ME G511 MECHANISMS AND ROBOTICS

ASSIGNMENT No. - NA Assignment Type: PHASE-III OF PROJECT

Individual / Group (mention) - GROUP

Id. No.:	Name:
2017B5A41031G	ADITYA MAJALI (Group Leader)
2017A4PS0293G	DEVANSHU WAKHALE
2017B4A40052G	DEVESH DIMBLE
2017B5A40765G	VENUGOPALAN IYENGAR

Title of Project: APPLE FRUIT PICKER

Problem statement: During the fruit harvest season, costs incurred due to the hiring of labor can be reduced by employing a semi-autonomous robot that can detect and pluck the ripe fruits at the right time and drop them in a carton for further processing.

Brief Description: The increasing amounts of investments in the field of agriculture have generated the need to reduce the costs incurred due to the hiring of labor to work on the farmland. This need for labor is more frequent in plantations that grow fruit-bearing trees, especially during the harvest season as these generally spread across many acres. To facilitate the task of plucking ripe fruits at the right time, a robot can be used that can work round the clock. The robot will be responsible for picking the apples from the trees accurately and dropping them in the collecting carton or bag.

Methodology:

- The robot will determine the existence of a ripe apple fruit based on deep learning models using an image recognition algorithm.
- To measure the distance to the apple, we can use an ultrasonic sensor and feed the distance measurement to the robotic arm so that it can be moved accordingly.
- This bot can be trained for other fruits of similar size and tree characteristics like oranges, etc.

Details of the Project:

Type of Motion:

The robot which will be used for the task will have parameters R-P-R-R-R-R which is a six degree of freedom robot. The robot can be seen as divided into two motions, one is the movement of the robot from the home position to the position of the apple (fruit), which will use the R-P-R-R sequence of operations. The second motion (Gripper) will utilize the last R-R for pitch and roll motions.

The robot will pluck the fruit and then palletize the collected fruits into a neat tray for further transport or processing.

Motion sequence:

1. The robot starts at the **home position.**
2. The robot rotates towards the tree (by varying **ϴ**1) so that the camera faces the tree line.
3. The **camera** on the robotic arm detects Apple's **x, y location**.
4. **The gripper** is aligned with the **detected location.** This is done by calculating all the parameters of the arm, from the home position to the apple’s x, y position.
5. The depth/distance of the apple from the gripper is detected using the **ultrasonic sensor.**
6. The arm moves forward towards the **apple.**
7. The gripper grasps the apple gently, **then rotates and retracts with the apple.** This is the **plucking** motion.
8. The arm then returns to the **home position.**
9. The robot then moves towards the **pallet and places** the apple on it.
10. It **again returns to the home position**.

Workspace Sketch:

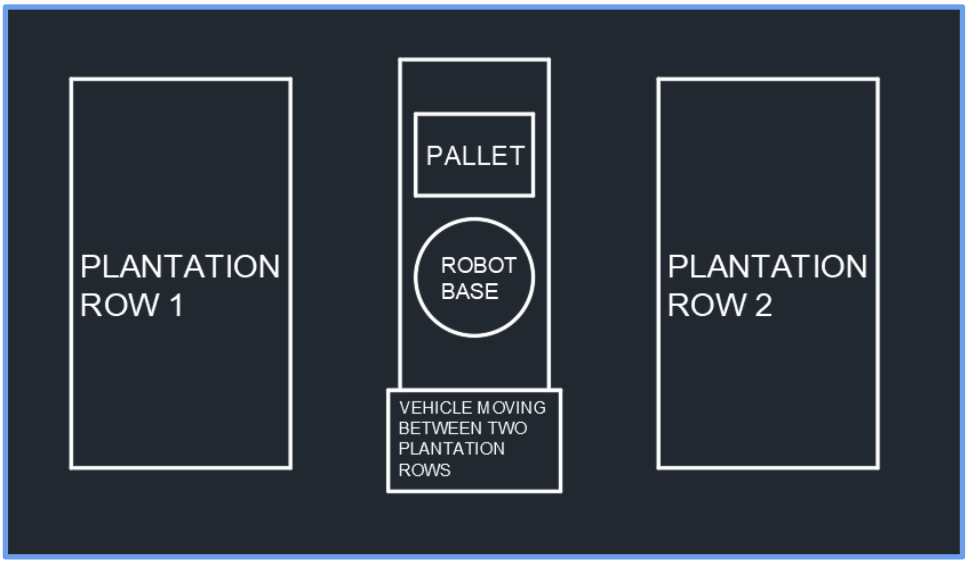


Figure 1: Top view of the layout.

Generally, trees are planted in rows in a plantation with enough space between two rows for a mini-truck / tractor to move. The robot will be mounted on the back of a mini-truck / tractor that will move it around the plantation. The robot will scan one tree at a time and will try to detect a ripe apple while the truck remains stationary in front of the tree. The detected apple will be plucked and palletized. After all the apples from one tree have been plucked, the robot rotates and starts scanning the tree in the opposite row. Hence, two trees can be covered at one location.

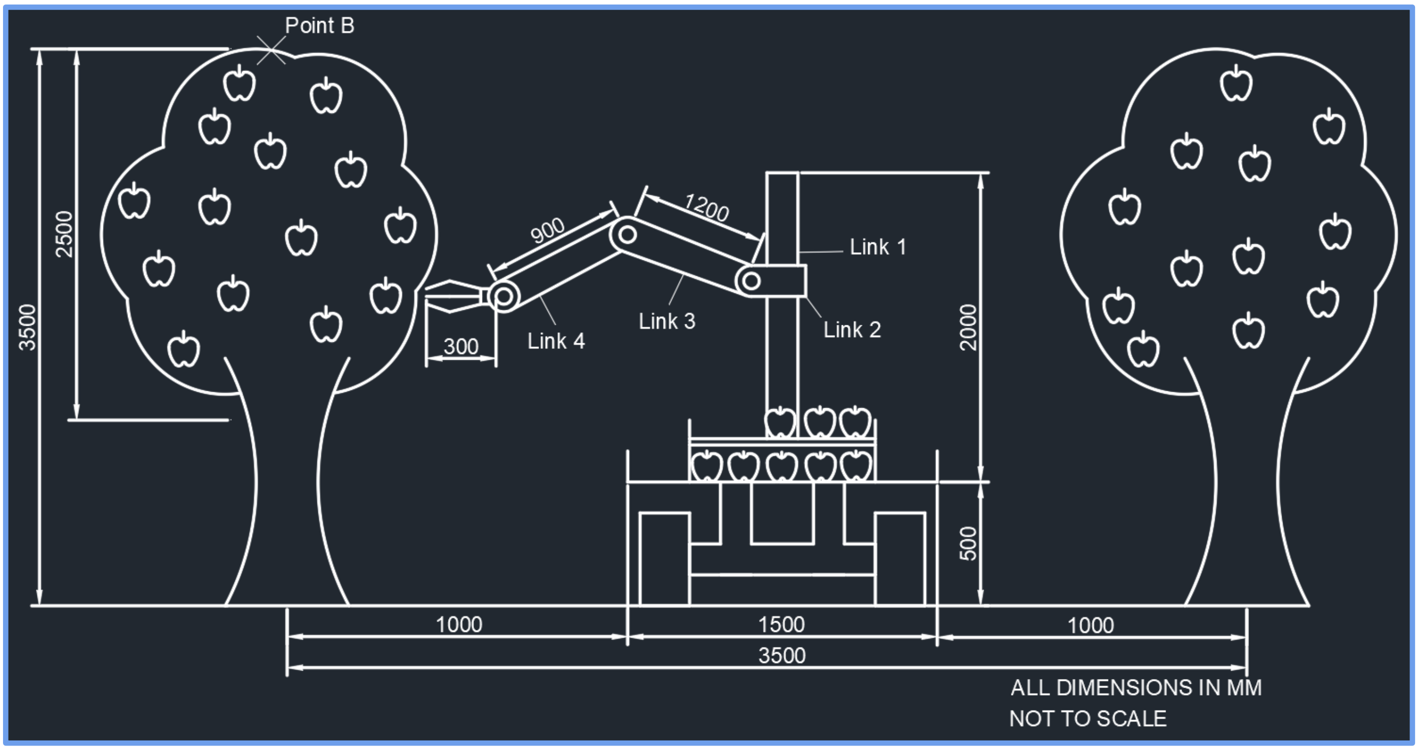


Figure 2(a): Ground view of the layout with dimensions (as seen from behind the vehicle).

Figure 2(b): Ground view of the layout with dimensions (as seen from behind the vehicle).

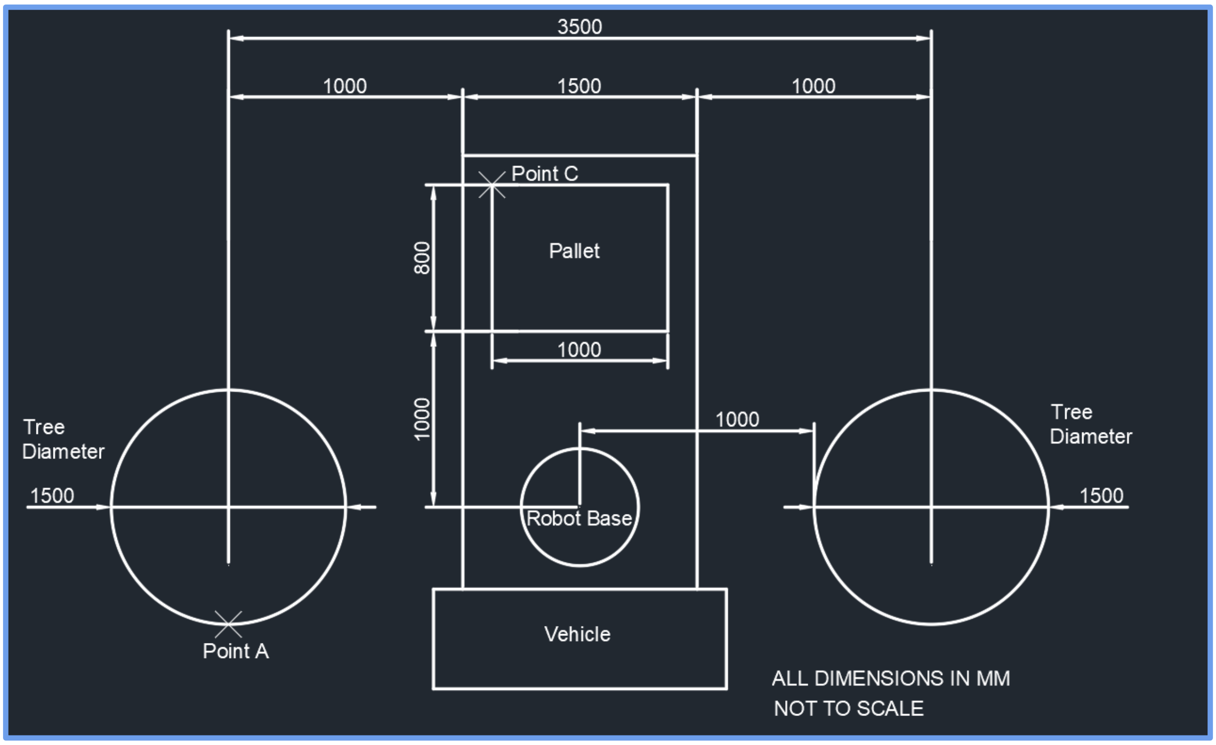


Figure 3(a): Top view of the layout with dimensions.

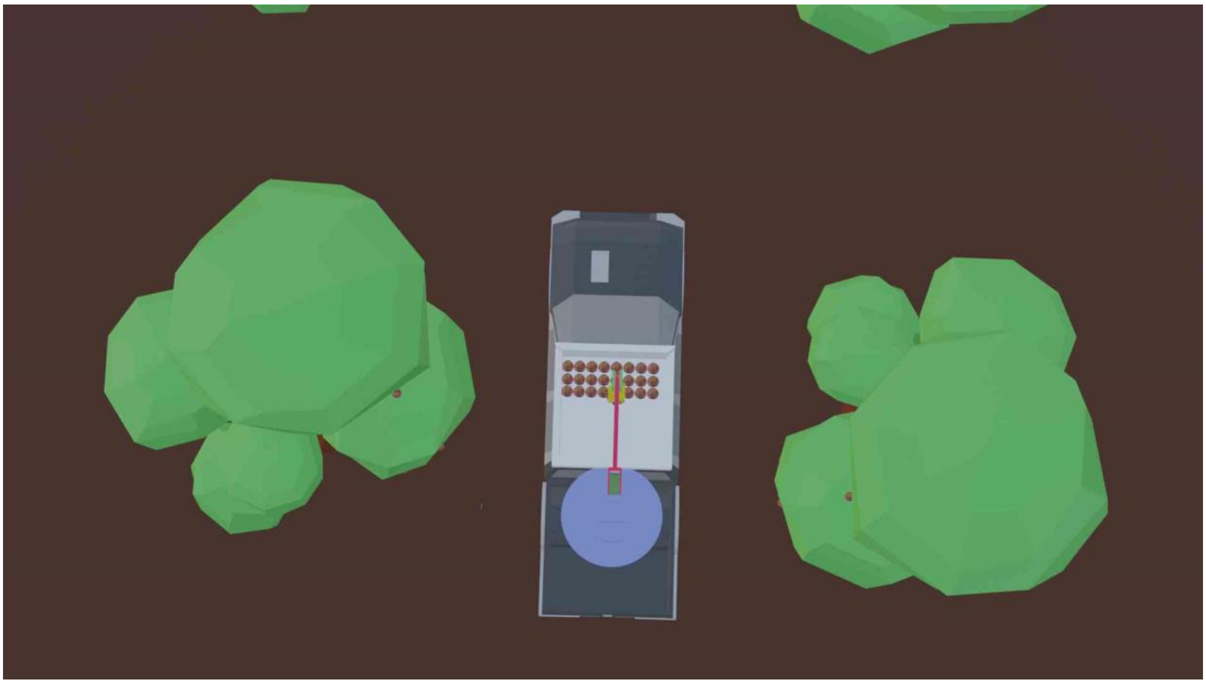


Figure 3(b): Top view of the layout.

Link length calculations:

Before calculating the link lengths, it is necessary to find the maximum and minimum extensions that the arm of the robot is expected to reach.

As it can be seen from the Figure 3 above, **the minimum distance from the robot base in the workspace is 1000 mm**.

The maximum distance, say X, is calculated by using the Pythagoras theorem, for two possible locations in the workspace:

A. For the first location, the Point A (see figure 3) on the circumference of the tree is chosen (Figure 4 below which depicts the top view of the layout).

From pythagoras theorem,

$$x = \sqrt{1750^2 + 750^2} = 1903.94mm$$

Hence the maximum distance is ≈ 1900 mm.

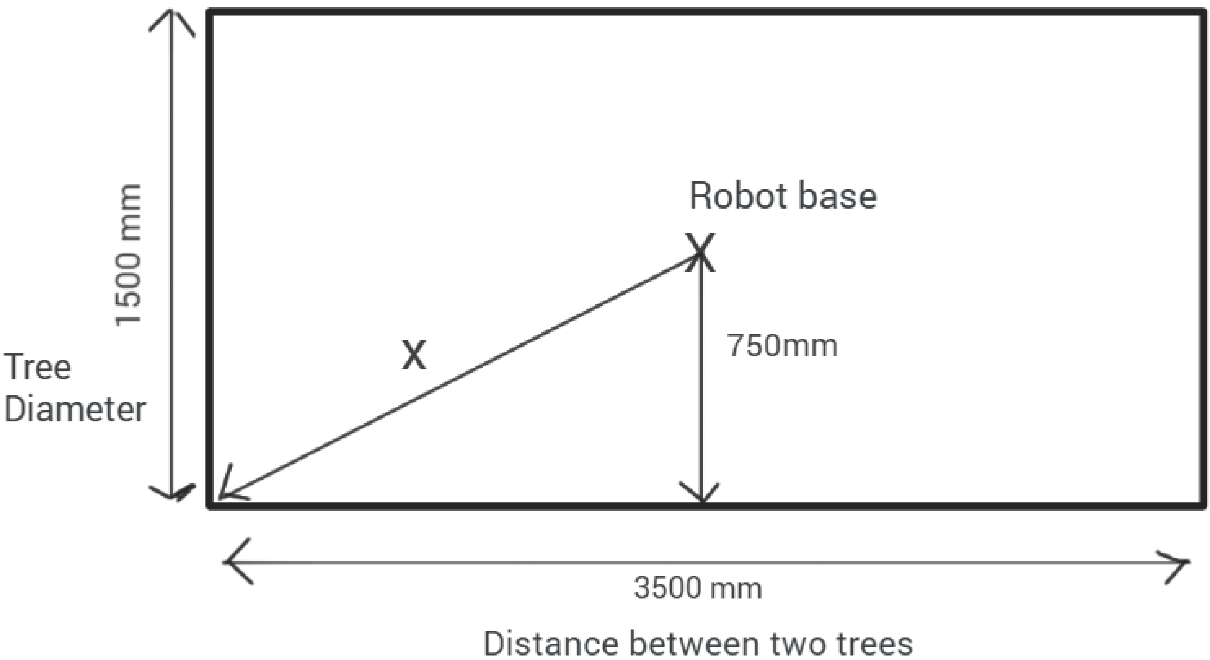


Figure 4: Calculating maximum extension of arm for Point A

B. For the second location, the Point B (see figure 2) on the top of the tree has been chosen. Firstly, to reach this position, **the Link 1 should be 2000 mm in height**, so that the arm can extend further. (Figure 5 below, which shows the front view of the layout)

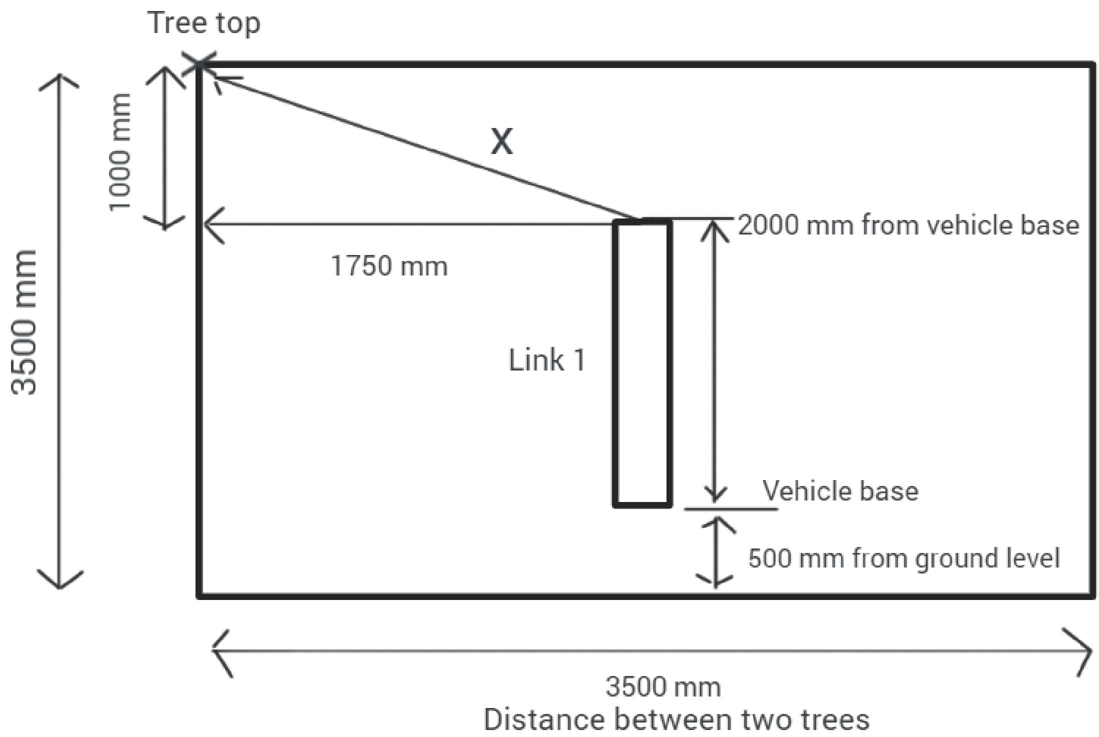


Figure 5: Calculating maximum extension of arm for Point B.

For finding the distance to the top of tree, from pythagoras theorem,

$$x = \sqrt{1750^2 + 1000^2} = 2015.56mm$$

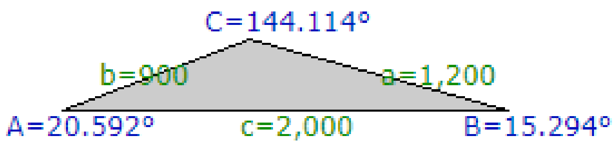
Hence, in this case, the maximum distance is ≈ 2015 mm.

To be on the safer side, we can say that the maximum extension for **the links 3 and 4 combined will be 2000 mm**.

Now that the maximum and the minimum distances are known, we can assume that the length of **Link 3 = 1200 mm**, and **Link 4 = 900 mm** (as taking the length of link 4 = 800 mm would not yield a feasible solution for the triangle thus formed).

Using properties of triangle, it can be shown that these link lengths are able to reach the maximum and the minimum distances:-

Link 3 => side a = 1200 mm Link 4 => side b = 900 mm



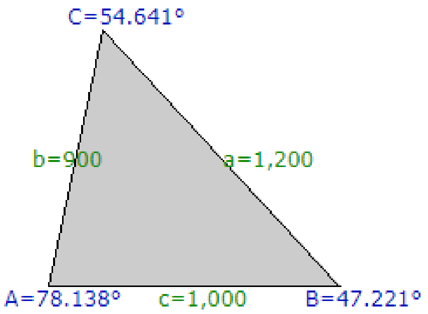


Fig. 6 : Maximum extension **Fig. 7 : Minimum extension**

Robot Sketch:

A. This is the configuration we want to achieve using the robot in **home position**:

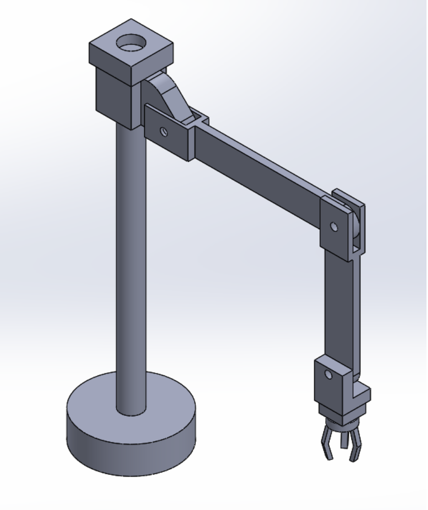
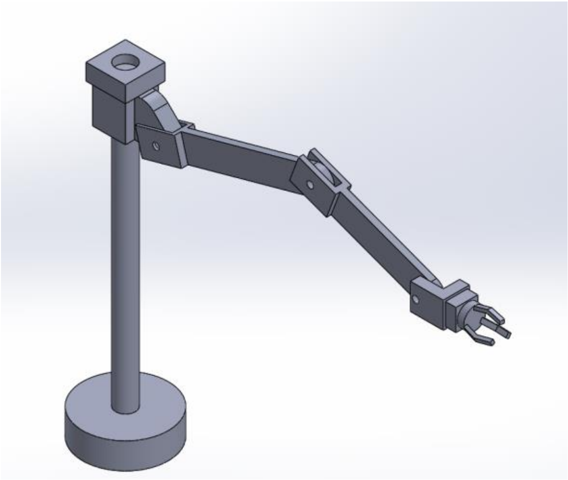


Fig.8 : Home position

B. The robot will look like this when it is plucking the apple:



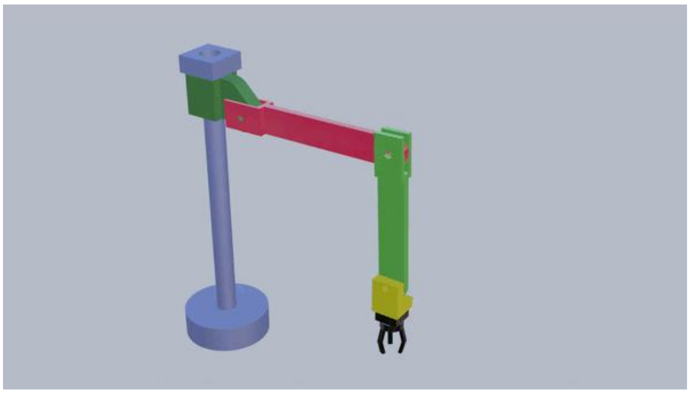


Fig. 9 (to the left) : Robot-maximum extension

Fig. 10 (top) : Robot moving to pluck apple

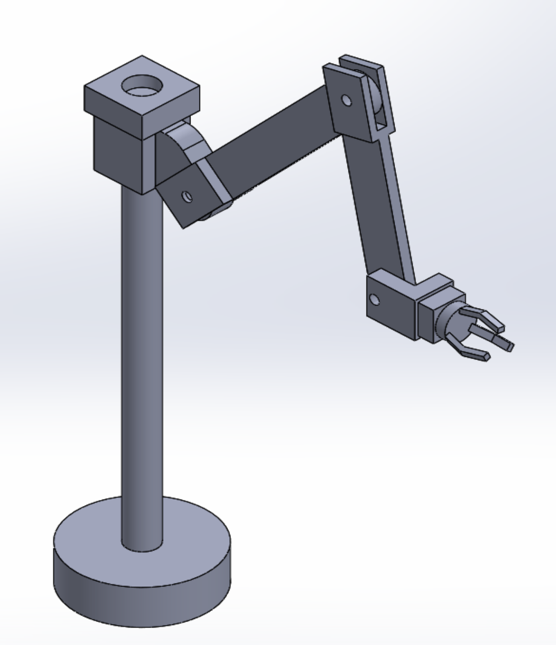


Fig. 11 : Robot-minimum extension

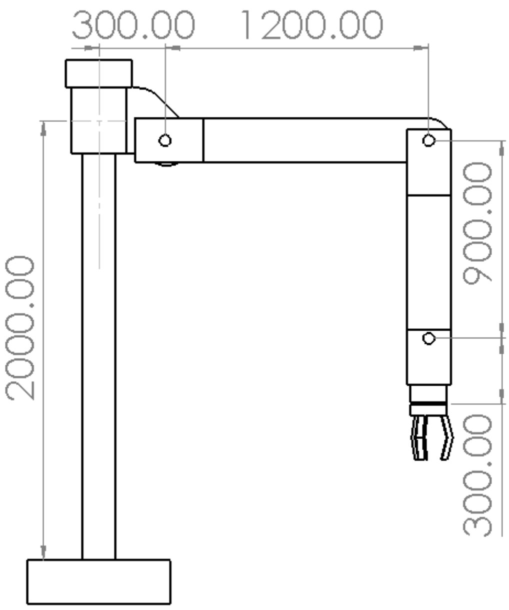


Figure 12: Drawing with dimensions

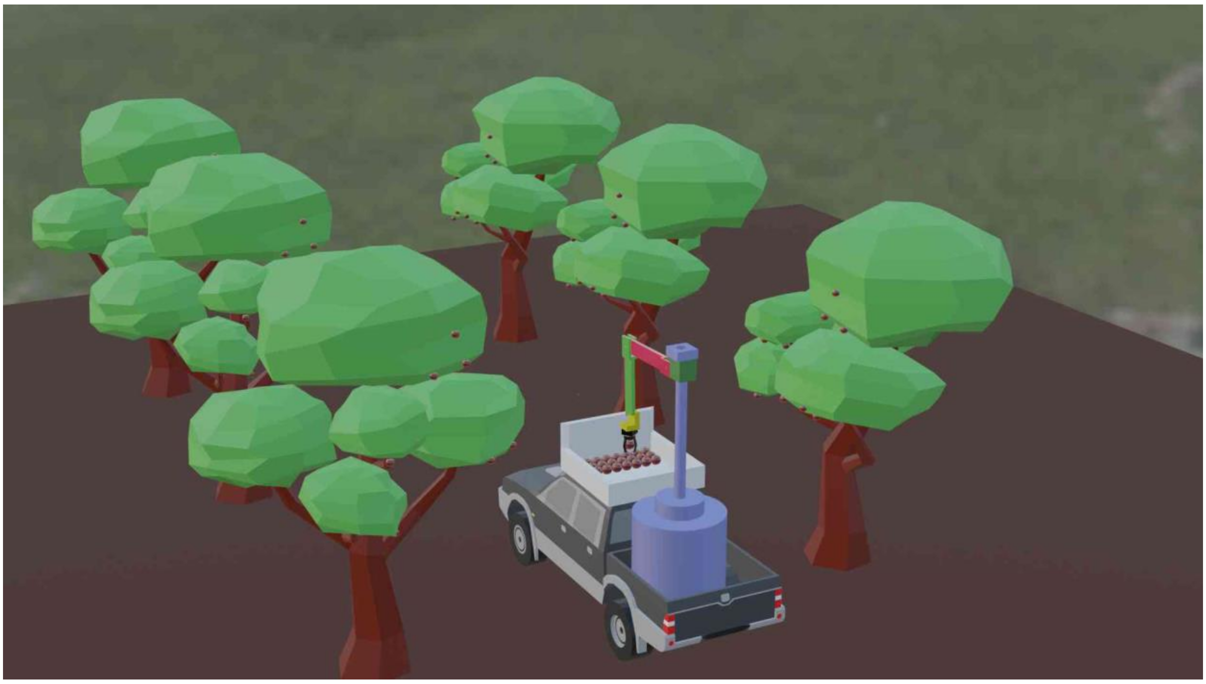


Figure 13:Isometric view of the home position

DH frame link assignment:

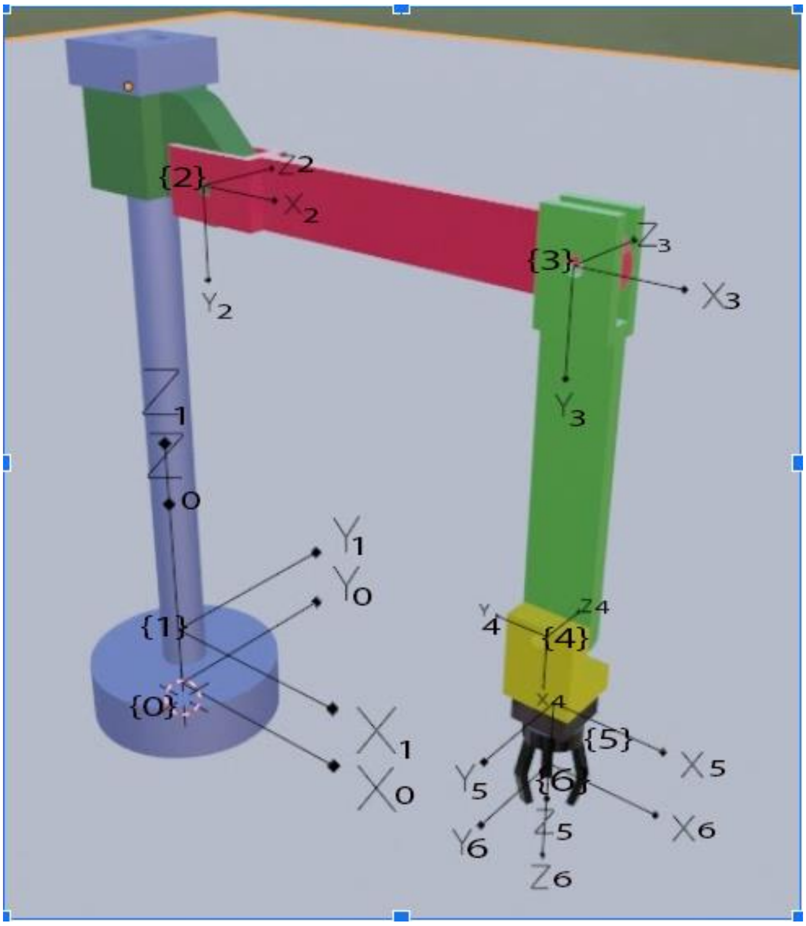


Figure 13 : DH frame link assignment

Joint link parameter table:

Link i	**θi** (degrees)	**αi** (degrees)	**ai**	**di**	**qi**
1	**t1**	**0**	**0**	**0**	**t1**
2	**0**	**-90**	**a2**	**d2**	**d2**
3	**t3**	**0**	**a3**	**0**	**t3**
4	**t4+90**	**0**	**a4**	**d4**	**t4**
5	**t5-90**	**-90**	**0**	**d5**	**t5**
6	**t6**	**0**	**0**	**d6**	**t6**

The values of link lengths and joint distances as obtained from the CAD model are:

a2 = 300mm; a3 = 1200 mm; a4 = 900 mm; d4 = -50mm ; d5 = 100mm ; d6 = 300mm;

MATLAB Code for forward kinematics:

```
clc
clear variables
syms theta alph a d;
syms t1 d2 t3 t4 t5 t6 t7;

% Input joint link parameter table below in the order [a alpha d theta]
linkVar = [0 0 0 t1; 300 sym(-pi/2) d2 0; 1200 0 0 t3; 900 0 -50 sym(t4+pi/2);
0 sym(-pi/2) 100 sym(t5-pi/2); 0 0 300 t6];

T = @(a, alph, d, theta)[cos(theta) -sin(theta)*cos(alph)
sin(theta)*sin(alph) a*cos(theta); sin(theta) cos(theta)*cos(alph) -
cos(theta)*sin(alph) a*sin(theta); 0 sin(alph) cos(alph) d; 0 0 0 1];

[r,~] = size(linkVar);
Tf = eye(4);

for i=1:r
    t = T(linkVar(i,1),linkVar(i,2),linkVar(i,3),linkVar(i,4));
    fprintf('%dT%d = \n', i-1, i)
    disp(simplify(t))
    Tf = simplify(Tf*t);
end

disp('Tf=')
disp(simplify(Tf))
```

Transformation Matrices:

$$^{0}T_1 = \begin{bmatrix} C1 & -S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{3}T_4 = \begin{bmatrix} -S4 & -C4 & 0 & -900 * S4 \\ C4 & -S4 & 0 & 900 * C4 \\ 0 & 0 & 1 & -50 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{1}T_2 = \begin{bmatrix} 1 & 0 & 0 & 300 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & d2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{4}T_5 = \begin{bmatrix} S5 & 0 & C5 & 0 \\ -C5 & 0 & S5 & 0 \\ 0 & -1 & 0 & 100 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{2}T_3 = \begin{bmatrix} C3 & -S3 & 0 & 1200 * C3 \\ S3 & C3 & 0 & 1200 * S3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{5}T_6 = \begin{bmatrix} C6 & -S6 & 0 & 0 \\ S6 & C6 & 0 & 0 \\ 0 & 0 & 1 & 300 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

End effector matrix:

$$
{}^{0}T_6 = \begin{bmatrix} S1 * S6 + C1 * C345 * C6 & S1 * C6 - C1 * C345 * S6 & -C1 * S345 & 300 * C1 * (1 + 4 * C3 - 3 * S34 - S345) - 50 * S1 \\ C345 * C6 * S1 - C1 * S6 & -C1 * C6 - C345 * S1 * S6 & -S1 * S345 & 300 * S1 * (1 + 4 * C3 - 3 * S34 - S345) + 50 * C1 \\ -C6 * S345 & S6 * S345 & -C345 & d2 - 300 * C345 - 900 * C34 - 1200 * S3 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Where:

- S1 = sin(t1)
- S34 = sin(t3 + t4)
- S345 = sin(t3 + t4 + t5)

Enumeration of Forward Kinematics for different positions:

1. **Forward Kinematics at home position:**

At home position, the gripper faces vertically downwards (please refer to figure 8) . The home position parameter table looks like the following:

Link i	**θi (degrees)**	**αi (degrees)**	**ai (mm)**	**di (mm)**	**qi**
1	0	0	0	0	t1
2	0	-90	300	2000	d2
3	0	0	1200	0	t3
4	90	0	900	-50	t4
5	-90	-90	0	100	t5
6	0	0	0	300	t6

2. **Forward Kinematics during plucking motion (figure 9):**

The position parameter table for plucking motion looks like the following:

Link i	**θi (degrees)**	**αi (degrees)**	**ai (mm)**	**di (mm)**	**qi**
1	90	0	0	0	t1
2	0	-90	300	2000	d2
3	15.294	0	1200	0	t3
4	-35.886+90	0	900	-50	t4
5	20.592-90	-90	0	100	t5
6	270	0	0	300	t6

3. **Forward Kinematics during palletizing:**

During these calculations, we are placing the apple at the far end (corner) of the palette at point C(1800,500) according to the robot base (please refer to figure 3) . The position parameter table for plucking motion looks like the following:

Link i	**θi (degrees)**	**αi (degrees)**	**ai (mm)**	**di (mm)**	**qi**
1	13.99	0	0	0	t1
2	0	-90	300	1760	d2
3	56.63	0	1200	0	t3
4	-100.083+90	0	900	-50	t4
5	-30.282-90	-90	0	100	t5
6	0	0	0	300	t6

Inverse Kinematics Solution

MATLAB code used to find inverse kinematics solution

```
clc
clear variables
syms t1 d2 t3 t4 t5 t6 t7;
syms nx ox ax dx ny oy ay dy nz oz az dz
T = @(a, alph, d, theta)[cos(theta) -sin(theta)*cos(alph) sin(theta)*sin(alph)
a*cos(theta); sin(theta) cos(theta)*cos(alph) -cos(theta)*sin(alph) a*sin(theta); 0
sin(alph) cos(alph) d; 0 0 0 1];

Te = [nx ox ax dx; ny oy ay dy; nz oz az dz; 0 0 0 1];    % End effector matrix
Tf = [sin(t1)*sin(t6)+cos(t3+t4+t5)*cos(t1)*cos(t6),cos(t6)*sin(t1)-
cos(t3+t4+t5)*cos(t1)*sin(t6),-sin(t3+t4+t5)*cos(t1),300*cos(t1)-
50*sin(t1)+1200*cos(t1)*cos(t3)-300*sin(t3+t4+t5)*cos(t1)-
900*cos(t1)*cos(t3)*sin(t4)-
900*cos(t1)*cos(t4)*sin(t3);cos(t3+t4+t5)*cos(t6)*sin(t1)-cos(t1)*sin(t6),-
cos(t1)*cos(t6)-cos(t3+t4+t5)*sin(t1)*sin(t6),-
sin(t3+t4+t5)*sin(t1),50*cos(t1)+300*sin(t1)+1200*cos(t3)*sin(t1)-
300*sin(t3+t4+t5)*sin(t1)-900*cos(t3)*sin(t1)*sin(t4)-900*cos(t4)*sin(t1)*sin(t3);-
sin(t3+t4+t5)*cos(t6),sin(t3+t4+t5)*sin(t6),-cos(t3+t4+t5),d2-300*cos(t3+t4+t5)-
900*cos(t3+t4)-1200*sin(t3); 0,0,0,1]; % 0T6 matrix

%% Make changes only here
disp('LHS = ')
disp(simplify(inv(T(0,0,0,t1))*Te))      % Perform pre/post multiplication
disp('RHS = ')
disp(simplify(inv(T(0,0,0,t1))*Tf))      % Perform pre/post multiplication

% Use inverse from here
% inv(T(0,0,0,t1)) inv(T(300,sym(-pi/2),d2,0)) inv(T(1200,0,0,t3))
% inv(T(900,0,-50,sym(t4+pi/2))) inv(T(0,sym(-pi/2),100,sym(t5-pi/2)))
inv(T(0,0,300,t6))
```

The inverse kinematics solution is found as follows:

$$T_E = \begin{bmatrix} n_x & o_x & a_x & d_x \\ n_y & o_y & a_y & d_y \\ n_z & o_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^0T_6$$

Eq. 1

- **Step-1**

Pre-multiply Eq. 1 with inverse of 0T1:

```
disp('LHS = ')
disp(simplify(inv(T(0,0,0,t1))*Te))     % Perform pre/post multiplication
disp('RHS = ')
disp(simplify(inv(T(0,0,0,t1))*Tf))     % Perform pre/post multiplication
```

Taking the element (2,3), we get:

=> ay*C1 - ax*S1 = 0

=> **θ1 = Atan2d(ay, ax)** Eq. 2

- **Step-2**

Taking elements (2,1) and (2,2) from the previous matrix, we get:

=> S6 = -(ny*C1 - nx*S1)

=> C6 = -(oy*C1 - ox*S1)

=> **θ6 = Atan2d(-ny*C1 + nx*S1, -oy*C1 + ox*S1)** Eq. 3

- **Step-3**

Pre-multiply Eq. 1 with inverse of 1T2, 0T1 and post-multiply with 5T6:

```
disp('LHS = ')
disp(simplify(inv(T(300,sym(-
pi/2),d2,0))*inv(T(0,0,0,t1))*Te*inv(T(0,0,300,t6))))  % Perform
pre/post multiplication
disp('RHS = ')
disp(simplify(inv(T(300,sym(-
pi/2),d2,0))*inv(T(0,0,0,t1))*Tf*inv(T(0,0,300,t6))))  % Perform
pre/post multiplication
```

Taking elements (1,3) and (2,3), we get:

=> S345 = -ax*C1 - ay*S1

=> C345 = -az,

=> **θ345 = Atan2d(-ax*C1 - ay*S1, -az)** Eq. 4

- **Step-4**

Pre-multiply Eq. 1 with inverse of 2T3, 1T2, 0T1, and post-multiply with inverse of 5T6:

```
disp('LHS = ')
disp(simplify(inv(T(1200,0,0,t3))*inv(T(300,sym(-
pi/2),d2,0))*inv(T(0,0,0,t1))*Te*inv(T(0,0,300,t6)))) % Perform
pre/post multiplication
disp('RHS = ')
disp(simplify(inv(T(1200,0,0,t3))*inv(T(300,sym(-
pi/2),d2,0))*inv(T(0,0,0,t1))*Tf*inv(T(0,0,300,t6)))) % Perform
pre/post multiplication
```

Taking elements (1,2) and (2,2), we get:

=> ox*C1*C3*C6 - nz*S3*S6 - oz*C6*S3 + nx*C1*C3*S6 + oy*C3*C6*S1 + ny*C3*S1*S6 = 0
=> -ox*C1*C6*S3 - nz*C3*S6 - oz*C3*C6 - nx*C1*S3*S6 - oy*C6*S1*S3 - ny*S1*S3*S6 = 0

Add both equations and simplify:

=> (C3 - S3)*[ox*C1*C6 + nx*C1*S6 + oy*C6*S1 + ny*S1*S6] = (C3 + S3)*[nz*S6 + oz*C6]

=> (C3 - S3) * k1= (C3 + S3) * k2

where, **k1 = ox*C1*C6 + nx*C1*S6 + oy*C6*S1 + ny*S1*S6**

and, **k2 = nz*S6 + oz*C6,** are constants.

=> C3 * (k1 - k2) = S3 * (k1 + k2)

=> S3 / C3 = (k1 - k2) / (k1 + k2)

=> **θ3 = Atan2d((k1 - k2), (k1 + k2))** Eq. 5

- **Step-5**

Post-multiply Eq. 1 with inverse of 5T6:

```
disp('LHS = ')
disp(simplify(Te*inv(T(0, 0, 300, t6))))    % Perform pre/post
multiplication
disp('RHS = ')
disp(simplify(Tf*inv(T(0, 0, 300, t6))))    % Perform pre/post
multiplication
```

Taking the element (1,4), we get:

=> k3 - 900*C1*(S34) = dx - ax*300

where, **k3 = C1*(300 + 1200*C3) + 50*S1 - 100*S1**, is a constant.

=> $S34 = k4$

=> $C34 = \pm \sqrt{1-(k4^2)}$

where, **k4 = -(dx - ax*300 - k3)/900*C1,** is a constant.

=> **θ3 + θ4 = Atan2d(k4, ± sqrt(1-($k4^2$))** Eq. 6

From Eq. 5:

=> **θ4 = Atan2d(k4, ± sqrt(1-($k4^2$)) - Atan2d((k1 - k2), (k1 + k2))** Eq. 7

There can be two solutions of in eq.7 for θ4 (One taking the positive sign and one with the negative sign)

From Eq. 4, and Eq. 6:

=> **θ5 = Atan2d(-ax*C1 - ay*S1, -az) - Atan2d(k4, ± sqrt(1-($k4^2$))** Eq. 8

Here in Eq.8 in the second term on the right hand side, it is visible that there can be two solutions for θ5 (one taking the positive sign and one taking the negative sign).

- **Step-6**

Taking the element (3,4) from the previous matrix, we get:

=> **d2 = dz - az*300 + 900*C34 + 1200*S3** Eq. 9

Hence, the inverse kinematics solution is now complete. The joint variables have been found in the above equations 1 to 9, using the constant k1, k2, k3 and k4.

Existence of solutions:

A close examination of the equations and constants obtained above reveals that the solutions to the inverse kinematics problem exist only if the term k4 has a value in the range [-1, 1].

Multiplicity of solutions:

The variable	**Multiplicity**
θ1	1
d2	1
θ3	1
θ4	2
θ5	2
θ6	1

Jacobian and Singularities Code

Generalized MATLAB code to find link velocities and Jacobian

```
clc
clear variables

%% Inputs (only make changes here)
syms t1 a2 d2 t3 a3 t4 a4 d4 t5 d5 t6 d6    % Variables in joint link parameter
table
syms t1dot d2dot t3dot t4dot t5dot t6dot    % Joint velocities

% List the joint variables q_i
q_i = [t1 d2 t3 t4 t5 t6];

% Input joint link parameter table below in the order [a alpha d theta]
linkVar = [0 0 0 t1; 300 sym(-pi/2) d2 0; 1200 0 0 t3; 900 0 -50 sym(t4+pi/2);
0 sym(-pi/2) 100 sym(t5-pi/2); 0 0 300 t6]; % a alpha d theta

% Define the joint velocities
dots = [t1dot d2dot t3dot t4dot t5dot t6dot];

% Define the type of joint, where 0 = prismatic, and 1 = revolute
joints = [1 0 1 1 1 1];

%% Initialize

zCap = [0 0 1]';
iDi = [0 0 0 1]';
omega = [0 0 0]';

%% Finding linear and angular link velocities

T = @(a, alph, d, theta)[cos(theta) -sin(theta)*cos(alph) sin(theta)*sin(alph)
a*cos(theta); sin(theta) cos(theta)*cos(alph) -cos(theta)*sin(alph)
a*sin(theta); 0 sin(alph) cos(alph) d; 0 0 0 1];

[r,~] = size(linkVar);
Tf = eye(4);

for i=1:r

    t = T(linkVar(i,1),linkVar(i,2),linkVar(i,3),linkVar(i,4));
    tsave = Tf;
    Tf = simplify(Tf*t);
    fprintf('0T%d = \n', i)
    disp(simplify(Tf))

    % Find linear velocity
    vel = [0 0 0 0]';
    for j = 1:i
        vel = vel + simplify(diff(Tf, q_i(j))*iDi*dots(j));
    end
    fprintf('V%d = \n', i)
    disp(vel(1:3))
```

```
    % Find angular velocity
    omega = omega + joints(i)*simplify(tsave(1:3,1:3)*zCap*dots(i));
    fprintf('omega%d = \n', i)
    disp(omega)

end

%% Finding the Jacobian
tsave = Tf;
Tf = eye(4);
J = sym(zeros(r));

for i=1:r

    Pi_1 = Tf(1:3,3);
    if joints(i)==0
        fprintf('J%d = \n', i)
        J_single = [Pi_1; zeros(3,1)];
        disp(J_single)
        J(:,i) = J_single;
    else
        Pi_1_n = tsave(:,4) - Tf(:,4);
        fprintf('J%d = \n', i)
        J_single = [simplify(cross(Pi_1, Pi_1_n(1:3))); Pi_1];
        disp(J_single)
        J(:,i) = J_single;
    end
    t = T(linkVar(i,1),linkVar(i,2),linkVar(i,3),linkVar(i,4));
    Tf = simplify(Tf*t);

end

fprintf('Final Jacobian, J = \n')
disp(J)

%% Taking first 4 rows of the first 4 columns
Jprime1 = J(1:4, 1:4);
disp('Jprime1 =')
disp(Jprime1)
disp('|Jprime1| = ')
disp(simplify(det(Jprime1)))
disp('invJprime1 = ')
disp(simplify(inv(Jprime1)));

% Taking 4th and 5th rows of the last 2 columns:
Jprime2 = J(4:5,5:6);
disp('Jprime2 =')
disp(Jprime2)
disp('|Jprime2| = ')
disp(simplify(det(Jprime2)))
disp('invJprime2 = ')
disp(simplify(inv(Jprime2)));
```

The Jacobian matrices obtained are as follows:

$$J1 = \begin{bmatrix} 300 * S1(S345 - 1) - 1200 * C3 * S1 - 50 * C1 + 900 * S1 * S34 \\ 300 * C1 - 50 * S1 + 1200 * C1 * C3 - 300 * S345 * C1 - 900 * C1 * S34 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$J3 = \begin{bmatrix} -C1 * (300 * C345 + 900 * C34 + 1200 * S3) \\ -S1 * (300 * C345 + 900 * C34 + 1200 * S3) \\ 300 * S345 + 900 * S34 - 1200 * C3 \\ -S1 \\ C1 \\ 0 \end{bmatrix}$$

$$J4 = \begin{bmatrix} -C1 * (300 * C345 + 900 * C34) \\ -S1 * (300 * C345 + 900 * C34) \\ 300 * S345 + 900 * S34 \\ -S1 \\ C1 \\ 0 \end{bmatrix}$$

$$J5 = \begin{bmatrix} -300 * C345 * C1 \\ -300 * C345 * S1 \\ 300 * S345 \\ -S1 \\ C1 \\ 0 \end{bmatrix}$$

$$J6 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -S345 * C1 \\ -S345 * S1 \\ -C345 \end{bmatrix}$$

Combining all the above jacobian matrices, we get:

Jacobian matrix, J =

$$J = \begin{bmatrix} 300 * S1(S345 - 1) - 1200 * C3 * S1 - 50 * C1 + 900 * S1 * S34, & 0, & -C1 * (300 * C345 + 900 * C34 + 1200 * S3), & -C1 * (300 * C345 + 900 * C34), & -300 * C345 * C1, & 0 \\ 300 * C1 - 50 * S1 + 1200 * C1 * C3 - 300 * S345 * C1 - 900 * C1 * S34, & 0, & -S1 * (300 * C345 + 900 * C34 + 1200 * S3), & -S1 * (300 * C345 + 900 * C34) & -300 * C345 * S1, & 0 \\ 0, & 1, & 300 * S345 + 900 * S34 - 1200 * C3, & 300 * S345 + 900 * S34 & 300 * S345, & 0 \\ 0, & 0, & -S1, & -S1 & -S1 & -S345 * C1 \\ 0, & 0, & C1, & C1 & C1 & -S345 * S1 \\ 1 & 0 & 0 & 0 & 0 & -C345 \end{bmatrix}$$

Considering only first four columns and first four rows corresponding to links which are not a part of wrist we get :

<u>Jprime1 =</u>

$$Jprime1 = \begin{bmatrix} 300 * S345 * S1 - 300 * S1 - 1200 * C3 * S1 - 50 * C1 + 900 * S1 * S34, & 0, & -C1 * (300 * C345 + 900 * C34 + 1200 * S3), & -C1 * (300 * C345 + 900 * C34) \\ 300 * C1 - 50 * S1 + 1200 * C1 * C3 - 300 * S345 * C1 - 900 * C1 * S34, & 0, & -S1 * (300 * C345 + 900 * C34 + 1200 * S3), & -S1 * (300 * C345 + 900 * C34) \\ 0, & 1, & 300 * S345 + 900 * S34 - 1200 * C3, & 300 * S345 + 900 * S34 \\ 0, & 0, & -S1, & -S1 \end{bmatrix}$$

<u>The determinant is given by:</u>

$$|Jprime1| = 360000 * sin(theta1) * [(3 * cos(2 * theta3 + theta4))/2 + 2 * sin(2 * theta3) + cos(2 * theta3 + theta4 + theta5)/2 - cos(theta4 + theta5)/2 - (3 * cos(theta4))/2 + sin(theta3)]$$

The <u>inverse of this jacobian</u> can be found out using the code mentioned above. We have not elaborated it here for brevity.

Considering the last 2 columns and 4th and 5th rows of the Jacobian to account for the wrist, we have:

$$Jprime2 = \begin{bmatrix} -S1 & -S345 * C1 \\ C1 & -S345 * S1 \end{bmatrix}$$

DetJprime2 =

$$|Jprime2| = sin(theta3 + theta4 + theta5)$$

invJprime2 =

[-sin(t1), cos(t1)]

[-cos(t1)/sin(t3 + t4 + t5), -sin(t1)/sin(t3 + t4 + t5)]

<u>Singularities</u>:

- Equating |Jprime1| to zero, we have:
  1) sin(θ_1) = 0 i.e. θ_1 = 0 or π or
  2) (3*cos(2*θ_3 + θ_4))/2 + 2*sin(2*θ_3) + cos(2*θ_3 + θ_4 + θ_5)/2 - cos(θ_4 + θ_5)/2 - (3*cos(θ_4))/2 + sin(θ_3) = 0

     By observation, this equation collapses to zero when θ_3 = 0.
     To verify that there is no other value of θ_3, we ran a python script which looped over all values of θ_3,θ_4 and θ_5 from 0 to 2π. We found that apart from θ_3 = 0, there are no other possibilities for which the expression becomes 0. We have uploaded the python code and the csv file (singularity.csv) of singularities generated in this drive link. Essentially, for θ_3=0, at all values of θ_4 and θ_5, a singularity is occurring.

- Equating |Jprime2| to zero :
  We can clearly see from the above two J' (J-prime) matrices, that the singularities exist where (θ_3+θ_4+θ_5) = 0, 90, 180, 270. (See fig. 14) This means that the number of combinations of the three quantities is infinite. In this case we have no option but to analyze the torques near these values to pull out pseudo singularities.

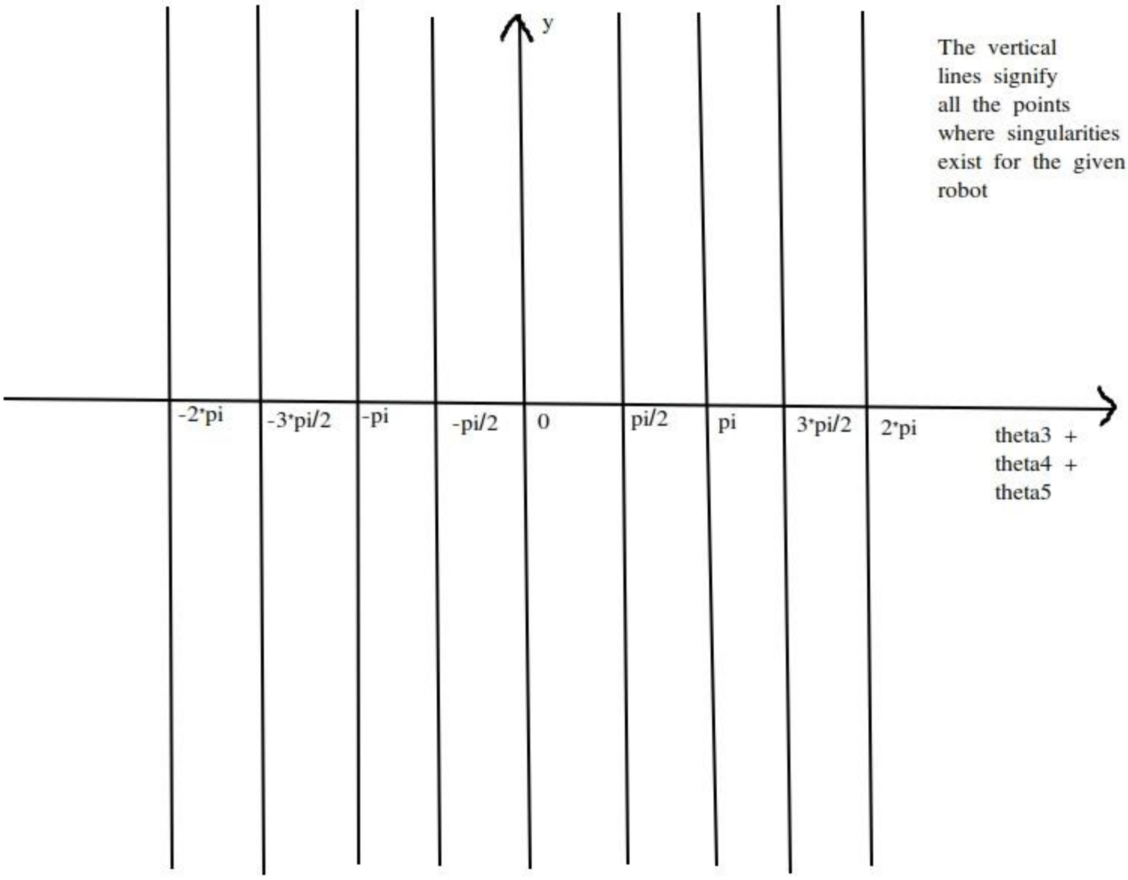


Fig. 14

Sensors:

The sensors which will be used are:

1. **Camera(1080 HDR MIPI Camera MCAM400)** for detection of the ripe apple on the tree. (Click here)
2. **Ultrasonic sensor** (U300.R50-GP1J.72N) can be used for distance measurement between the gripper and the apple. The sensor will be placed inside the gripper. The sensor has a range between 0mm and 500mm and can be used effectively for the measurement of depth of apple from the gripper.

References:

- Baeten, Johan & Donné, Kevin & Boedrij, Sven & Beckers, Wim & Claesen, Eric. (2007). Autonomous Fruit Picking Machine: A Robotic Apple Harvester. Springer Tracts in Advanced Robotics. 42. 10.1007/978-3-540-75404-6_51.
- "Automatic fruit picker demonstration by FF Robotics" YouTube, GoodFruitGrower, 3rd April 2017, https://youtu.be/UaL3UxUclKY.
- "Robotic apple picker trials continue in Washington by FF Robotics" YouTube, GoodFruitGrower, 28th October 2016 https://youtu.be/mS0coCmXiYU.
- "Robotics Arms Race by FF Robotics" YouTube, GoodFruitGrower, 30th November 2019 https://youtu.be/-PtqZA2enkQ.