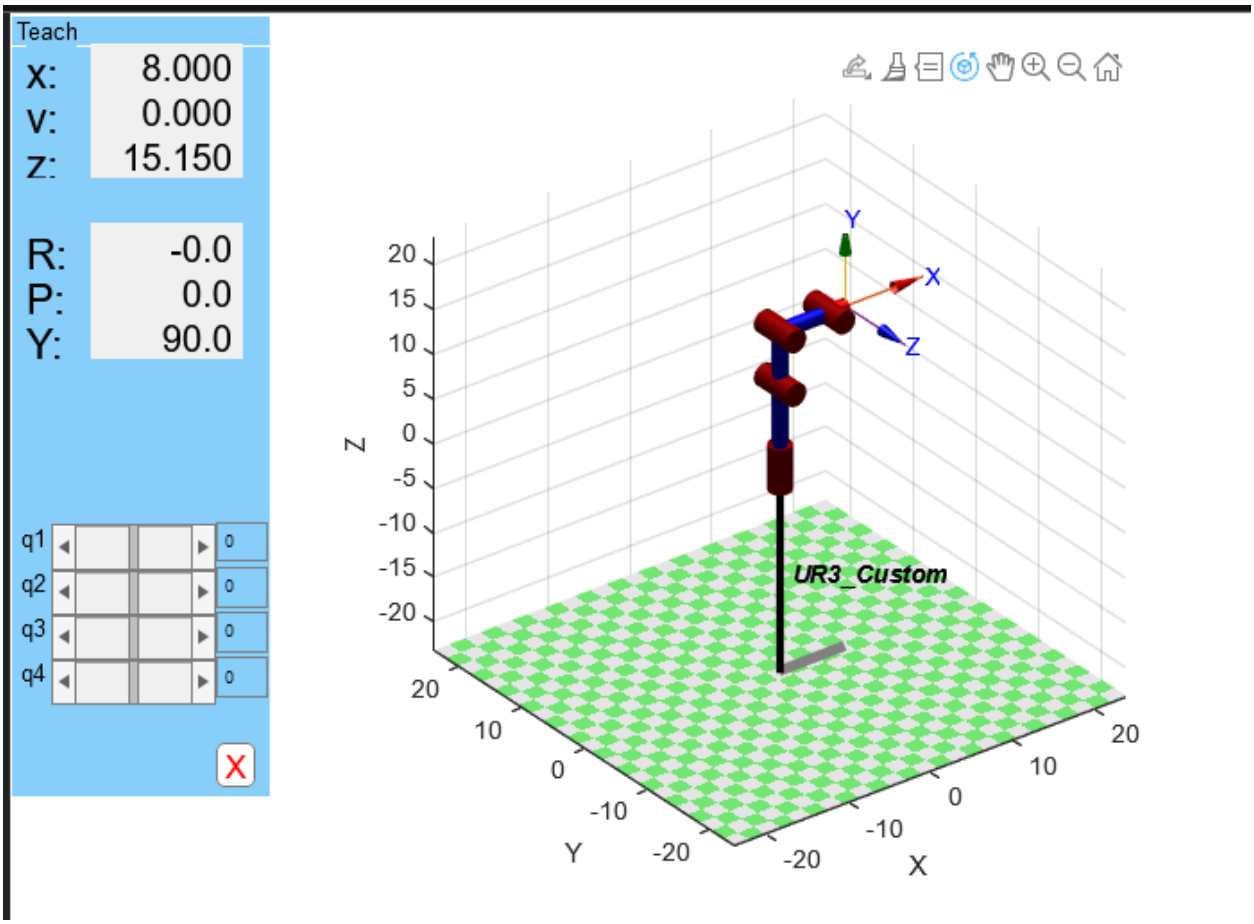
To validate the end-effector transformation matrix, each joint was rotated by 90 degree successively, and each time the end-effectors position was determined using a python script, then compared with the same measurement made using Solidworks and the Peter Corke Matlab tool.

First Position

Input (same as the home position containing the DH frames):
theta1=theta2=theta3=theta4=0.

End-effector position: (8, 0, 15.15) inches or (0.2032, 0, 0.3848) meters
The screenshots below of the results from the terminal from running the script, and the screenshot from using the matlab tool, match.

```
Enter 4 joint angles in degrees, separated by commas (no commas after last angle): 0, 0, 0, 0
[0.0, 0.0, 0.0, 0.0]

end-effector transformation matrix computed with the chosen joint angles is:

 [[ 1.    0.    0.    8.  ]
 [ 0.    0.   -1.    0.  ]
 [ 0.    1.    0.   15.15]
 [ 0.    0.    0.    1.  ]]

the (x, y, z) coordinate of the end-effector is: (8.0, 0.0, 15.15) inches

the (x, y, z) coordinate of the end-effector is: (0.2032, 0.0, 0.3848) meters
```



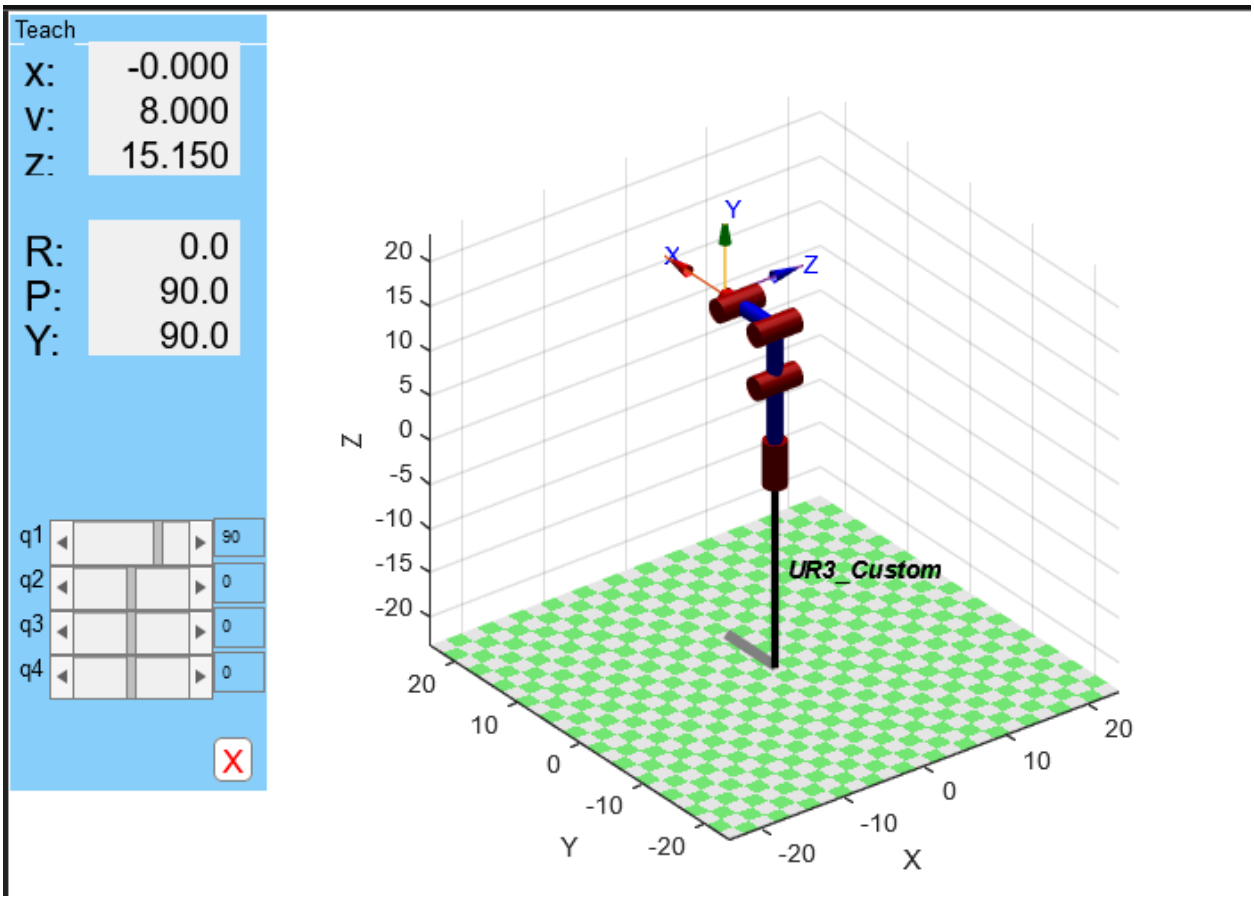
Second Position

Input: Theta1=90, theta2=theta3=theta4=0

End-effector position: (0, 8, 15.15) inches or (0, 0.2032, 0.3848) meters.

The results from the python code and the matlab tool after it, match.

```
Enter 4 joint angles in degrees, separated by commas (no commas after last angle): 90, 0, 0, 0
[90.0, 0.0, 0.0, 0.0]

end-effector transformation matrix computed with the chosen joint angles is:

 [[ 0.    0.    1.    0.  ]
 [ 1.    0.    0.    8.  ]
 [ 0.    1.    0.   15.15]
 [ 0.    0.    0.    1.  ]]

the (x, y, z) coordinate of the end-effector is: (0.0, 8.0, 15.15) inches

the (x, y, z) coordinate of the end-effector is: (0.0, 0.2032, 0.3848) meters
```

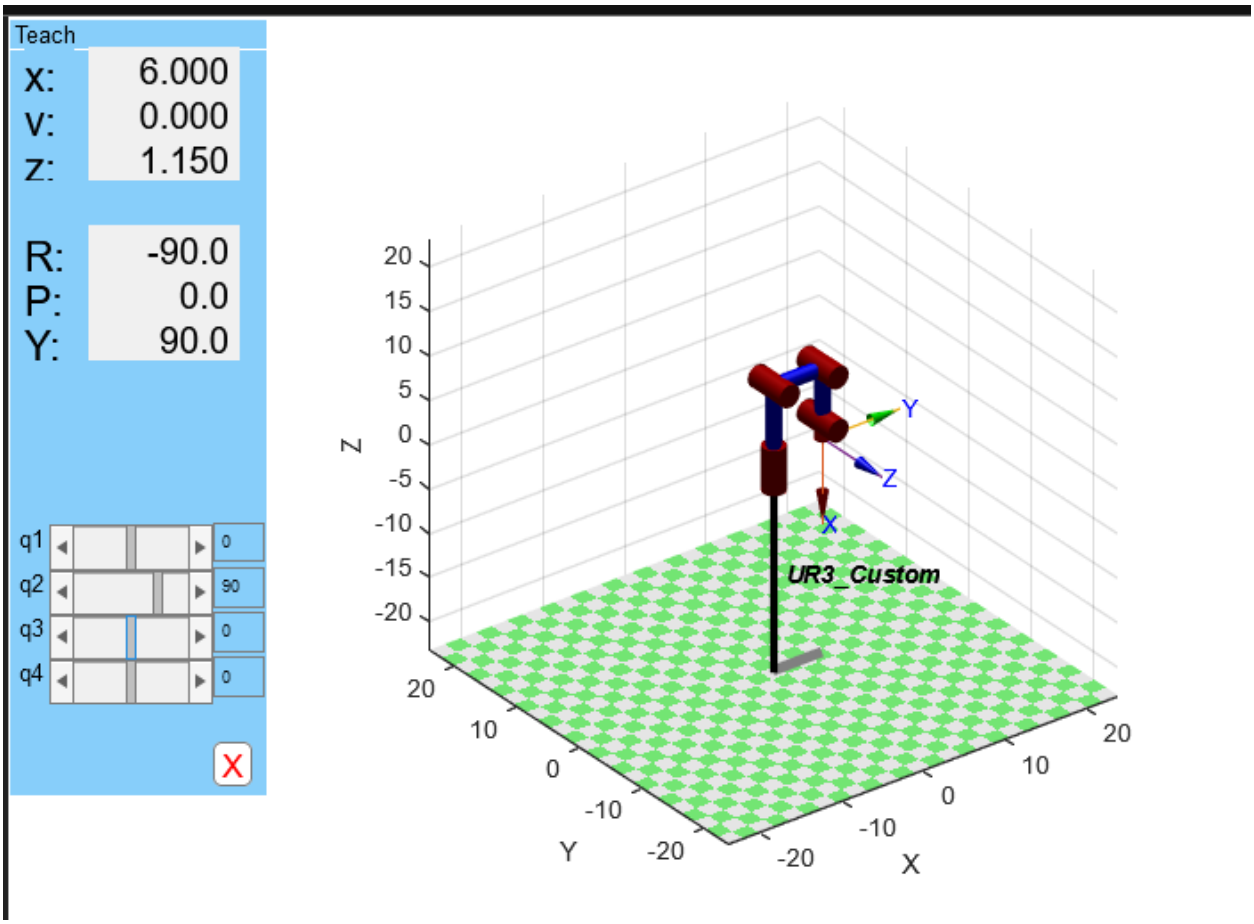


Third Position

Input: Theta1 = theta3 = theta4 = 0, theta2= 90

End-effector position: (6, 0, 1.15) inches or (0.1524, 0, 0.0292) meters
The screenshots below show the result from the python script and matlab tool, match.

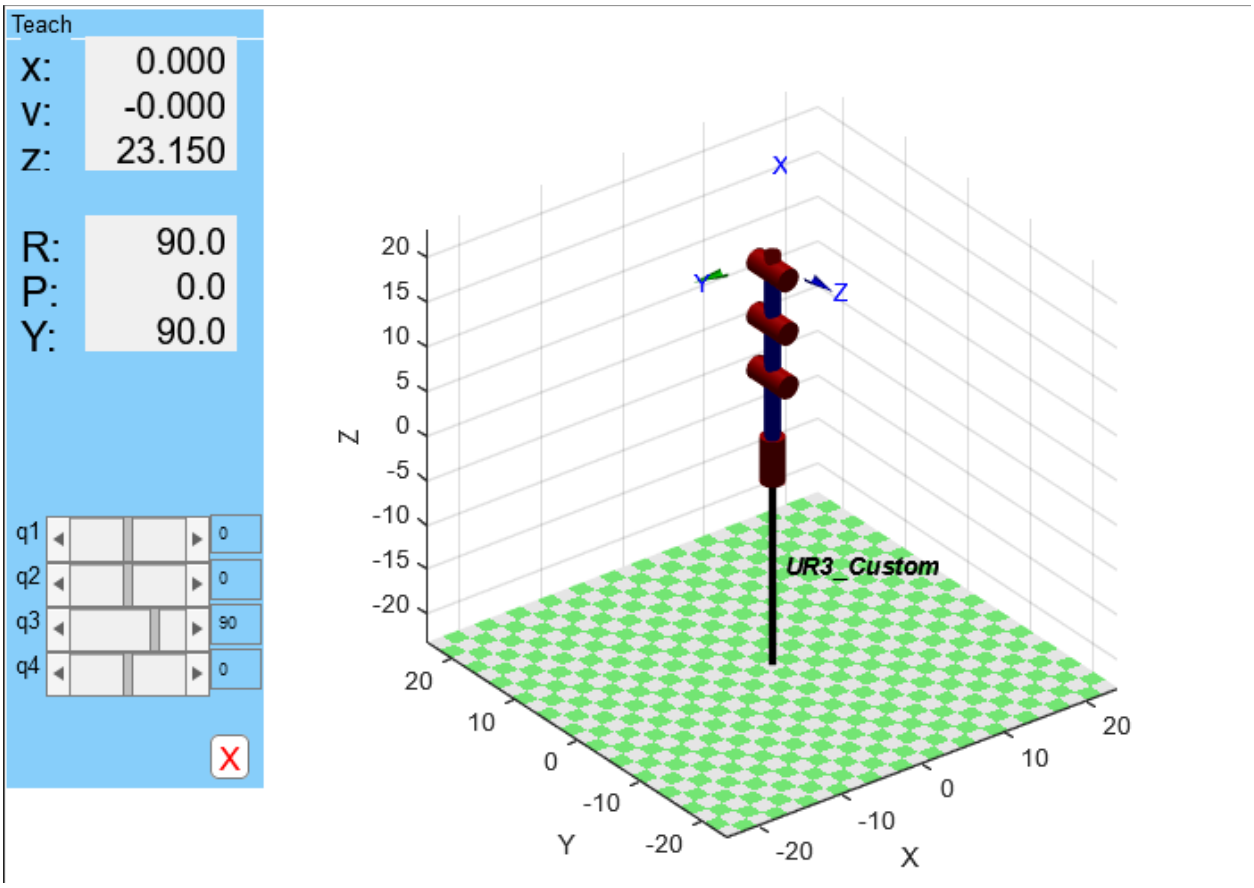
```
Enter 4 joint angles in degrees, separated by commas (no commas after last angle): 0, 90, 0, 0
[0.0, 90.0, 0.0, 0.0]

end-effector transformation matrix computed with the chosen joint angles is:

 [[ 0.    1.    0.    6.  ]
 [ 0.    0.   -1.    0.  ]
 [-1.    0.    0.    1.15]
 [ 0.    0.    0.    1.  ]]

the (x, y, z) coordinate of the end-effector is: (6.0, 0.0, 1.15) inches

the (x, y, z) coordinate of the end-effector is: (0.1524, 0.0, 0.0292) meters
```



Fourth Position

Input: Theta3 = 90, theta1 = theta2 = theta4 =0
End-effector Position: (0, 0, 23.15) inches or (0, 0, 0.588) meters
The screenshots below show the result from the python script and matlab tool, match.

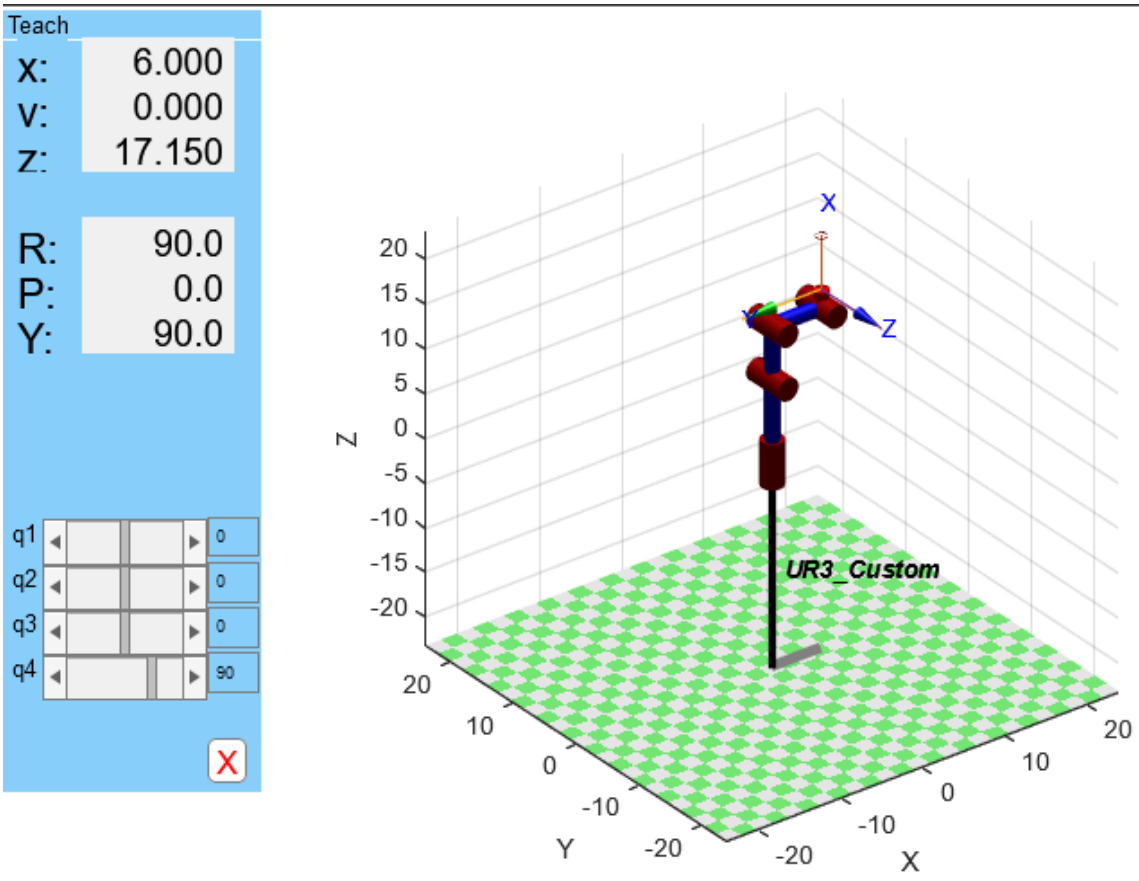
```
Enter 4 joint angles in degrees, separated by commas (no commas after last angle): 0, 0, 90, 0
[0.0, 0.0, 90.0, 0.0]

end-effector transformation matrix computed with the chosen joint angles is:

 [[ 0.   -1.    0.    0.  ]
 [ 0.    0.   -1.    0.  ]
 [ 1.    0.    0.   23.15]
 [ 0.    0.    0.    1.  ]]

the (x, y, z) coordinate of the end-effector is: (0.0, 0.0, 23.15) inches

the (x, y, z) coordinate of the end-effector is: (0.0, 0.0, 0.588) meters
```



Fifth Position

Input: Theta4 = 90, theta1 = theta2 = theta 3 = 0
End-effector position: (6, 0, 17.15) inches or (0.1524, 0, 0.4356) meters
The results below from the python script and the matlab tool match.

```
Enter 4 joint angles in degrees, separated by commas (no commas after last angle): 0, 0, 0, 90
[0.0, 0.0, 0.0, 90.0]

end-effector transformation matrix computed with the chosen joint angles is:

 [[ 0.   -1.    0.    6.  ]
 [ 0.    0.   -1.    0.  ]
 [ 1.    0.    0.   17.15]
 [ 0.    0.    0.    1.  ]]

the (x, y, z) coordinate of the end-effector is: (6.0, 0.0, 17.15) inches

the (x, y, z) coordinate of the end-effector is: (0.1524, 0.0, 0.4356) meters
```



INVERSE KINEMATICS & VALIDATION

The Jacobian relationship ξ = Jq' specifies the end-effector velocity that will result when the joints move with velocity q', where ξ = [x', y', z', wx, wy, wz] in which x', y', z' are the linear velocity of end-effector and wx, wy, wz are the angular velocity of the end-effector with respect to base frame. The inverse velocity problem is the problem of finding the joint velocities q' that produce the desired end-effector velocity. It is perhaps a bit surprising that the inverse velocity relationship is conceptually simpler than inverse position. When the Jacobian is square (i.e., J ∈ Rn×n) and nonsingular, this problem can be solved by simply inverting the Jacobian matrix to give q' = J−1ξ. For manipulators that do not have exactly six links, the Jacobian can not be inverted.

For the case when J ∈ Rm×n, such that m < n, we can solve for q' using the right pseudoinverse of J. To construct this pseudo-inverse, we use the following result from

linear algebra.

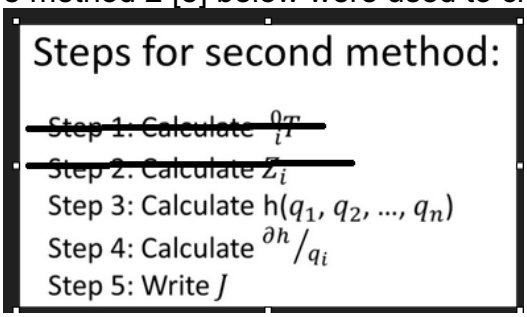
Proposition: For $J \in \mathbb{R}^{m \times n}$, if $m < n$ and rank $J = m$, then $(JJ^T)^{-1}$ exists.

In this case $(JJ^T) \in \mathbb{R}^{m \times m}$, and has rank m. Using this result, we can regroup terms to obtain

$$\begin{aligned}(JJ^T)(JJ^T)^{-1} &= I \\ J\left[J^T(JJ^T)^{-1}\right] &= I \\ JJ^+ &= I\end{aligned}$$

Here, $J^+ = J^T(JJ^T)^{-1}$ is called a right pseudoinverse of J, since $JJ^+ = I$.

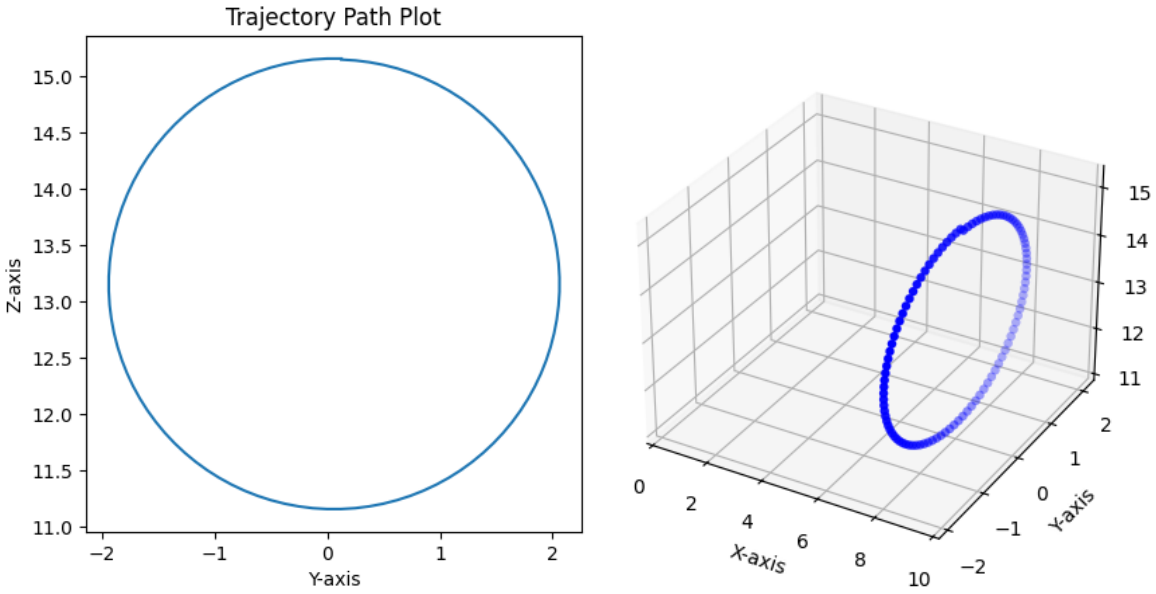
In our case, since the robot is a 3D manipulator with 4 revolute joints, the geometric Jacobian will be a 6×4 matrix, where the first 3 rows represent the end-effector's linear velocity components and the last 3 rows represent its angular velocity components in the x, y, z directions; the 4 columns correspond to the contribution of each joint velocity. This matrix is not square and therefore not directly invertible. Since this project focuses only on straight-line motions of the end-effector, we can use just the linear velocity portion (top 3 rows) of the Jacobian matrix, denoted as Jv. This results in a 3×4 Jv matrix, which simplifies the velocity kinematics calculations while still achieving the desired end-effector motion control. As a result, only steps 3 to step 5 from the Lecture 8 method 2 [8] below were used to create Jv.



The screenshot below shows the printed-out Linear Velocity Jacobian matrix (Jv) from the terminal:

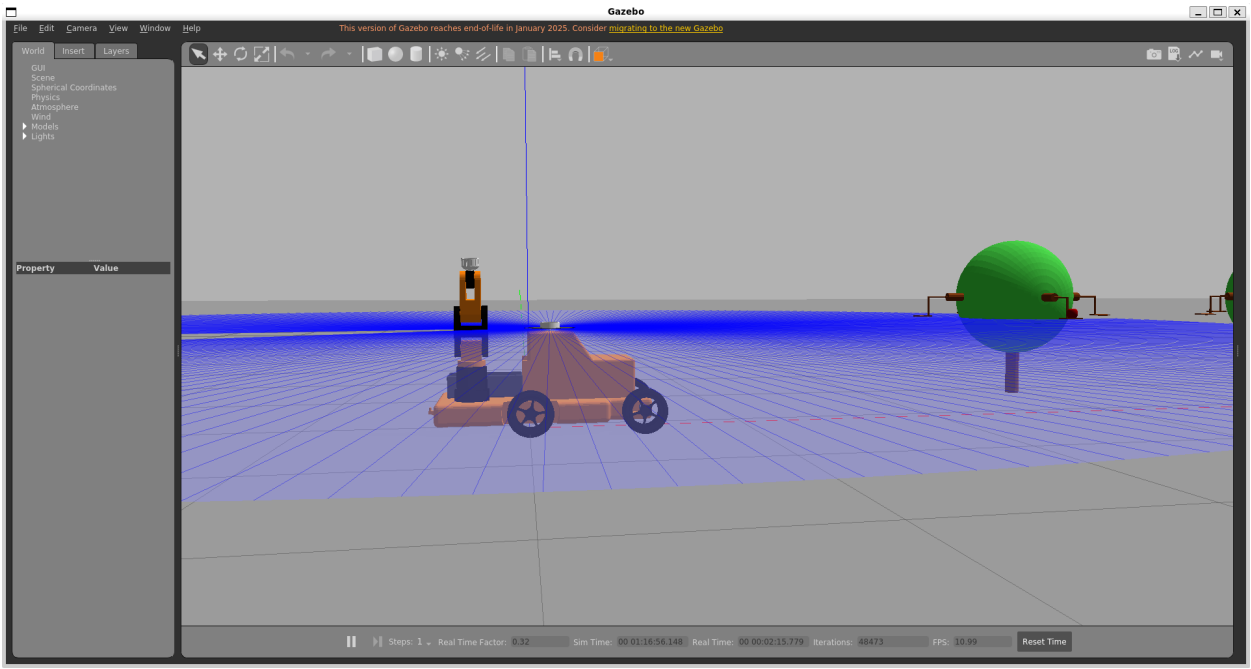
```
Linear Velocity Jacobian Matrix (Jv) for given Robot Arm:
⎡2·(-sin(θ₁)·sin(θ₂)·sin(θ₃) - sin(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) + 2·(-sin(θ₁)·sin(θ₂)·cos(θ₃) + si ↪
⎢                                                                                              ↪
⎢ 2·(sin(θ₂)·sin(θ₃)·cos(θ₁) + cos(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) + 2·(sin(θ₂)·cos(θ₁)·cos(θ₃) - sin ↪
⎢                                                                                              ↪
⎣                                                                                              ↪

↪ n(θ₁)·sin(θ₃)·cos(θ₂))·sin(θ₄) - 6·sin(θ₁)·sin(θ₂)·sin(θ₃) - 6·sin(θ₁)·sin(θ₂) - 6·sin(θ₁)·cos(θ ↪
↪                                                                                              ↪
↪ (θ₃)·cos(θ₁)·cos(θ₂))·sin(θ₄) + 6·sin(θ₂)·sin(θ₃)·cos(θ₁) + 6·sin(θ₂)·cos(θ₁) + 6·cos(θ₁)·cos(θ₂ ↪
↪                                                                                              ↪
↪     0                                                                                        ↪

↪ ₂)·cos(θ₃)  2·(sin(θ₂)·sin(θ₃)·cos(θ₁) + cos(θ₁)·cos(θ₂)·cos(θ₃))·sin(θ₄) + 2·(-sin(θ₂)·cos(θ₁)· ↪
↪                                                                                              ↪
↪ )·cos(θ₃)   2·(sin(θ₁)·sin(θ₂)·sin(θ₃) + sin(θ₁)·cos(θ₂)·cos(θ₃))·sin(θ₄) + 2·(-sin(θ₁)·sin(θ₂)· ↪
↪                                                                                              ↪
↪                                      2·(-sin(θ₂)·sin(θ₃) - cos(θ₂)·cos(θ₃))·cos(θ₄) + 2·(-sin( ↪

↪ cos(θ₃) + sin(θ₃)·cos(θ₁)·cos(θ₂))·cos(θ₄) - 6·sin(θ₂)·cos(θ₁)·cos(θ₃) + 6·sin(θ₃)·cos(θ₁)·cos(θ ↪
↪                                                                                              ↪
↪ cos(θ₃) + sin(θ₁)·sin(θ₃)·cos(θ₂))·cos(θ₄) - 6·sin(θ₁)·sin(θ₂)·cos(θ₃) + 6·sin(θ₁)·sin(θ₃)·cos(θ ↪
↪                                                                                              ↪
↪ θ₂)·cos(θ₃) + sin(θ₃)·cos(θ₂))·sin(θ₄) - 6·sin(θ₂)·sin(θ₃) - 6·sin(θ₂) - 6·cos(θ₂)·cos(θ₃)       ↪

↪ ₂) + 6·cos(θ₁)·cos(θ₂)  2·(-sin(θ₂)·sin(θ₃)·cos(θ₁) - cos(θ₁)·cos(θ₂)·cos(θ₃))·sin(θ₄) + 2·(sin( ↪
↪                                                                                              ↪
↪ ₂) + 6·sin(θ₁)·cos(θ₂)  2·(-sin(θ₁)·sin(θ₂)·sin(θ₃) - sin(θ₁)·cos(θ₂)·cos(θ₃))·sin(θ₄) + 2·(sin( ↪
↪                                                                                              ↪
↪                                           2·(sin(θ₂)·sin(θ₃) + cos(θ₂)·cos(θ₃))·cos(θ₄) +  ↪

↪ θ₂)·cos(θ₁)·cos(θ₃) - sin(θ₃)·cos(θ₁)·cos(θ₂))·cos(θ₄) + 6·sin(θ₂)·cos(θ₁)·cos(θ₃) - 6·sin(θ₃)·c ↪
↪                                                                                              ↪
↪ θ₁)·sin(θ₂)·cos(θ₃) - sin(θ₁)·sin(θ₃)·cos(θ₂))·cos(θ₄) + 6·sin(θ₁)·sin(θ₂)·cos(θ₃) - 6·sin(θ₁)·s ↪
↪                                                                                              ↪
↪ 2·(sin(θ₂)·cos(θ₃) - sin(θ₃)·cos(θ₂))·sin(θ₄) + 6·sin(θ₂)·sin(θ₃) + 6·cos(θ₂)·cos(θ₃)            ↪

↪ os(θ₁)·cos(θ₂)  -2·(sin(θ₂)·sin(θ₃)·cos(θ₁) + cos(θ₁)·cos(θ₂)·cos(θ₃))·sin(θ₄) + 2·(sin(θ₂)·cos( ↪
↪                                                                                              ↪
↪ in(θ₃)·cos(θ₂)  -2·(sin(θ₁)·sin(θ₂)·sin(θ₃) + sin(θ₁)·cos(θ₂)·cos(θ₃))·sin(θ₄) + 2·(sin(θ₁)·sin( ↪
↪                                                                                              ↪
↪                                  2·(sin(θ₂)·sin(θ₃) + cos(θ₂)·cos(θ₃))·cos(θ₄) - 2·(-sin(θ₂)·cos( ↪

↪ θ₁)·cos(θ₃) - sin(θ₃)·cos(θ₁)·cos(θ₂))·cos(θ₄)⎤
↪                                              ⎥
↪ θ₂)·cos(θ₃) - sin(θ₁)·sin(θ₃)·cos(θ₂))·cos(θ₄)⎥
↪                                              ⎥
↪ θ₃) + sin(θ₃)·cos(θ₂))·sin(θ₄)               ⎦
```

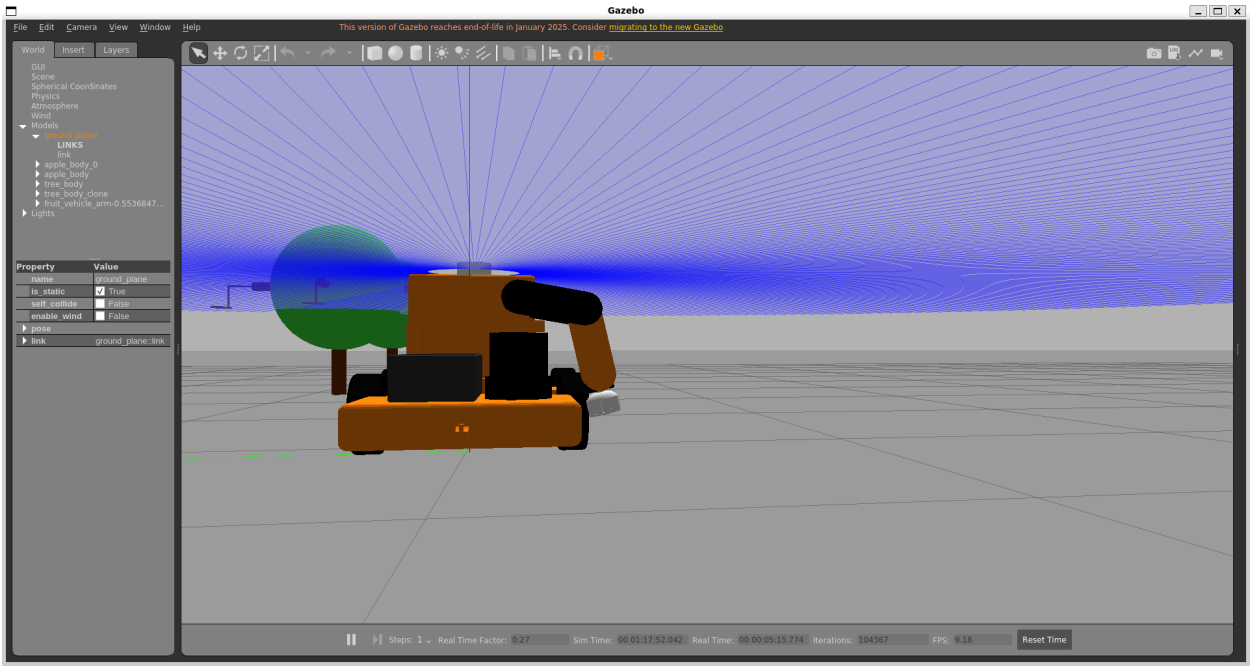
The inverse kinematics validation was done by writing a script to have the robot draw a circle of radius 2 inches. The plotted circle on the YZ - plane is shown below:



WORKSPACE STUDY

Our robot can drive forward and backwards, and we can control its orientation with our steering wheels. This gives us a very large workspace for the robot that is only limited by where the robot can drive to, and the highest and lowest heights that the robot can reach with the manipulator arm.





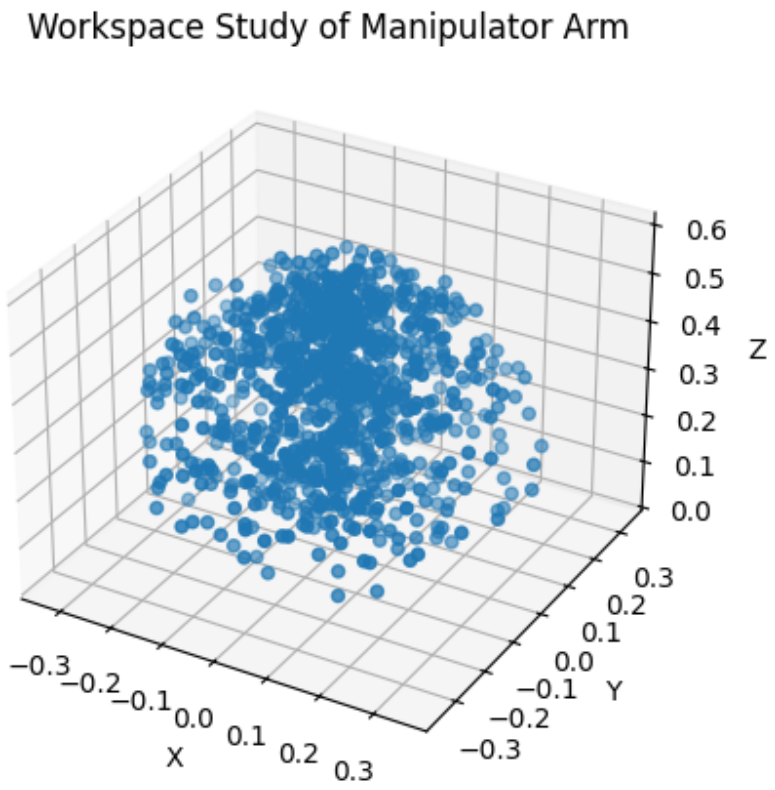
Now we will focus on the workspace of the robot arm when the truck base is fully stopped. When stopped, the robot's workspace consists of part of a hemisphere with limits to avoid contact between links, and the truck body. The joint limits of our manipulator arm were determined in Gazebo using a teleop script. We determined the following limits to be applied to each joint of the robot-arm, in radians, in the URDF.

Arm Joint	Minimum Range	Maximum Range
Joint 1	-pi/2	pi
Joint 2	-pi/2	pi/2
Joint 3	-1	4.5
Joint 4	-2.5	2.5

Here's the link to the video for the workspace study being done with teleop:

https://drive.google.com/file/d/17SFGPVSB7CHLmiUaYZs7CN_LTJuTty-l/view?usp=sharing

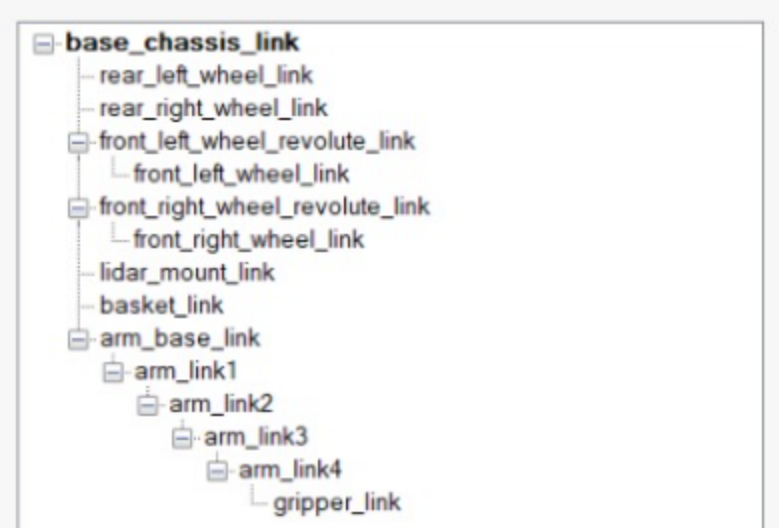
A random sample of 1000 end effector positions that could be reached based on the previous determined joint angles was taken to determine our manipulator workspace.



For our robot, if our target isn't in our manipulator's workspace, we can move the truck base of the robot until it is.

URDF

The links and joints were set-up with the parent-child relationship in the URDF Exporter tool, as shown in the screenshot below.Some parts, like the wheel rims and tires, and the truck-body and chassis, were combined into single links to help reduce the size of the generated URDF file.



Note that several adjustments were made to the initial URDF created with the URDF exporter, once that file was used in the ros package.

CONTROL METHOD

Our truck base for the fruit picker is a rear wheel drive vehicle with the rear wheel used for motion and the front wheels steering. To achieve rear wheel drive, velocity controllers were placed on the rear wheels of the robot to drive how fast the robot is going. . And position controllers were placed on the truck's front revolute steering joints to allow turning the car. The closed-loop P-controller node that was used for project 1 is re-used to command & control the truck close to a tree

Our manipulator to harvest our apples is a 4 link revolute manipulator. It will allow us to pick our apple from the tree and drop it in the collection basket in our vehicle. We place a position controller on each of our manipulator's 4 revolute joints. We used position controllers because our fruit picking robot has to reach specific points such as the location of the apple and basket. Position controllers will also allow us to control our robot with an approach similar to the one used in homework 2. The forward and inverse kinematics equations for the robot arm is used to create the code for controlling the robot. We perform a combination of simple movements, such as following a straight line or an arc, depending on the location of the apple.

GAZEBO AND RVIZ VISUALIZATION

Gazebo

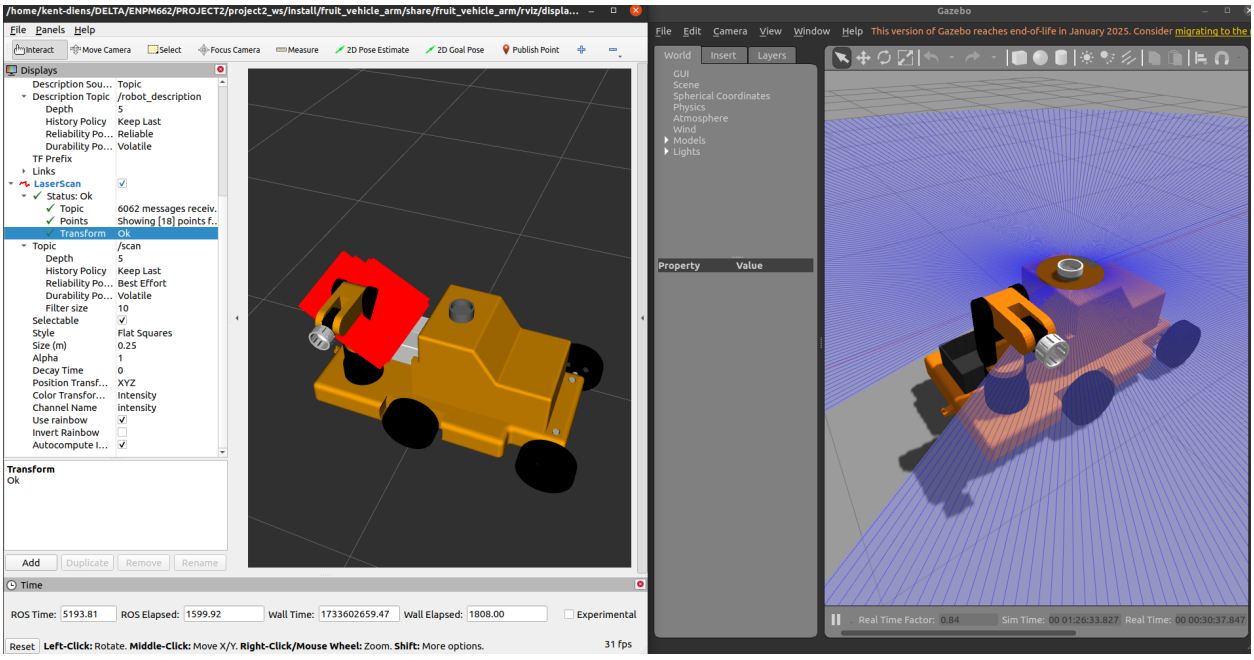
We created two worlds to demonstrate our robot's abilities in gazebo. Our first world consists of trees with apples attached to them. This allows us to demonstrate both motion and manipulation since we have to first navigate to the tree, then manipulate the robot arm to extract the apples.

Our second world had three apples on a flat surface. This allowed us to further demonstrate manipulation by having apples that we can pick and drop in our basket using limited motion.

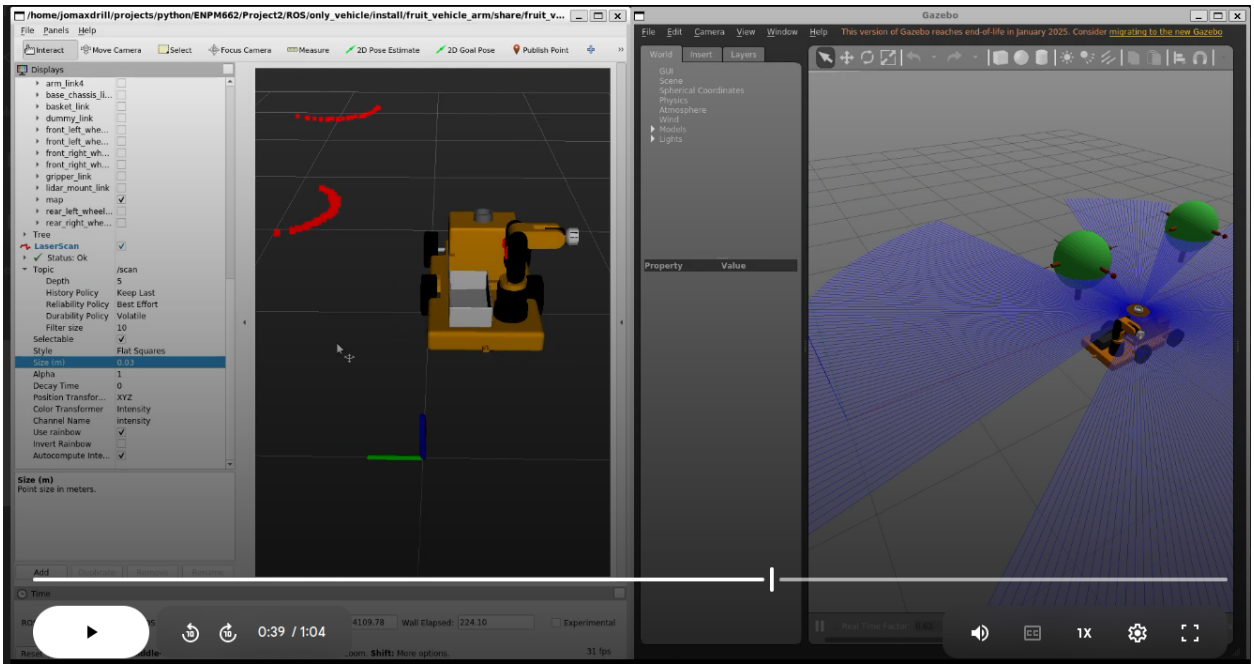
The following is a video showing the mobile robot harvesting apples from the trees: https://drive.google.com/file/d/1bWp0tc4YbuVi7mIBvuK-GSt5amngVuD6/view?usp=sharing

RVIZ

The picture below shows the lidar active in RViz (with a large size set for the red detection points) and Gazebo (in an empty world), properly detecting part of the robot arm behind it.



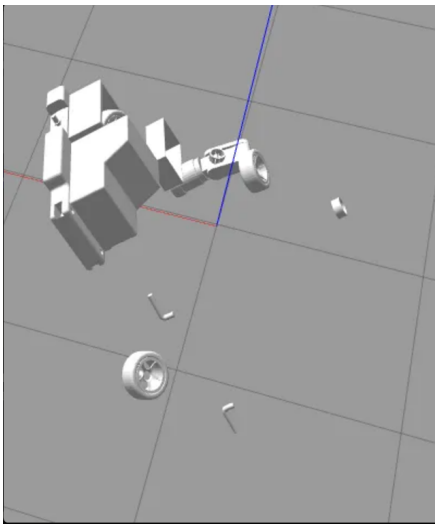
The screenshot below shows the lidar properly detecting the trees and the robot arm, in RViz (see red dots) and the Gazebo world with fruit trees



The link below shows a video of the same thing.
https://drive.google.com/file/d/1abLG8P37SsgXIo9mUsSpZqwMaZRfMZdK/view?usp=drive_link

PROBLEMS FACED

- The truck and robot arm initially would not spawn in Gazebo with the initial URDF that was made, even though best practices were followed, the URDF using the URDF Exporter tool. After extending the spawn delay, the robot appeared in Gazebo but most of the parts were not connected.



- The robot arm assembly and fruit container were reassembled with the truck in solidworks and a brand new URDF file was remade. With this new file, the robot spawn properly in Gazebo. However, several additional adjustments in the URDF had to be made manually through trial and error, to adjust the orientation of the links, and to also fix the direction of rotation of their joints, while moving the robot in RVIZ and gazebo. The link mass, joint damping, friction and effort also had to be adjusted.
- Our robot initially would not move when a velocity was applied to the wheels due to our wheels having reverse z axes. We had to apply equal and opposite velocities to get the wheels to move in a straight line.
- The vacuum gripper posed a few problems. The vacuum gripper would not grip. This was because the vacuum gripper plugin required the gripper to be attached to the manipulator with a revolute joint. After the force of gravity on the apples would be greater than the suction of the vacuum gripper. This was fixed by reducing the weight of the apples. Finally, the robot and apple colliding with each other would send the robot far away from each other. Fixing this issue required many manual adjustments in the URDF files such as decreasing the collision velocity.

LESSONS LEARNED

- The URDF Exporter tool does not always produce a good URDF without any issues, especially as the number of joints and links increases. Additional time needs to be allocated for manual adjustments to be made using Gazebo and Rviz.
- It's necessary to consider some dynamics variables such as gravity, friction and damping to improve the behavior of the simulation.
- Some plugins need more adjustments to work properly.
- ROS and its plugins are not fully documented and some tasks will require trial and error to be performed.

CONCLUSION

This project helped us learn how to apply the kinematics concepts and theories learned in this course to design a mobile robot, made of a robot arm and a truck, for the specific task of harvesting apples. It also helped us gain a better understanding of how to modify a URDF file, and work with various spawn and world files in the ros package, to properly simulate the robot's motion in Gazebo.

FUTURE WORK

Instead of giving the robot the position of the tree to go to, a camera could be added to the truck to allow it to determine the exact position of the tree and fruit that it needs to target. The gripper part of the robot arm could be made removable, so that the robot arm could be used with different gripper types, like a saw, to help cut branches for better access to the fruit, or a regular gripper with padding to pick some fruits more delicately if needed.

GITHUB LINKS

https://github.com/Jomaxdrill/fruit_vehicle_arm

KEY CONTRIBUTIONS OF EACH MEMBER

- Jonathan - Adjusted the truck and robot arm assembly in solidworks, then created the first functional URDF files and ros package, then manually adjusted it. Designed the apple trees and apple fruits, converted them to SDF format and added them into the Gazebo world. Double-checked forward position kinematics validation results with matlab tool. Kept track of progress and testing of every step of the project. Created & organized github repository.

- Robens- Built-on the initial ros package by adding controllers, vacuum gripper, and further modifying the URDF to make them work and allow the robot to move and properly pick apples. Did the workspace study in Gazebo, using the Teleop script. Combined the robot base and manipulator controller.

- Hamsa - Redid DH frames & DH Tables in a way to work better in Gazebo. Performed the inverse kinematics calculations, and inverse kinematics validation. Wrote the code to allow the robot to autonomously go to an apple and pick a fruit, then recorded a video of it.

- Kent- Designed the fruit basket, and the entire 4 DOF fruit picker robot. Made the first version of the truck and robot arm assembly and the first URDF and mesh files. Set-up the initial DH frames, DH tables (used in the presentation slides),performed the forward position validation using solidworks and a python script. Created most of the report by refining and extending the contents from the PowerPoint presentation.

REFERENCES

[1] “Apple harvesting robot plucks a piece of fruit every 7 seconds,” New Atlas, Apr. 28, 2021. https://newatlas.com/robotics/apple-harvesting- robot-fresh-seven-seconds/

[2] “Four wheel driving and steering vehicle with Suspension System”, RoboDyne. https://www.robo-dyne.com/explorer-4wd-ugv-four-wheel- driving-steering-robot-vehicle/

[3] Guangrui Hu, Chao Chen, Jun Chen, Lijuan Sun, Adilet Sugirbay, Yu Chen, Hongling Jin, Shuo Zhang, Lingxin Bu, “Simplified 4-DOF manipulator for rapid robotic apple harvesting”, Computers and Electronics in Agriculture, Volume 199, 2022.

[4] Guangbing Bao, Shizhao Liu, Hong Zhao, Kinematics Simulation of 4 - DOF

Manipulator, Advances in Engineering Research, volume 123, MSMEE, 2017.

[5] Y. Luan, W. Xu, J. Li, D. Zhou, H. Wang and H. Ji, "Kinematics Modeling and Simulation of a 4-DOF Manipulator," *2017 International Conference on Computer Systems, Electronics and Control (ICCSEC)*, Dalian, China, 2017, pp. 302-305, doi: 10.1109/ICCSEC.2017.8446769.

[6] Z. B. Hazem, R. Ince and S. Dilibal, "Joint Control Implementation of 4-DOF Robotic Arm Using Robot Operating System," *2022 International Conference on Theoretical and Applied Computer Science and Engineering (ICTASCE)*, Ankara, Turkey, 2022, pp. 72-77, doi: 10.1109/ICTACSE50438.2022.10009733.

[7] Spong, Mark W.; Hutchinson, Seth, *Robot Modeling and Control*.

[8] Monfaredi, Reza, *ENPM662-2024-Lect8-Velocity Kinematics and Jacobian-v1.pptx*