Imperial College London

Robotic Manipulation Report

Group: EVE

Members: Kaling Ng, Jing Lin Loh, Nur Mohd Zafer

Module: Robotic Manipulation

Lecturer: Ad Spiers

Contents

Task 1 – Modelling the Robot

1.a Co-ordinate Frame & DH Notation Table: Forward Kinematics

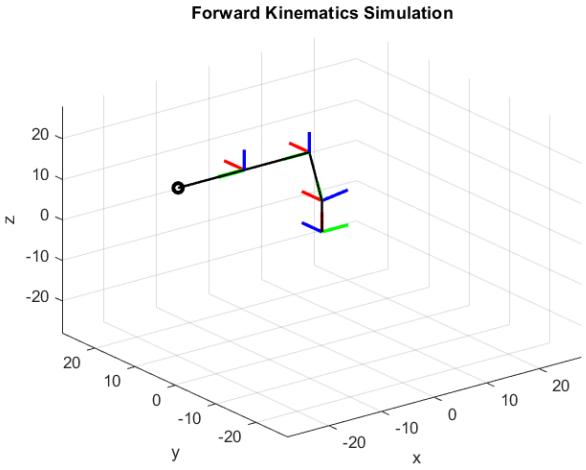

Figure 1: Co-ordinate Frame

Figure 1 illustrates the modelled co-ordinate frame for the robot. The green, blue, and red line is the x, y and z-axis respectively. The co-ordinate frames are modelled so that the x-axis goes along the link length, z-axis goes along the joint axis while the y-axis follows the right-hand rule. The Open Manipulator-X robot is a 4DOF robot. The first joint rotates perpendicularly (denoted by θ_1) in comparison to the other joints.

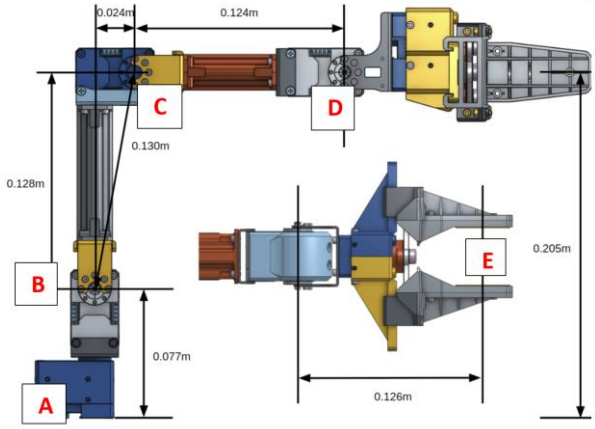

Figure 2: Open Manipulator-X Joints

The offset angle between joint B and joint C of the robot, $\gamma = \tan^{-1}\frac{0.024}{0.128} = 10.62°$, is considered in our forward kinematics calculation.

i	α_{i-1}	a_{i-1}	d_i	$\boldsymbol{\theta_i}$
1	0	0	7.7	θ_1
2	-90	0	0	-90- γ- θ_2
3	0	13	0	0
4	0	0	0	γ-90
5	0	12.4	0	θ_3
6	0	12.6	0	θ_4

Table 1: DH Notation Table

This version of the DH Transform was used in the forward kinematics calculation:

$$T_i = \begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & \alpha_{i-1} \\ sin\theta_i cos\alpha_{i-1} & cos\theta_i cos\alpha_{i-1} & -sin\alpha_{i-1} & -sin\alpha_{i-1} * d_i \\ sin\theta_i sin\alpha_{i-1} & cos\theta_i sin\alpha_{i-1} & cos\alpha_{i-1} & cos\alpha_{i-1} * d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The base joint, A, is positioned at (0, 0, 0) and is characterized by the 4x4 identity matrix,

$$T_A = T_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

From joint A to joint B: $T_B = T_A * T_1$

From joint B to joint C: $T_C = T_B * T_2 * T_3$

From joint C to joint D: $T_D = T_c * T_4 * T_5$

From joint D to joint E: $T_E = T_D * T_6$

1.b. Simulation: Forward Kinematics

Angle	Degrees
θ_1	0° to 28°
θ_2	0° to 28°
θ_3	0° to -28°
θ_4	0° to 28°

Table 2: Table of Angles for Simulation

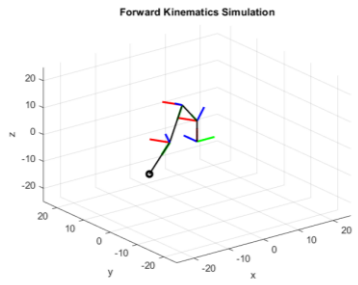

Figure 3: Whole view

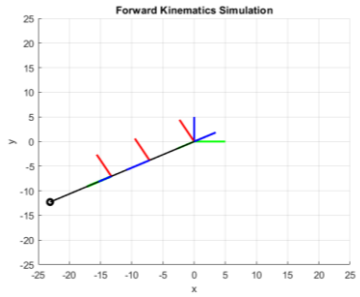

Figure 4: Top view

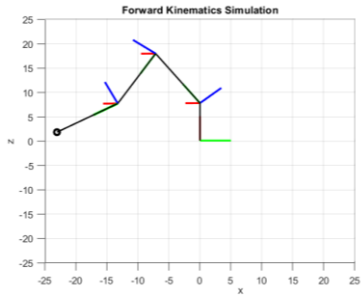

Figure 5: Side view

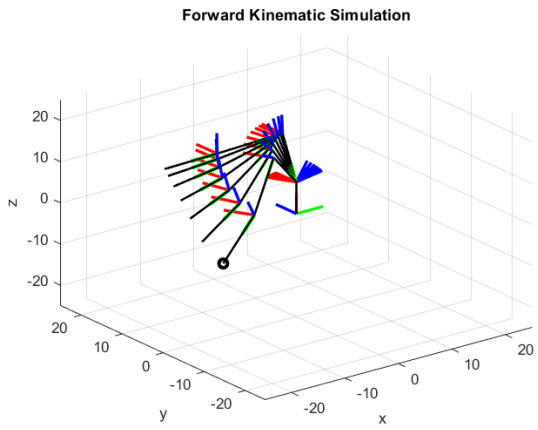

Figure 6: Frame-by-frame simulation

Case	Joint Angles				End-effector Coordinates		
	θ_1	θ_2	θ_3	θ_4	p_x	p_y	p_z
1	1	1	-1	1	-27.6047	-0.4818	19.7809
2	3	3	-3	3	-27.9376	-1.4641	18.3789
3	6	6	-6	6	-28.2228	-2.9663	16.2618
4	10	10	-10	10	-28.2038	-4.9731	13.4382
5	15	15	-15	15	-27.5584	-7.3842	9.9608
6	21	21	-21	21	-25.9477	-9.9604	5.9575
7	28	28	-28	28	-27.9376	-1.4641	1.6616

Table 3: Results of forward kinematic modelling

Figures 3 to 6 illustrates the forward kinematic simulation when the program is fed with different theta values. By plotting the robot in 3D space, one can easily view the different sides of the robot. Figure 6 also shows all the frames of the robot as it moves through the 3D space to the desired position. The end-effector coordinates of each frame are shown on table 3.

1.b.1 Coordinate Frames

Coordinate frames are added for easy visualisation of the different axes. The green, blue and red lines respectively represent the x-, y- and z-axis. The x-axis is modelled along the link length, the z-axis is modelled along the axis of rotation and the y-axis is modelled using the right-hand rule.

The coordinate frames at each joint are plotted using the individual transformation matrix. The x-direction is obtained from $(1,1)$, $(2,1)$ and $(3,1)$ of the transformation matrix. The y-direction is obtained from $(1,2)$, $(2,2)$ and $(3,2)$ of the transformation matrix. The z-direction is obtained from $(1,3)$, $(2,3)$ and $(3,3)$ of the transformation matrix. The respective vector is then multiplied by 5 to form a vector of length 5.

1.c. Inverse Kinematics

(For calculations refer to Appendix Section 1) A solution to the inverse kinematics for the frame located between the jaws of the gripper is found. The target position of the end effector is characterized by [x, y, z]. By inputting the target position as well as the angle of the end-effector (denoted by φ), the inverse kinematics equations will determine the value of the angles $\theta_1, \theta_2, \theta_3$ and θ_4. Projection was used to simplify the calculations.

1.d. Graphical Simulation: Inverse Kinematics

Using inverse kinematics, a simulation is made where the robot arm traces a 10x10 square in each Cartesian plane.

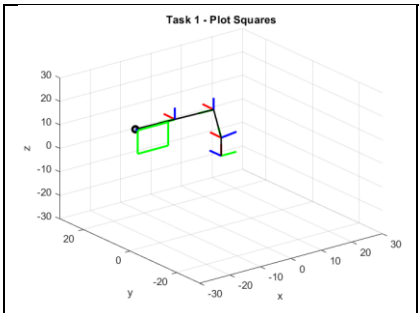

Figure 7: 10x10 square in XZ plane

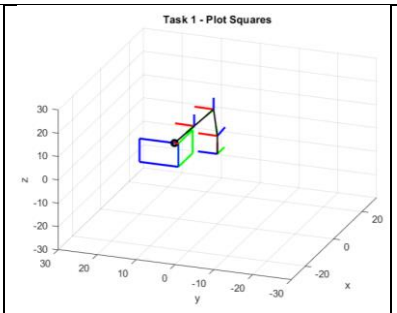

Figure 8: 10x10 square in YZ plane

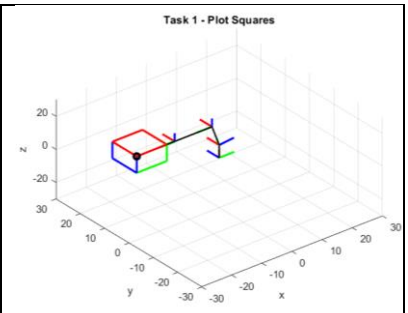

Figure 9: 10x10 square in XY plane

1.2 Generating smooth motion

A few factors were considered while transitioning from a virtual to a physical setting. One of the basic requirements for robotic manipulation is smoothness of motion. The following steps are applied to all tasks.

1.2.a Time-Based Profile

The drive mode in the robot arm is configured to time-based profile (4) to reduce jerkiness. Profile velocity is set according to each individual tasks to give the most optimal velocity with minimal jerkiness.

1.2.b Trajectory Planning

Waypoints were set between starting position and target position to obtain an optimal trajectory for the robot to move without crashing into other objects. To reduce the time taken to complete a task, waypoints were generated through trial and error such that unnecessary movements are reduced, such as producing just 1 waypoint between start and end position.

1.3 Software to Hardware

Before incorporating the inverse kinematic controller into the hardware system, some calibration was carried out by running some test codes to compare the angle values that were given by the controller and the encoder values generated by Dynamixel.

Due to the convention of our inverse kinematic calculations, before converting our angle values into encoder values, the angle values have to be shifted by 180°. The following table illustrates this:

	θ_1	θ_2	θ_3	θ_4
Inverse Kinematic Calculation	θ_1	θ_2	θ_3	θ_4
Dynamixel	$\theta_1 + 180$	$\theta_2 + 180$	$-\theta_3 + 180$	$-\theta_4 + 180$

Table 4: Angle Conversion

The value of θ_5 to control the opening and closing of the gripper is hardcoded by reading the value of θ_5 from the encoder since in our inverse kinematic calculations, the value of θ_5 is not considered.

Task 2 – Pick and Place

2a. Translation

The general flow of the process is as follow:

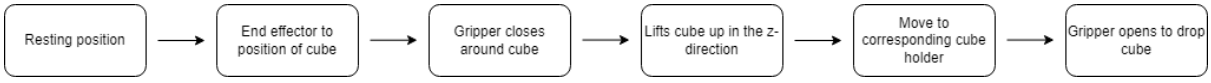

Figure 10: Translation Process

This task entails moving 3 different cubes to three different locations. Additional waypoints between starting and end positions are added if the robot arm hits any other objects while moving.

The precise locations of the cubes and cube holders are calculated by measuring the horizontal and vertical distance between the base location and the corresponding location. The starting and end coordinates are then fed into our inverse kinematics program to calculate the corresponding angles to achieve the target pose.

Different speeds were tested out when picking and placing the cubes. As there is a trade-off between speed and smoothness of the robot's motion, we decided on 500 for the profile velocity as it gives an optimal performance. A faster speed is also prioritised because in real life applications, speed is an important factor in applications such as picking and placing objects after accuracy.

2b. Rotation

Inward rotation is carried out in this task. The general flow of this task is as follow:

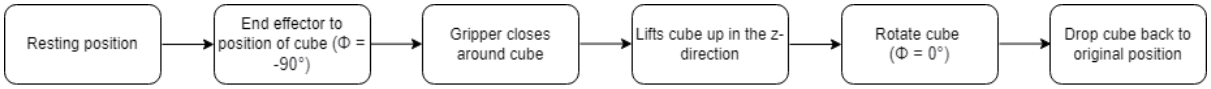

Figure 11: Rotation Process

The process is carried out twice if the red face is facing down and once if the red cube is facing outwards originally. A profile velocity of 1300 is used.

2c. Stacking

This task combines both the translation and rotation tasks. Cube 2 is first rotated once. Next, cubes 1 and 3 are moved to the location of cube 2. The height at which the cubes are released is increased by 2.5cm every time a cube is released. A profile velocity of 1300 is used.

Task 3 – Trajectory Following (Drawing)

3.a. Pen Gripper Design

The pen gripper was designed with the ability to pick the pen up from the surface in mind. The final version of the pen gripper can do that as well as pick up pens of varying sizes (provided that the pen is slightly bigger than the cylindrical tube through the gripper). The pen gripper design was purposely left opened on both sides to be able to pick up pens of different lengths. The pen is picked up from a pre-determined position at the start of the task and placed back at the same position at the end of the task.

3.b. Draw

The task was to be able to draw vertical lines, horizontal lines, diagonal lines, and arcs of varying angles.

The equation for each type of line/arc was determined and fed into an array which would be then used as the input to the inverse kinematics. The z-coordinate is kept constant as 8.5 throughout the drawing process as well as the angle of the pen ($\varphi = 0$). Therefore, only the x and y axis are needed to be considered for this portion of the task, which simplifies the whole process.

Our code is designed to run in a self-sustained fashion where only the step size, start and end coordinates are needed for the horizontal and vertical lines to be drawn.

The diagonal line uses the straight-line equation, $y = mx + c$, to determine the corresponding values of y to the set values of x.

As the arc needed to be generalisable, using polar equations was the easiest in order to be able to vary the start position $[x_{start}, y_{start}]$, centre $[x_{centre}, y_{centre}]$, radius and angle of the arc.

As the typical polar equations of a circle are centred around the origin (0, 0), some modifications need to be done to consider other cases where the arc is not centred at the origin. This can be done by offsetting the polar equations with the coordinates of the centre of the arc.

The arc should also be able to start anywhere, not only from the horizontal or vertical axis. To allow that, the starting theta needs to be determined as it varies depending on the location of the starting position. The starting theta is calculated by:

$$\theta_{start} = \text{atan}\left(\frac{x_{start} - x_{centre}}{y_{start} - y_{centre}}\right)$$

This would also shift the ending theta by the same amount. With this method, an arc can be drawn from any specified starting position (within reachable limits of the robot) easily. The direction in which the robot draws the arc can also be controlled by either increasing or decreasing the steps in the for loop.

3.c. Trajectory Planning

Due to the initial situation whereby, all groups had limited access with the robot, a simulation for task 3 was made in order to test out varying shapes/patterns and limits of the robot. The simulation considered the IK solutions therefore, the user can see whether certain patterns can actually be drawn by the robot or not. The simulation (see Appendix section 3) also included dots that indicated the centre of the cut-out squares on the base plate. This allowed for easy position tweaking and reduced the number of times needed to test on the robot. The simulation also allows the user to observe from which direction the pattern is being drawn from.

3.d. Speed

As speed was an important factor to this task, the aim was to pick up and place the pen at a slower speed as too fast of a speed would cause rough intense motions. Therefore, the profile velocity to pick up and place back the pen is set to 1300. In the drawing portion of the task however, the robot should draw the pattern as fast as possible, therefore the profile velocity is set to 250. It was tested and concluded that this was an appropriate speed as this did not affect the accuracy and smoothness of the pattern drawn.

Task 4 – All-In-One Garden

4.a. Motivation

The chosen task is a self-sustaining, all-in-one garden.

This idea was chosen due to its novelty and ability to demonstrate several of the key knowledge that was gained throughout this module. But most importantly, this idea showcases the potential of incorporating large-scaled robots in farming industries to streamline farming processes. Other functions such as turning the soil can also easily be carried out by the robot. Furthermore, using a robot to carry out mundane and repetitive tasks can save time and reduce risk of injuries.

4.b. Solving

The whole process of this task is as follow:

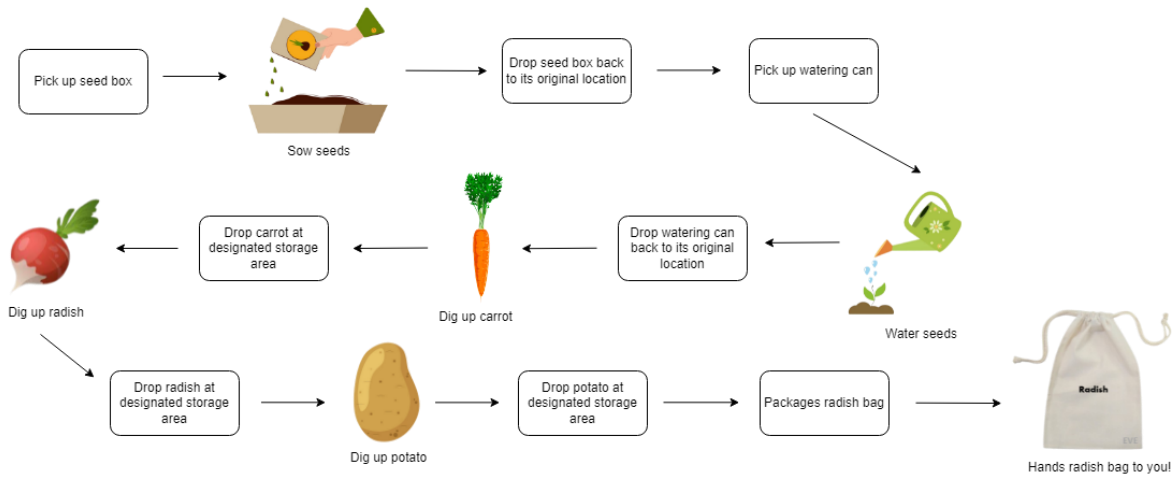

Figure 12: All-In-One Garden Process

The task is achieved by using inverse kinematics calculations and trajectory planning. The location of each object is recorded and fed into our code. The profile velocity of the robot arm is set to 1300 as due to the compact operating space, too fast of a speed may incur unnecessary damage to the robot if something foreign happens to be in the way. A slower speed is used when the gripper closes around the vegetables because we do not want the vegetables to be damaged by having a large force applied on it suddenly.

Trajectory planning is also used to prevent the robot from bumping into other objects in the tiny working space. Waypoints were calculated to allow a smooth trajectory and reduce unnecessary movements.

Furthermore, custom grippers are designed for the purpose of this task. First of all, a general gripper was designed to allow the robot arm to grip different kind of root vegetables effectively. Since it is more feasible to only have a single gripper for the entirety of the task, we also modified the handle of the watering can so that the existing gripper can hold the watering can stable and effectively. Different handle positions were tested on the watering can and it was concluded that the midpoint of the handle provides the best stability. On the other hand, a simple storage box is also designed to separate the storage area into 3 different areas for the carrot, potato, and radish. This demonstrates the ability to sort items into different categories/locations.

4.c. Applicability

In reality, the robot will not be limited to picking up carrots, potatoes, and radishes. Other vegetables could be picked up if the whole system is scaled up.

The motion and functionality of this advance feature can be used in other industries as well. For example, it can be used in factories, in assembly lines to move around both regular and irregular objects as well as packing groceries.

Appendix

1. Inverse Kinematics Calculations

These values are known/chosen by the user:

$$\gamma = \text{atan}(0.024/0.128) = 10.62°$$

$$\text{End effector position} = [x, y, z]$$

$$\text{End effector angle} = \varphi \text{ (in the range -90 to 90), where } \varphi = \theta_2 + \theta_3 + \theta_4.$$

The x and y coordinate of the end effector position are combined into r-coordinate using Pythagoras' equation. Projection of the r-z surface is also done.

$$r_3 = \sqrt{x^2 + y^2}$$

$$z_3 = z - 7.7$$

$$r_2 = r_3 - 12.6 * \cos(\varphi)$$

$$z_2 = z_3 - 12.6 * \sin(\varphi)$$

$$\cos(\theta_3) = \frac{r_2^2 + z_2^2 - (13^2 + 12.4^2)}{2 * 13^2 * 12.4^2}$$

A solution only exists when $-1 < \cos(\theta_3) < 1$.

There are two solutions for each angle:

Solutions for θ_3:

$$\theta_3^1 = \text{acos}(\cos(\theta_3)) - \gamma + 90$$

$$\theta_3^2 = -\text{acos}(\cos(\theta_3)) - \gamma + 90$$

$$k_1 = 13 + 12.4 * \cos(\text{acos}(\text{acos}(\theta_3)))$$

$$k_2^1 = 12.4 * \sin(\text{acos}(\theta_3))$$

$$k_2^2 = 12.4 * \sin(-\text{acos}(\theta_3))$$

Solutions for θ_2:

$$\theta_2^1 = 90 - \text{atan}\left(\frac{z_2}{r_2}\right) - \text{atan}\left(\frac{k_2^1}{k_1}\right) - \gamma$$

$$\theta_2^2 = 90 - \text{atan}\left(\frac{z_2}{r_2}\right) - \text{atan}\left(\frac{k_2^2}{k_1}\right) - \gamma$$

Solutions for θ_4:

$$\theta_4^1 = \varphi - \theta_2^1 - \theta_3^1$$

$$\theta_4^2 = \varphi - \theta_2^2 - \theta_3^2$$

Solutions for θ_1:

$$\theta_1^1 = \text{atan}\left(\frac{y}{x}\right)$$

$$\theta_1^2 = -\text{atan}\left(\frac{y}{x}\right)$$

As the answers come in pairs, there are four solutions:

Solution 1 = $[\theta_1^1, \theta_2^1, \theta_3^1, \theta_4^1]$

Solution 2 = $[\theta_1^1, \theta_2^2, \theta_3^2, \theta_4^2]$

Solution 3 = $[\theta_1^2, \theta_2^1, \theta_3^1, \theta_4^1]$

Solution 4 = $[\theta_1^2, \theta_2^2, \theta_3^2, \theta_4^2]$

2. CAD Models

2.1 Task 3 Pen Gripper

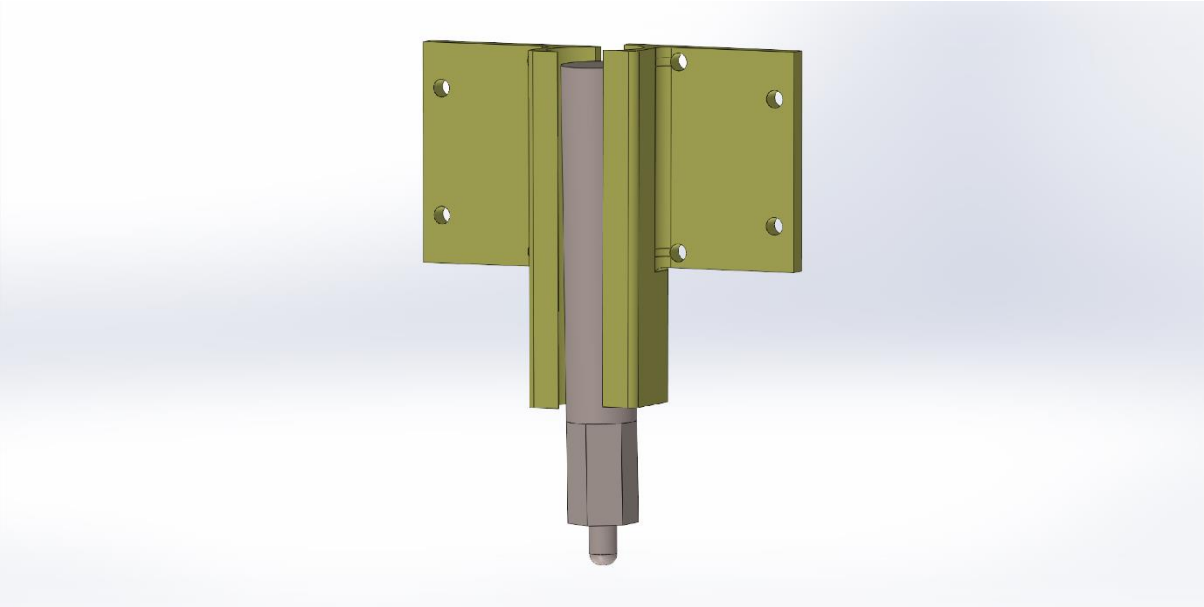

Figure 13: Pen Gripper with Pen

2.2 Task 4 Vegetable Digger

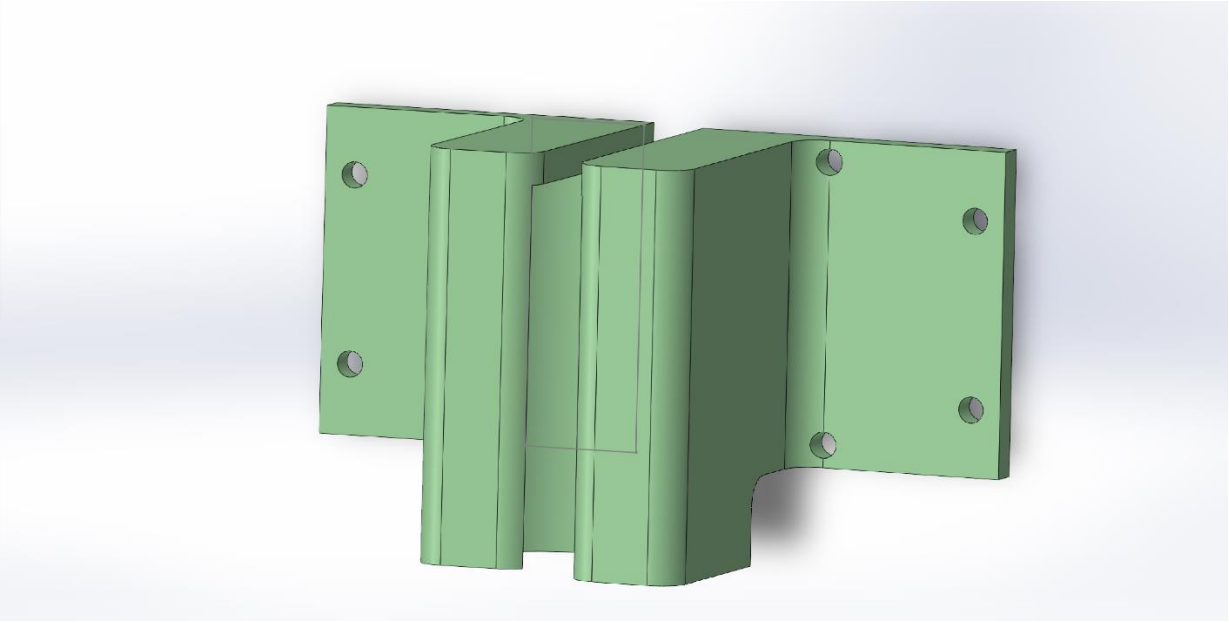

Figure 14: Vegetable Digger

2.3 Task 4 Watering Can Modifier

Figure 15: Watering Can Holder

2.4 Task 4 Sorting Box

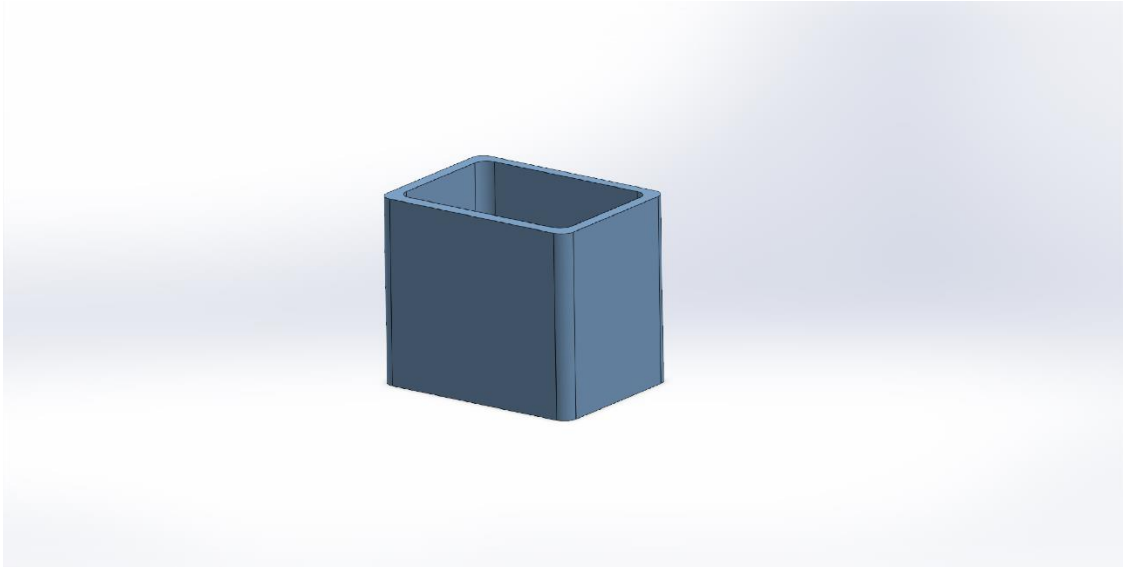

Figure 16: Sorting Box

3. MATLAB simulation for Drawing

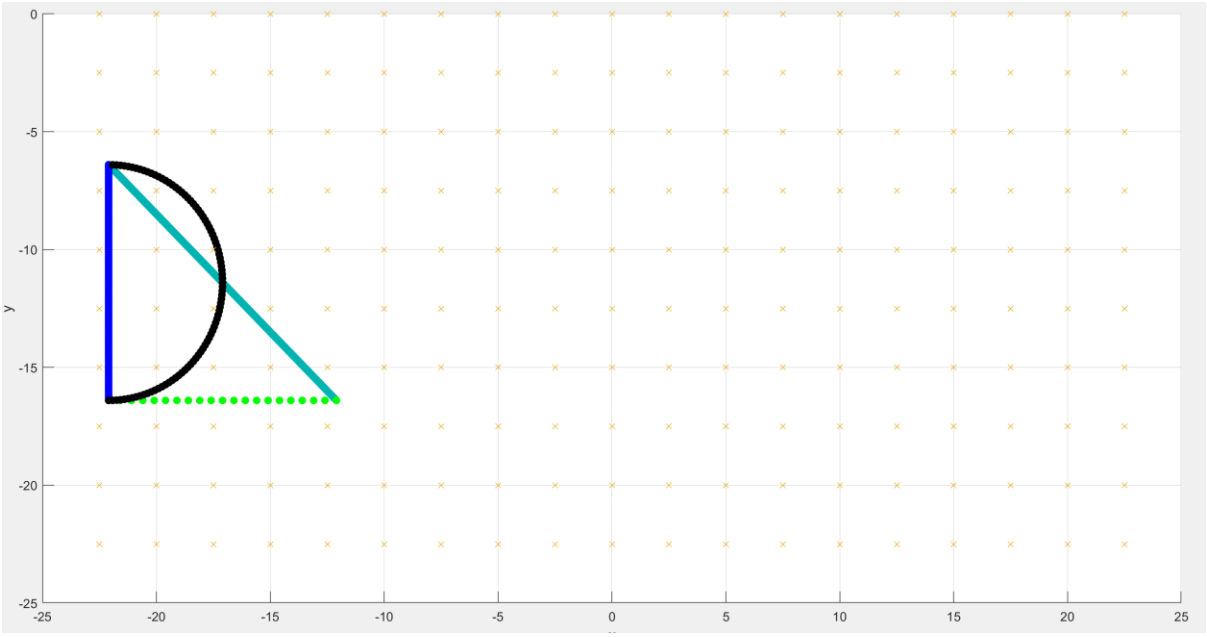

Figure 17: Task 3 Video Demo Simulation

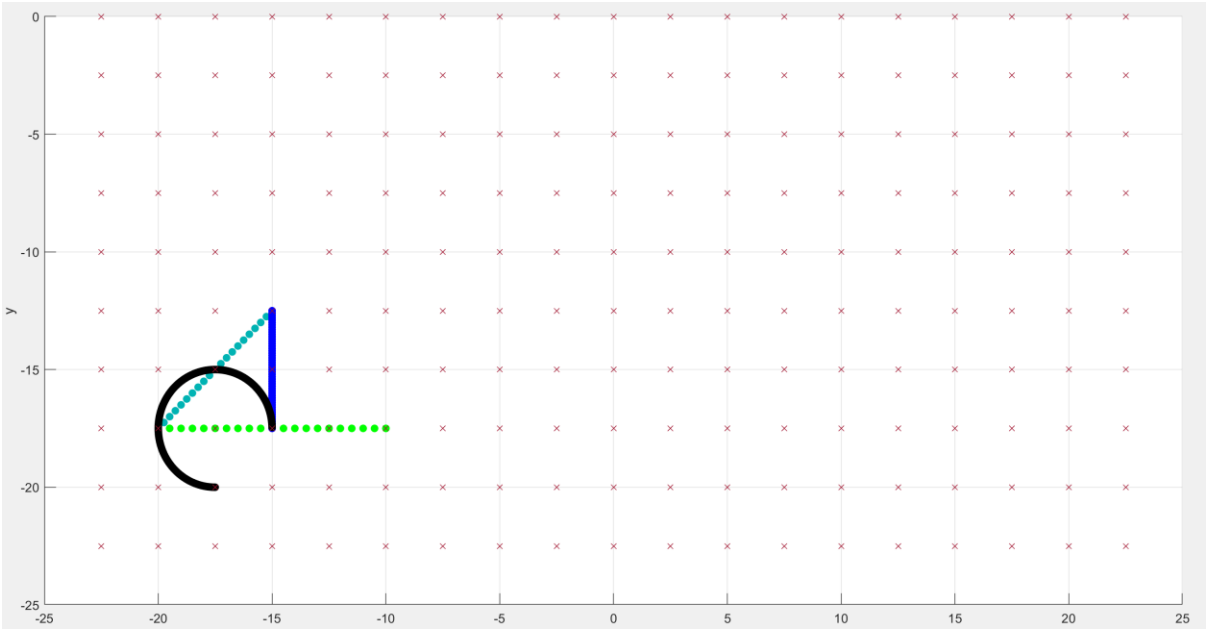

Figure 18: Task 3 Demo Day Simulation