Design and Fabrication of a 4-DOF Industrial Robotic Arm

Mohamed Amr Sabry*, Youssef Khaled Mahran*,
Lina Tamer Ghonim*, Shaheer Sherif Shawky* , Mohammed Saeed* and Mariam Abdelrahman Fathi*
*German University in Cairo (GUC), Egypt
Emails: mohamedamramer@gmail.com ,youssef.mahran@student.guc.edu.eg , mohammed.saeed@student.guc.edu.eg,
lina.ghonim@student.guc.edu.eg, shaheer.aziz@student.guc.edu.eg, mariam.fathi@student.guc.edu.eg,

***Keywords*-Industrial Robotic Applications , robotic manipulators , DOF ,**

I. Introduction

Industrial robotics is one of the fastest-growing fields in modern technology, enabling significant advances in automation, efficiency, and safety. Once limited to repetitive manufacturing tasks, robots are now entering industries that previously had limited automation, such as healthcare, food processing, and agriculture. This review explores recent advancements in industrial robotics, focusing on human-robot collaboration, object recognition, path planning, and optimization, as well as applications in the medical, agricultural, and food industries.

In robotic applications, time data processing allows these machines to achieve higher precision in tasks such as surgical procedures, food sorting, and crop harvesting. These innovations are driving a new era of automation, where robots are not only tools but integral partners in diverse industrial operations.

The scope of this review includes: Recent developments in human-machine interaction (HMI) and its increasing role in industries whether medical , food industry , assembly and material handling applications.

II. Literature Review

A. Human-machine Interaction (HMI)

Recent advancements in Human-Robot Collaboration (HRC) emphasize human-centered automation by integrating human activities with Cyber-Physical Production Systems (CPPS). This approach focuses on adapting industrial manipulators to the physiological characteristics of human operators, using biometric signals like stress, fatigue, and motion tracking. In a collaborative environment, a CPPS enables a robotic arm to assist a human operator in a joint manipulation task, where the robot adapts its task execution speed and operation based on real-time monitoring of the worker's condition. This ensures improved interaction and efficiency in the manufacturing process [1].

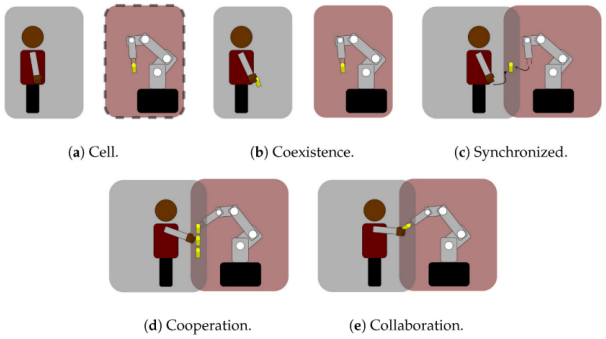

Fig. 1: Human–robot cooperation levels [5]: (a) no collaboration, the robot remains inside a closed work cell; (b) coexistence, removed cells, but separate workspaces; (c) synchronisation, sharing of the workspace, but never at the same time; (d) cooperation, shared task and workspace, no physical interaction; (e) collaboration, operators and robots exchange forces. [2]

B. Medical Applications

Serial robotic manipulators have revolutionized medical applications, particularly in surgery, by enhancing precision, dexterity, and reducing human error in complex procedures. One of the most prominent examples is the da Vinci Surgical System, which has been widely adopted for minimally invasive surgeries such as urology, gynecology, and cardiac operations. Yang et al. in 2024 adressed these systems, which utilize serial linkages to control end-effectors, enabling surgeons to perform tasks with enhanced precision and stability. In recent years, research has focused on improving the haptic feedback, control algorithms, and integration of machine learning (ML) for adaptive robotic behavior during surgery [3]. These manipulators are often structured with a series of revolute joints connected in a chain-like formation, enabling rotation around a fixed axis at each joint. This 7-DOF structure allows for intricate maneuvers such as rotation, pivoting, and precise angular adjustments, which are crucial in surgeries that require high precision in confined spaces, such as laparoscopic procedures [4].

C. Applications in Agriculture

The development of agricultural robots is gaining momentum to address labor shortages and rising food production

demands. Deploying multiple cooperating robots can reduce task duration, accomplish tasks impossible for a single robot, or enhance efficiency. This study [5] presents a cooperation strategy where two heterogeneous robots work together for grape harvesting: one robot (the expert) performs the harvesting, while the other (the helper) carries the harvested grapes. The cooperative methodology ensures safe and effective robot interactions. Field experiments validated the coordinated navigation algorithm and demonstrated the cooperative harvesting method, with recommendations for future improvements. The study also explores using logical, explainable decision-making based on mathematical lattice theory to enhance cooperation between autonomous robots in agricultural applications [5], [6].

In the field of agricultural robotics, end effectors designed for fruit harvesting are considered mechatronic subsystems. Their primary function is to individually detach fruit from stems and deposit them into temporary storage containers [7]. Robotic harvesting is a multifaceted process that integrates video cameras and recognition algorithms to identify ripe fruit, followed by gripping the fruit with a specialized gripper, moving it using a manipulator arm, and placing it in a designated storage area.

Yeshmukhametov et al. showcased in 2019 that harvesting grippers in agricultural applications can generally be categorized into two types. The first type includes precision grip end effectors, which are commonly employed for crops such as strawberries, apples, tomatoes, and sweet peppers (as discussed in [8]. The second type involves compression-based grippers, which feature larger contact areas between the fingers and offer limited or no capacity to transfer movement through the fingers. An example of these grippers is those that function by gripping and vibrating the trunks, as noted in [36].

D. Food Industry

As the global population grows, so does the demand for food, putting pressure on suppliers to improve efficiency and sustainability. Robotics and automation are seen as critical solutions in this sector, although the food industry has been slower to adopt these technologies compared to others. Robotics is being used in food production, packaging, and even cooking. With the development of robotic devices, such as soft grippers, handling delicate food items has become more efficient, reducing the risk of damage and contamination. In recent years, especially due to COVID-19, there has been a notable increase in automation and AI-enabled robotic applications in the restaurant industry, enhancing both food processing and service operations. [9] A study by Sanket et al. in 2022 [10] offers an in-depth evaluation of the use of robotics in the food processing industry, highlighting a relatively novel application area. The review emphasizes the transformative potential of robots in food handling, serving, palletizing, and packaging operations. Key considerations such as robot dynamics, economic efficiency, kinematics, human-robot interaction, hygiene, safety, and maintenance are discussed.

E. Assembly and Material Handling

Segura et al. [11] introduces and analyzes Context-Aware Cloud Robotics (CACR) for advanced material handling. Unlike the One-Time On-Demand Delivery (OTODD) method, CACR features context-aware services and effective load balancing. The paper outlines the system architecture, advantages, challenges, and applications of CACR. It also details key functions for material handling, such as decision-making mechanisms and cloud-enabled simultaneous localization and mapping. A case study demonstrates CACR's energy-efficient and cost-saving capabilities, with simulations confirming its superiority in improving energy efficiency and reducing costs in cognitive industrial IoT applications.

Another Four key structural components, in the following paper [12], were identified: interaction levels, work roles, communication interfaces, and safety control modes. The study found that physical contact-based collaboration, such as screwing assembly of small parts and handling heavyweight objects, is well-suited for the automotive industry. Additionally, certified augmented and virtual reality devices emerged as effective assistive technologies for safety and training needs. The categorization provided helps practitioners select compatible structural components that align with modern manufacturing requirements for highly personalized products.

A specific functionality is proposed by Blatnicky et al. of the designed robotic manipulator which is the possibility of gripping of circular objects. [13]

III. Model Proposal

In our selected industrial application, we propose designing a desktop 4-DOF serial manipulator system specifically for **cooperative pick-and-place tasks**. This setup will involve two manipulators working in tandem to demonstrate multi-robot cooperation by playing an **interactive X-O game**. The game not only makes the demonstration more engaging but also effectively showcases the concept of Multi-Robot Systems (MRS) Cooperation in a clear, tangible manner. By using the serial robotic manipulators to strategically place game pieces, this research illustrates how cooperative robots can work together to achieve common objectives, highlighting both their coordination and adaptability in industrial applications. By exploring the dynamics of robot cooperation in a controlled setting, we aim to lay the groundwork for practical implementations of multi-robot systems in real-world scenarios, driving innovation and improving operational workflows.

IV. List of Components

Part Name	Qty	Price Range (EGP)	Total Price (EGP)	Available / Place
Double Axis High-torque Servo Motors	4	550-760	2750-3800	Hand me downs / Future Electronics
Single Axis Servo Motor	1	-	-	-
Arduino	1	500-1500	500-1500	Personal/ Future Electronics/ RAM Electronics
Power Supply	1	150-300	150-300	-
Jumpers	Several	30-50	120	Maamoun/ Future Electronics
Motor driver/ servo shield (L293D)	1	100	100	RAM Electronics
3D Printed Parts	Several	1.5-3 EGP/gram	500-1500	-
Bearings	Several	10-30	-	-
Screws/ Nuts/ Bolts	Several	30	-	-

TABLE I: Components Summary

V. Proposed GrabCAD Models

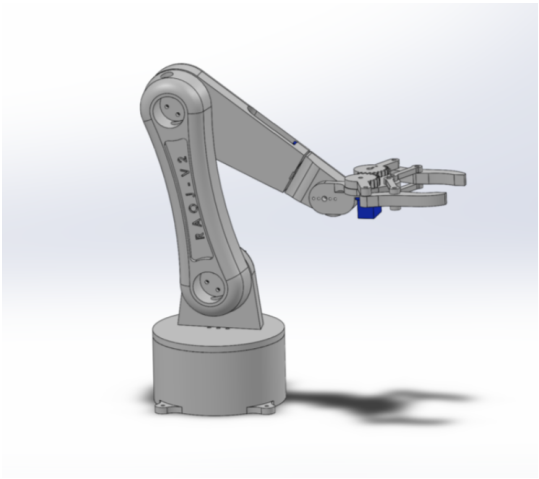

Fig. 2: Side view 1 of 4 DOF Proposed Robotic Arm

This is a gesture-controlled, 3D-printable, 5-degrees-of-freedom, desktop-sized robotic arm. The robotic arm is actuated using three standard servos and two micro servos. The arm is able to mimic human arm movements, which are detected using the Python OpenCV library. The PWM signals of the servos are processed in Python and sent to an Arduino Uno via serial. The links are short which decreases the dominance of the forces which will result in easier analysis and motors selection.

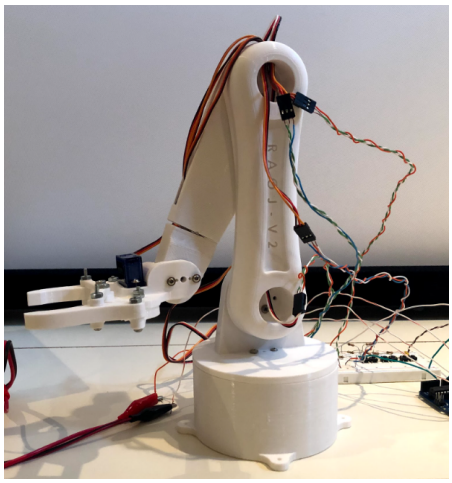

Fig. 3: Side view 2 of 4 DOF Proposed Robotic Arm

VI. Finalized Hardware

The fabrication of the robot involved a detailed and precise process, starting with the 3D printing of the robot's components. 3D printing was chosen due to its flexibility in creating complex geometries with high precision, allowing for customized parts that fit the specific design of the robot. Various materials, including PLA and ABS, were used based on their strength, durability, and weight considerations. For example, the structural components that needed to bear more load were printed in ABS, while lighter, non-load-bearing parts were made from PLA. The 3D printed components were designed to be modular, facilitating easier assembly and maintenance.

Once the parts were printed, the focus shifted to the installation of the motors, which are essential for driving the robot's movement. The motors were selected based on the torque requirements and their compatibility with the robot's design. These motors were mounted into pre-designed motor housings, ensuring that they were securely fixed and aligned correctly. Special attention was given to wiring and ensuring the motors were connected to the power supply and control system, allowing for smooth operation.

After the motors were installed, the robot's links—rigid connections between different parts of the robot—were assembled. The links were carefully attached to the motor shafts, ensuring that each joint allowed for smooth movement while maintaining the robot's stability. The assembly of the links required precise alignment, as even minor misalignments could

affect the robot's functionality and performance. Each link was secured with screws or bolts, and the joints were designed to minimize friction and maximize mobility.

The final steps involved testing and calibrating the assembled robot to ensure everything was functioning as expected. This included adjusting the motors' response to input signals, fine-tuning the movement of the links, and ensuring the overall balance and functionality of the robot. The entire fabrication and assembly process was iterative, with adjustments made to improve performance and ensure the robot met the desired specifications.

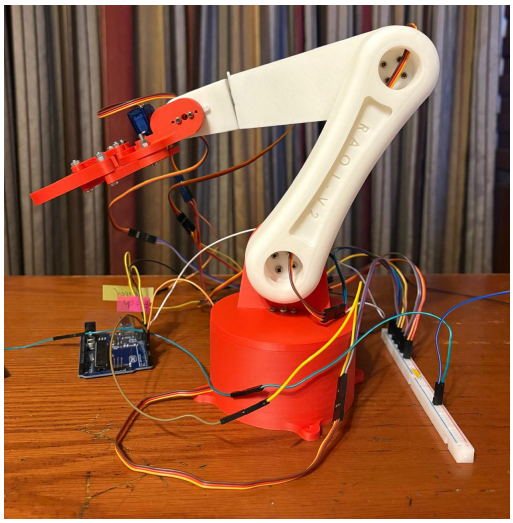

Fig. 4: Front view of the fabricated Robotic Arm

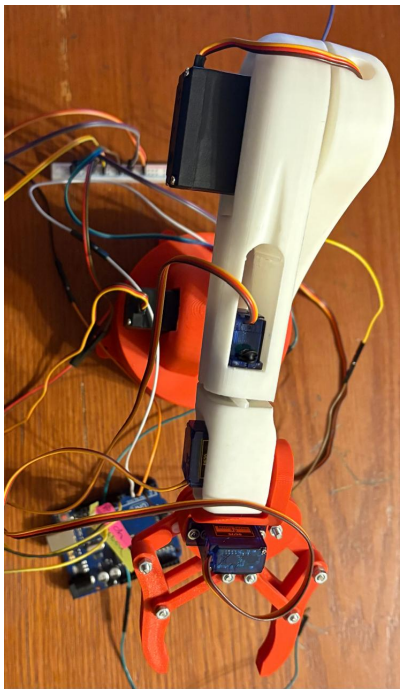

Fig. 5: Top view of the fabricated Robotic Arm

The circuit for the robot was designed using a simulator to provide a clear and easily visualized representation of the electrical components and their connections. This approach allowed us to organize and plan the wiring between the motors, sensors, microcontroller, and power supply efficiently. By designing the circuit in a virtual environment, we ensured that each component was correctly placed and interconnected, offering a straightforward overview of how the system would function once assembled. The figure of the circuit below illustrates the complete design, providing a clear schematic of the components and their connections for better understanding and further development.

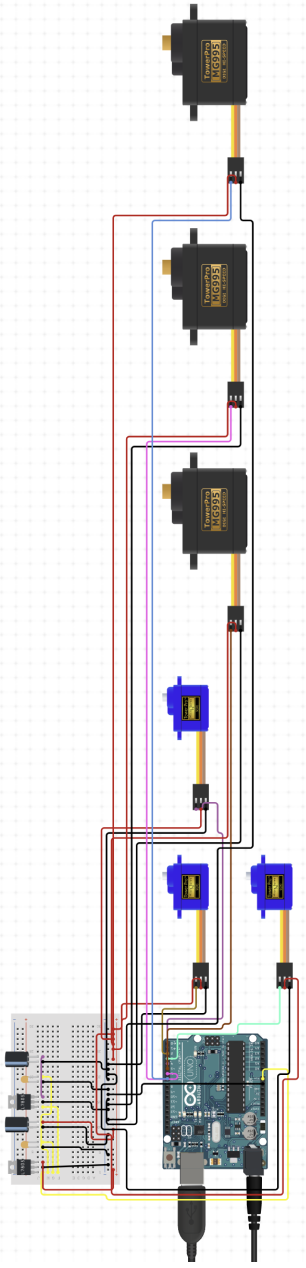

Fig. 6: Motor and Arduino Circuitry

VII. Forward Position Kinematics

The forward position kinematics followed the Denavit–Hartenberg (DH) Convention. Firstly the robot frames were assigned as shown in Figure 7.

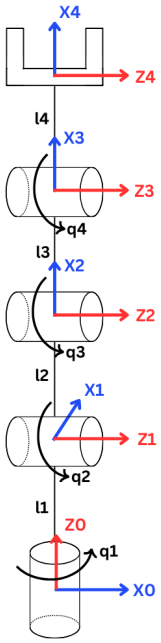

Fig. 7: Frame Assignment for DH Convention

Then the DH parameters table was constructed as shwon in Table II. Where (θ_i) and (d_i) represent the rotation and translation along the (Z_{i-1}) axis respectively while (α_i) and (a_i) represent the rotation and translation along the (X_i) axis respectively.

TABLE II: DH Parameters

$\mathbf{Joint}_i$	θ_i	$\mathbf{d}_i$	α_i	$\mathbf{a}_i$
1	q_1	l_1	0	$\frac{\pi}{2}$
2	q_2	0	l_2	0
3	q_3	0	l_3	0
4	q_4	0	l_4	0

The general form for the Transformation Matrix of frame (i) in respect with frame (i-1) is shown in Equation 1.

$$ {}^{i-1}T_i = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

The four transformation matrices are as follows

$$ {}^{0}T_1 = \begin{bmatrix} Cq_1 & 0 & Sq_1 & 0 \\ Sq_1 & 0 & -Cq1 & 0 \\ 0 & 1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2} $$

$$ {}^{1}T_2 = \begin{bmatrix} Cq_2 & -Sq_2 & 0 & l_2Cq_2 \\ Sq_2 & Cq_2 & 0 & l_2Sq_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3} $$

$$ {}^{2}T_3 = \begin{bmatrix} Cq_3 & -Sq_3 & 0 & l_3Cq_3 \\ Sq_3 & Cq_3 & 0 & l_3Sq_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4} $$

$$ {}^{3}T_4 = \begin{bmatrix} Cq_4 & -Sq_4 & 0 & l_4Cq_4 \\ Sq_4 & Cq_4 & 0 & l_4Sq_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5} $$

The total homogenous transformation matrix is as follows

$$ {}^{0}T_4 = {}^{0}T_1.{}^{1}T_2.{}^{2}T_3.{}^{3}T_4 \tag{6} $$

VIII. Inverse Position Kinematics

A. Forward Kinematics

The forward kinematics equations for the end-effector position (X, Y, Z) are:

$$ \begin{aligned} X = {} & l_2 C_{q_1} C_{q_2+\frac{\pi}{2}} - l_3 S_{q_3} S_{q_2+\frac{\pi}{2}} C_{q_1} + l_3 C_{q_1} C_{q_3} C_{q_2+\frac{\pi}{2}} \\ & + l_4 \left(-S_{q_3} S_{q_2+\frac{\pi}{2}} C_{q_1} + C_{q_1} C_{q_3} C_{q_2+\frac{\pi}{2}} \right) C_{q_4} \\ & + l_4 \left(-S_{q_3} C_{q_1} C_{q_2+\frac{\pi}{2}} - S_{q_2+\frac{\pi}{2}} C_{q_1} C_{q_3} \right) S_{q_4} \end{aligned} $$

$$ \begin{aligned} Y = {} & l_2 S_{q_1} C_{q_2+\frac{\pi}{2}} - l_3 S_{q_1} S_{q_3} S_{q_2+\frac{\pi}{2}} + l_3 S_{q_1} C_{q_3} C_{q_2+\frac{\pi}{2}} \\ & + l_4 \left(-S_{q_1} S_{q_3} S_{q_2+\frac{\pi}{2}} + S_{q_1} C_{q_3} C_{q_2+\frac{\pi}{2}} \right) C_{q_4} \\ & + l_4 \left(-S_{q_1} S_{q_3} C_{q_2+\frac{\pi}{2}} - S_{q_1} S_{q_2+\frac{\pi}{2}} C_{q_3} \right) S_{q_4} \end{aligned} $$

$$ \begin{aligned} Z = {} & l_1 + l_2 S_{q_2+\frac{\pi}{2}} + l_3 S_{q_3} C_{q_2+\frac{\pi}{2}} + l_3 S_{q_2+\frac{\pi}{2}} C_{q_3} \\ & + l_4 \left(-S_{q_3} S_{q_2+\frac{\pi}{2}} + C_{q_3} C_{q_2+\frac{\pi}{2}} \right) S_{q_4} \\ & + l_4 \left(S_{q_3} C_{q_2+\frac{\pi}{2}} + S_{q_2+\frac{\pi}{2}} C_{q_3} \right) C_{q_4} \end{aligned} $$

B. Jacobian Matrix

The Jacobian matrix $\mathbf{J}$ is derived as the partial derivatives of (X, Y, Z) with respect to the joint angles $[q_1, q_2, q_3, q_4]$:

$$ \mathbf{J} = \begin{bmatrix} \frac{\partial X}{\partial q_1} & \frac{\partial X}{\partial q_2} & \frac{\partial X}{\partial q_3} & \frac{\partial X}{\partial q_4} \\ \frac{\partial Y}{\partial q_1} & \frac{\partial Y}{\partial q_2} & \frac{\partial Y}{\partial q_3} & \frac{\partial Y}{\partial q_4} \\ \frac{\partial Z}{\partial q_1} & \frac{\partial Z}{\partial q_2} & \frac{\partial Z}{\partial q_3} & \frac{\partial Z}{\partial q_4} \end{bmatrix} $$

After substituting the trigonometric identities $\cos(q_i) = C_{q_i}$ and $\sin(q_i) = S_{q_i}$, the Jacobian matrix becomes:

$$ \mathbf{J} = \begin{bmatrix} J_{11} & J_{12} & J_{13} & J_{14} \\ J_{21} & J_{22} & J_{23} & J_{24} \\ J_{31} & J_{32} & J_{33} & J_{34} \end{bmatrix} $$

where each $J_{ij} = \frac{\partial (X,Y,Z)}{\partial q_j}$ is evaluated numerically.

C. Newton-Raphson Iterative Update

The joint angles are updated iteratively using the Newton-Raphson method:

$$\Delta \mathbf{q} = \mathbf{J}^{+}(\mathbf{X}_d - \mathbf{X}_{\text{fk}})$$

where:

- $\mathbf{J}^{+}$ is the pseudoinverse of the Jacobian matrix
- $\mathbf{X}_d = [X_d, Y_d, Z_d]^T$ is the desired position
- $\mathbf{X}_{\text{fk}} = [X, Y, Z]^T$ is the current position from forward kinematics

The joint angles are updated as:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \Delta \mathbf{q}$$

Convergence is checked using:

$$\|\mathbf{X}_d - \mathbf{X}_{\text{fk}}\| < \epsilon$$

where ϵ is the tolerance for the solution.

IX. Simscape Simulations

This section contains the simscape model of the robot, the 3D viewer of the robot using Simscape's Mechanics Explorer, and the results of two cases on input angles where the robot position from Simscape is compared to that of the DH convention and the inverse position kinematics. First, the Simscape model of the robot can be seen in Figure 8. Additionally, the 3D view of the robot on Mechanics Explorer can be observed in Figure 9. The angle q_1 is responsible for the base link's rotation, the angle q_2 is responsible for link 1's rotation, the angle q_3 is responsible for link 2's rotation and finally, the angle q_4 is responsible for the gripper's rotation.

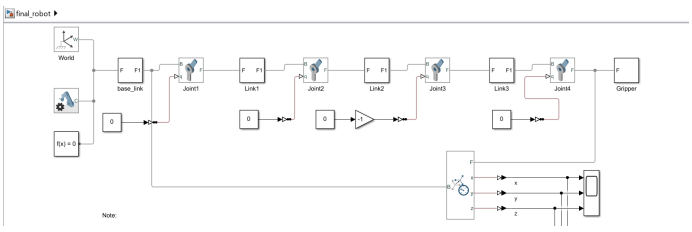

Fig. 8: Simscape Model of the Robot

Fig. 9: 3D view of the robot on Simscape

A. Validation of Forward Position Kinematics (DH-convention)

Two cases are used to simulate the robot and ensure correctness between the DH convention equations and the simscape end effector position. The first case is the equilibrium position where the robot is standing vertically upward and the joint angles are all equal to 0. The values obtained from the MATLAB code and Simscape simulations are seen in Figures 10 and 11. It is apparent that they are approximately equal within a small tolerance due to dimensions not taken into account in the DH convention such as the small depth values between the links. Finally, the robot's position in 3D is readily seen in Figure 12.

```
X =
   1.6778e-14

Y =
    0.2182

Z =
  317.5499
```

Fig. 10: Case 1 MATLAB Results

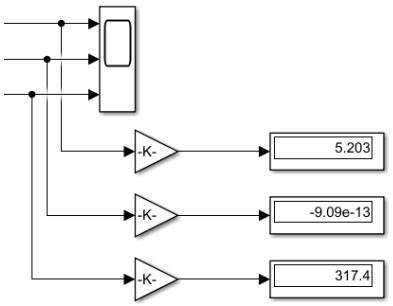

Fig. 11: Case 1 Simscape Results

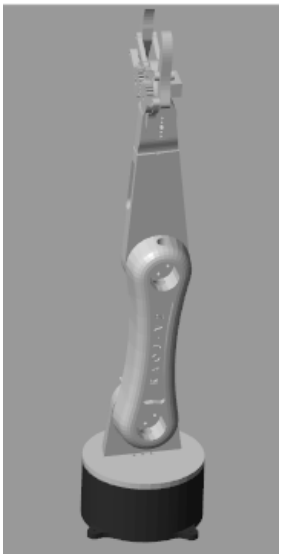

Fig. 12: Case 1 Robot Position

The second case is a more general case where:

$$q_1 = 0.5\ rad \tag{7}$$
$$q_2 = 0.4\ rad \tag{8}$$
$$q_3 = 0.5\ rad \tag{9}$$
$$q_4 = 0\ rad \tag{10}$$

The values obtained from the MATLAB code and Simscape simulations are seen in Figures 13 and 14. Since the joint q_1 is actuated, the robot has a position on the Y-axis as if q_1 was not actuated, then the joints q_2 and q_3 would simply move the robot in the Z-X plane. Note that q_1 is a rotation around a vertical axis and q_2 and q_3 are rotations around axes out of the page. Finally, q_4 is the rotation of the gripper also around an axis out of the page. It is apparent that they are approximately equal within a small tolerance due to dimensions not taken into account in the DH convention such as the small depth values between the links. Finally, the robot's position in 3D is readily seen in Figure 15.

```
X =
 -140.0417

Y =
  -76.3126

Z =
  255.7942
```

Fig. 13: Case 2 MATLAB Results

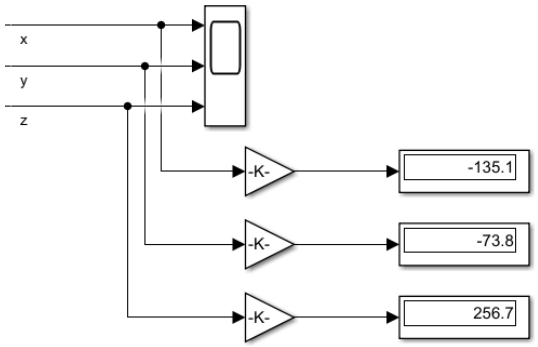

Fig. 14: Case 2 Simscape Results

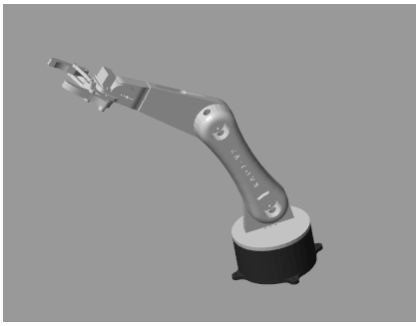

Fig. 15: Case 2 Robot Position

B. Validation of Inverse Position Kinematics

Two cases are utilized to validate the employed inverse position kinematics algorithm based on the Newton-Raphson method. First, a certain desired position which consists of three components: (X_d, Y_d, Z_d) is chosen. Second, the inverse position kinematics function is run to obtain the required joint angles which achieve this desired position. With the joint angles in hand, they are given as input to both the forward position kinematics function and the Simscape joints to validate that they yield the same position as the aforementioned desired position (X_d, Y_d, Z_d).

1) Case 1: The first case consists of a position near the equilibrium position of the robot where the desired position is defined as $(-0.1266, -0.0535, 0.2711)$. This can be seen in Figure 16. The required joint angles to achieve this position were determined using inverse position kinematics MATLAB function which is called:

`"inverse_position_nr "`

. After obtaining these values, they are inputted into the forward position kinematics function to ensure its output is equal to the desired position which is called:

`"forward_kinematics"`

. The results of the aforementioned functions can be seen in Figure 17. Finally, the required joint angles obtained were inputted into the Simscape model of the robot and the resulting

position of the end-effector and the final robot position can be observed in Figure 18 and Figure 19 respectively.

```
% Desired position
Xd = -0.126644546635844;
Yd = -0.0535444555078038;
Zd = 0.271105807530105;
```

Fig. 16: Case 1 Desired Position

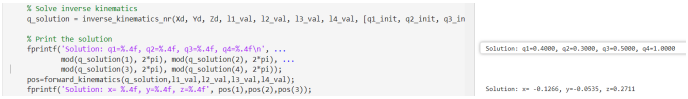

Fig. 17: Case 1 Validation of Inverse Position Kinematics Algorithm with Forward Position Kinematics Algorithm

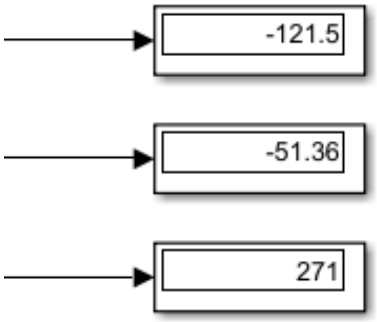

Fig. 18: Case 1 Validation of Inverse Position Kinematics with Simscape Model

Fig. 19: Case 1 Robot Position

The results of case 1 show that the forward position kinematics algorithm yields the desired position when inputted with the joint angles obtained from the inverse position kinematics algorithm which is observed in Figure 17. Furthermore, Figure 18 shows the position of the robot with the joint angles acquired from inverse position kinematics. It is apparent that this position very closely matches the desired position which is further validation of our inverse position kinematics algorithm. These minor inaccuracies can be attributed to the depths of the links which are not taken into account in the DH convention. Finally, the resultant position of the robot in Simscape can be seen in Figure 19.

2) Case 2: The second case consists of a position more similar to the ones observed in pick and place applications where the desired position is defined as $(-0.125, -0.125, 0.08)$. This can be seen in Figure 20. The required joint angles to achieve this position were determined using inverse position kinematics MATLAB function which is called:

`"inverse_position_nr "`

. After obtaining these values, they are inputted into the forward position kinematics function to ensure its output is equal to the desired position which is called:

`"forward_kinematics"`

. The results of the aforementioned functions can be seen in Figure 21. Finally, the required joint angles obtained were inputted into the Simscape model of the robot and the resulting position of the end-effector and the final robot position can be observed in Figure 22 and Figure 23 respectively.

```
%2. Case 2
Xd = -0.125;
Yd = -0.125;
Zd = 0.08;
```

Fig. 20: Case 2 Desired Position

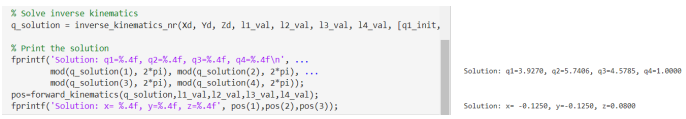

Fig. 21: Case 2 Validation of Inverse Position Kinematics with Forward Position Kinematics

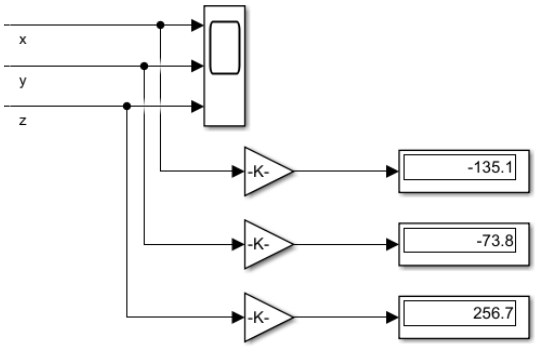

Fig. 22: Case 2 Validation of Inverse Position Kinematics with Simscape

Fig. 23: Case 2 Robot Position

The results of case 2 show that the forward position kinematics algorithm yields the desired position when inputted with the joint angles obtained from the inverse position kinematics algorithm which is observed in Figure 21. Furthermore, Figure 22 shows the position of the robot with the joint angles acquired from inverse position kinematics. It is apparent that this position very closely matches the desired position which is further validation of our inverse position kinematics algorithm. These minor inaccuracies can be attributed to the depths of the links which are not taken into account in the DH convention. Finally, the resultant position of the robot in Simscape can be seen in Figure 23.

C. Constraints of the System

The constraints of the system are represented as below. Only the base link's angle q_1 can rotate through $360°$ while the others are restricted to $90°$ to avoid extreme positions.

$$q_1 \in 0, 2\pi \; rad \tag{11}$$

$$q_2 \in 0, \frac{\pi}{2} \; rad \tag{12}$$

$$q_3 \in 0, \frac{\pi}{2} \; rad \tag{13}$$

$$q_4 \in 0, \frac{\pi}{2} \; rad \tag{14}$$

X. Coppeliasim Simulations

This section contains the Coppelisim simulation of the robot and the results of two cases on input angles where the robot position from Coppeliasim is compared to that of the DH convention. First, the Simscape model of the robot can be seen in Figure 24.

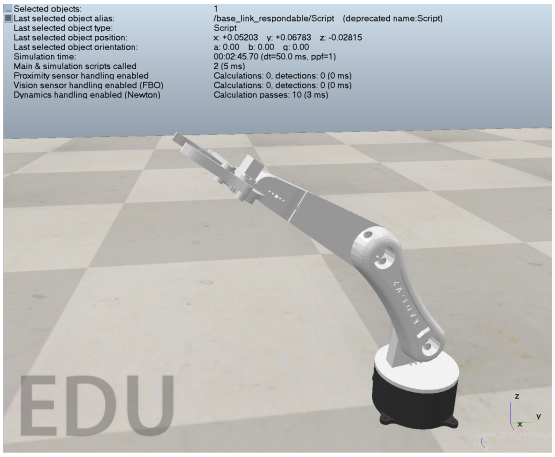

Fig. 24: Coppelisim Robot Simulation

Two cases are used to simulate the robot and ensure correctness between the DH convention equations and the Coppelisim end effector position. The first case is the equilibrium position where the robot is standing vertically upward and the joint angles are all equal to 0. The values obtained from the python code and Coppeliasim simulations are seen in Figures 25 and 26. It is apparent that they are approximately equal within a small tolerance due to dimensions not taken into account in the DH convention such as the small depth values between the links.

```
PS C:\Users\youss\Desktop\Optimization> & C:/Users/youss/AppData/L
Gripper Position:  [1.67776611e-17 1.67776611e-17 3.18000000e-01]
```

Fig. 25: Case 1 Python Results

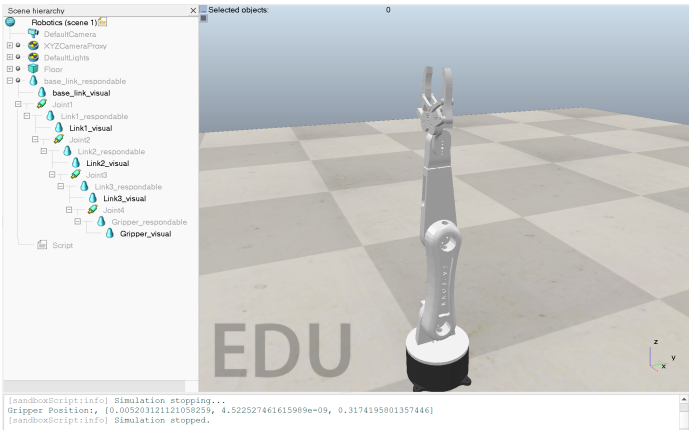

Fig. 26: Case 1 Coppeliasim Results

The second case is a more general case where:

$$q_1 = 0.5\ rad \tag{15}$$
$$q_2 = 0.4\ rad \tag{16}$$
$$q_3 = 0.5\ rad \tag{17}$$
$$q_4 = 0\ rad \tag{18}$$

The values obtained from the MATLAB code and Simscape simulations are seen in Figures 27 and 28. It is apparent that they are approximately equal within a small tolerance due to dimensions not taken into account in the DH convention such as the small depth values between the links.

```
PS C:\Users\youss\Desktop\Optimization> & C:/Users/youss/
Gripper Position:  [-0.13996071 -0.07646088  0.25624427]
```

Fig. 27: Case 2 Python Results

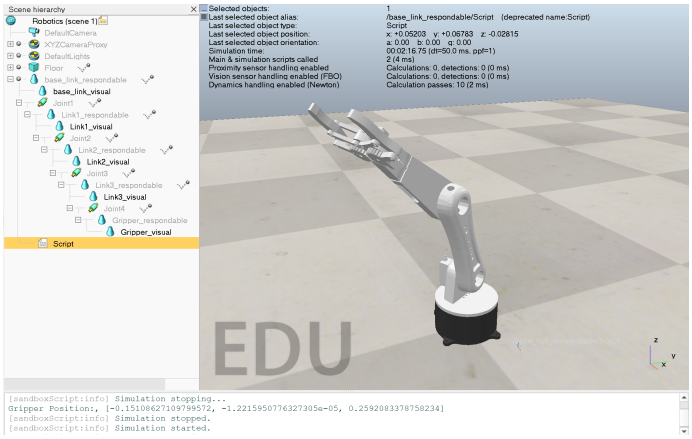

Fig. 28: Case 2 Coppeliasim Results

The constraints of the system are represented as below. Only the base link's angle q_1 can rotate through $360°$ while the others are restricted to $90°$ to avoid extreme positions.

$$q_1 \in 0, 2\pi\ rad \tag{19}$$
$$q_2 \in 0, \frac{\pi}{2}\ rad \tag{20}$$
$$q_3 \in 0, \frac{\pi}{2}\ rad \tag{21}$$
$$q_4 \in 0, \frac{\pi}{2}\ rad \tag{22}$$

XI. Hardware Assembly

This section shows the hardware assembly process.

XII. Trajectory Generation and Visualization

This section introduces the design of the task-space and joint-space trajectories that are consistent with the targeted pick-and-place operation. The specific trajectories were chosen to emulate the robot starting in a home position and going down to the position of the object to carry it, which is essentially the required application scenario. Both task and joint space trajectories were designed with the same initial and final positions however, they ultimately yield different trajectories due to the nature of their derivation. This will be discussed in detail in the following sections.

A. Task-Space Trajectories

The task-space trajectory designed is a linear one which adheres to the following form:

$$x(t) = x_0 + \alpha_x t \tag{23}$$
$$y(t) = y_0 + \alpha_y t \tag{24}$$
$$z(t) = z_0 + \alpha_z t \tag{25}$$
$$\tag{26}$$

Where $\alpha_x, \alpha_y, \alpha_z$ are the slopes of the trajectories in the x, y, z axes respectively. The aforementioned constants can be determined in terms of the initial and final positions (x_0, y_0, z_0) and (x_f, y_f, z_f) as well as the duration of the trajectory t_d.

$$\alpha_x = \frac{x_f - x_0}{t_d} \tag{27}$$
$$\alpha_y = \frac{y_f - y_0}{t_d} \tag{28}$$
$$\alpha_z = \frac{z_f - z_0}{t_d} \tag{29}$$

Furthermore, the chosen initial and final positions in meters and the time duration are as follows:

$$p_0 = (-0.1266, -0.05354, 0.2711)\ m \tag{30}$$
$$p_f = (-0.17, -0.1, 0.01)\ m \tag{31}$$
$$t_d = 10\ s \tag{32}$$

It is apparent that the initial position is one where the robot is near its equilibrium position (large z coordinate) and the final position is one where the robot is near the ground ($z \approx 0$). Finally, for these specific start and end points the values of the slopes are as follows:

$$\alpha_x = -4.34 * 10^{-3} \frac{m}{s} \tag{33}$$
$$\alpha_y = -4.646 * 10^{-3} \frac{m}{s} \tag{34}$$
$$\alpha_z = -0.02611 \frac{m}{s} \tag{35}$$

Figure 29 visualizes the 3D linear trajectory undertaken by the robot to go from its initial to its final position.

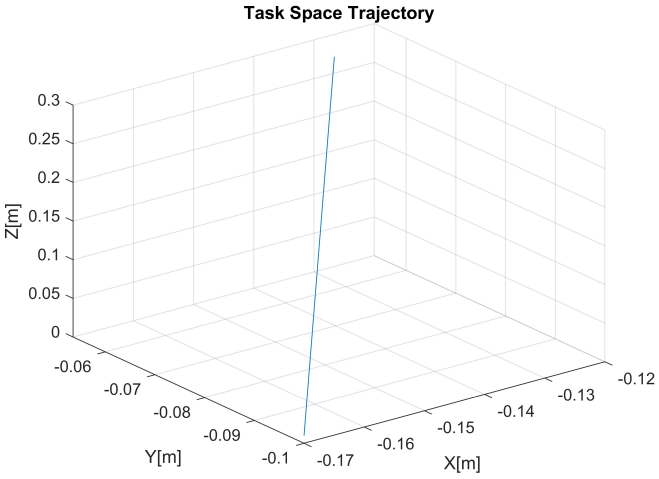

Fig. 29: Task-Space Trajectory Visualization

To implement this on Simscape or CoppeliaSim, the desired position at time t is calculated using the task-space trajectory equations. These values are then inputted to the inverse position kinematics algorithm to be converted to joint angles, which are then used to simulate the robot's motion.

B. Joint Space Trajectories

The joint space trajectory equations designed are cubic polynomials which are designed assuming the robot initial and final velocities are equal to zero.

$$q(t) = C_0 + C_1 t + C_2 t^2 + C_3 t^3 \tag{36}$$

The joint-space trajectories were designed considering the same initial and final positions of the robot as well as the same time duration.

$$p_0 = (-0.1266, -0.05354, 0.2711)\ m \tag{37}$$
$$p_f = (-0.17, -0.1, 0.01)\ m \tag{38}$$
$$t_d = 10\ s \tag{39}$$

To get the initial and final joint angles $[q_1, q_2, q_3, q_4]$, the inverse position kinematics algorithm was used which yielded:

$$q(0) = [0.4001, -0.4318, 1.3774, 0.3837] rad \tag{40}$$
$$q(10) = [0.5317, 0.3852, 1.8571, 0.4544] rad \tag{41}$$

Additionally, the initial and final joint velocities were set to zero and hence

$$\dot{q}(0) = [0, 0, 0, 0] \frac{rad}{s} \tag{42}$$
$$\dot{q}(10) = [0, 0, 0, 0] \frac{rad}{s} \tag{43}$$

Thus, for every joint angle equation, there are 4 unknowns (C_0, C_1, C_2, C_3) and 4 equations $q(0), q(10), \dot{q}(0), \dot{q}(10)$ and thus the coefficients of the third degree polynomial can hence be determined exactly for each joint angle. The corresponding equations of each of the joint angles are:

$$q_1(t) = 0.4001 + 3.948 * 10^{-3} t^2 - 2.632 * 10^{-4} t^3 \tag{44}$$
$$q_2(t) = -0.4318 + 0.02451 t^2 - 1.634 * 10^{-3} t^3 \tag{45}$$
$$q_3(t) = 1.3774 + 0.0144 t^2 - 9.6 * 10^{-4} t^3 \tag{46}$$
$$q_4(t) = 0.3837 + 2.121 * 10^{-3} t^2 - 1.414 * 10^{-4} t^3 \tag{47}$$

Figure 30 visualizes the joint space trajectory in the task-space. As apparent from the figure, the trajectory is curved compared to the task-space trajectory which was designed to be linear, however, the joint space equations were cubic polynomials. Nevertheless, the initial and final positions are the same, as expected.

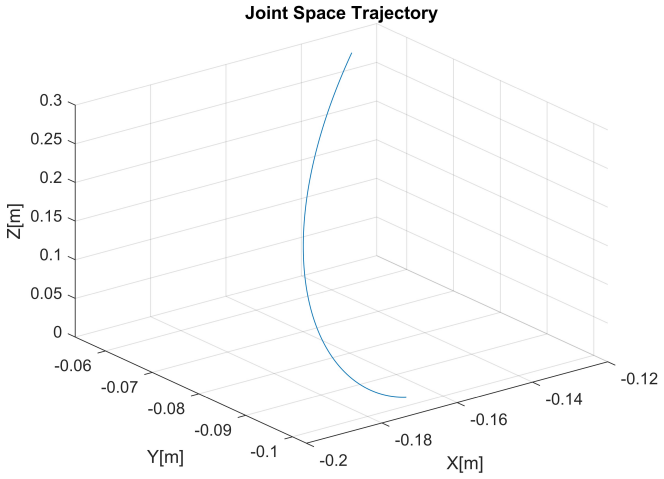

Fig. 30: Joint-Space Trajectory Visualization

C. Validation of Trajectories on Simscape

In Simscape, two Simulink subsystems were designed. The first is a task space trajectory algorithm which takes the initial and final position of the robot, the time duration, and initial guess for the joint angles as inputs. The output of this block is the required joint angles which are calculated using an inverse position kinematics algorithm. The second Simulink subsystem implements the joint space trajectory equations which are calculated externally and simply inputted into the Simulink subsystem. Figure 31 exhibits the aforementioned Simulink subsystems.

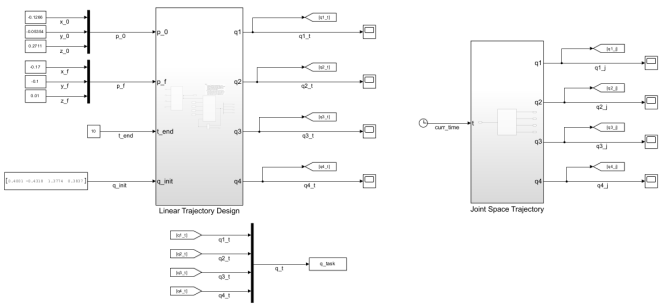

Fig. 31: Simulink Subsystems for Task and Joint Space Trajectories

The trajectories were simulated in Simulink, and the video results can be seen in Team 7's github. The initial position of the robot can be seen in Figure 32 and the final position of the robot can be observed in Figure 33. Finally, the final position of the robot from Simscape's sensor can be seen in Figure 34 where it is apparent that the robot achieves the desired position with high accuracy.

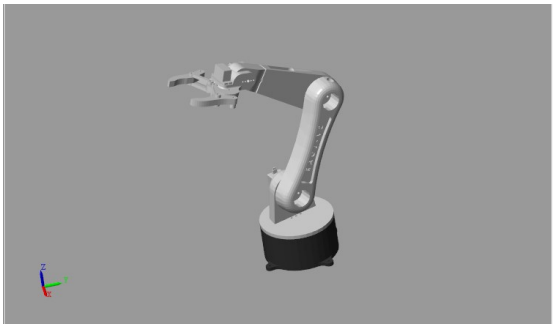

Fig. 32: Initial Robot Position in Task and Joint Space Trajectories

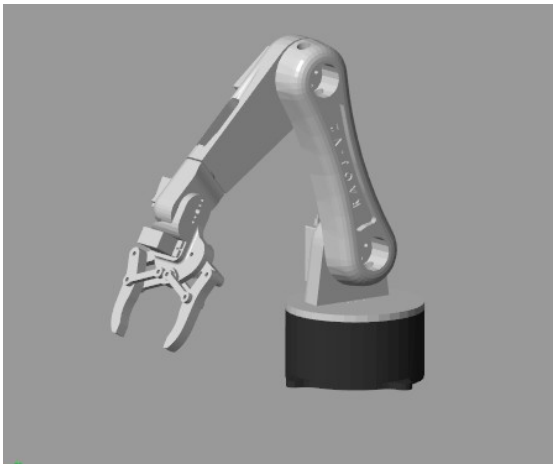

Fig. 33: Final Robot Position in Task and Joint Space Trajectories

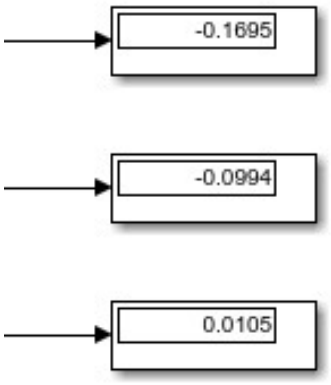

Fig. 34: Value of Final Robot Position in Task and Joint Space Trajectories

D. Validation of Trajectories on Coppeliasim

In coppeliasim, two python codes were designed. The first is a task space trajectory code which takes an array of joint angles generated from matlab script and sends it to the robot. The output of this block is the required joint angles which are calculated using an inverse position kinematics algorithm. The second is a joint space trajectory code where it computes the joint values online using the joint space equations and sends it to the robot. The trajectories were simulated in Coppelia, and the video results can be seen in Team 7's github. The initial position of the robot can be seen in Figure 35 and the final position of the robot can be observed in Figure 36.

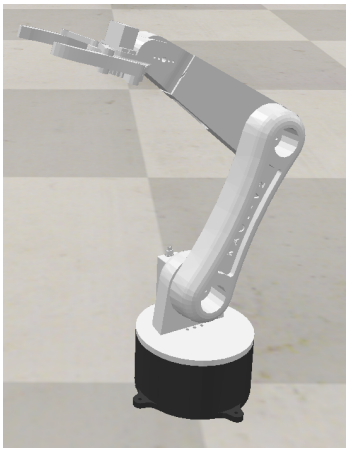

Fig. 35: Initial Robot Position in Task and Joint Space Trajectories in Coppeliasim

Fig. 36: Final Robot Position in Task and Joint Space Trajectories in Coppeliasim

As shown Coppelia sim achieved consistent results with simscape.

REFERENCES

[1] L. Antão, R. Pinto, J. Reis, G. Gonçalves, and F. L. Pereira, "Cooperative human-machine interaction in industrial environments," in *2018 13th APCA International Conference on Automatic Control and Soft Computing (CONTROLO)*, pp. 430–435, IEEE, 2018.

[2] A. Olivares-Alarcos, S. Foix, and G. Alenyà, "On inferring intentions in shared tasks for industrial collaborative robots," *Electronics*, vol. 8, no. 11, p. 1306, 2019.

[3] J. C.-Y. Ngu, C. C.-W. Lin, C. J.-Y. Sia, and N.-Z. Teo, "A narrative review of the medtronic hugo ras and technical comparison with the intuitive da vinci robotic surgical system," *Journal of Robotic Surgery*, vol. 18, no. 1, p. 99, 2024.

[4] I. F. Zidane, Y. Khattab, S. Rezeka, and M. El-Habrouk, "Robotics in laparoscopic surgery-a review," *Robotica*, vol. 41, no. 1, pp. 126–173, 2023.

[5] C. Lytridis, C. Bazinas, I. Kalathas, G. Siavalas, C. Tsakmakis, T. Spirantis, E. Badeka, T. Pachidis, and V. G. Kaburlasos, "Cooperative grape harvesting using heterogeneous autonomous robots," *Robotics*, vol. 12, no. 6, p. 147, 2023.

[6] C. Lytridis, V. G. Kaburlasos, T. Pachidis, M. Manios, E. Vrochidou, T. Kalampokas, and S. Chatzistamatis, "An overview of cooperative robotics in agriculture," *Agronomy*, vol. 11, no. 9, p. 1818, 2021.

[7] C. Morar, I.-A. Doroftei, I. Doroftei, and M. Hagan, "Robotic applications on agricultural industry. a review," in *IOP Conference Series: Materials Science and Engineering*, vol. 997, p. 012081, IOP Publishing, 2020.

[8] A. Yeshmukhametov, K. Koganezawa, Z. Buribayev, Y. Amirgaliyev, and Y. Yamamoto, "Development of continuum robot arm and gripper for harvesting cherry tomatoes," *Preprints*, 2019. Available online: https://www.preprints.org/manuscript/201910.0162.v1.

[9] A. Dzedzickis, J. Subačiūtė-Žemaitienė, E. Šutinys, U. Samukaitė-Bubnienė, and V. Bučinskas, "Advanced applications of industrial robotics: New trends and possibilities," *Applied Sciences*, vol. 12, no. 1, 2022.

[10] B. Kokane Sanket, S. Kalamnurikar Shalaka, and B. Khose Suyog, "Robotics in food processing industries: A review," *The Pharma Innovation Journal*, vol. 11, no. 7S, pp. 948–953, 2022.

[11] J. Wan, S. Tang, Q. Hua, D. Li, C. Liu, and J. Lloret, "Context-aware cloud robotics for material handling in cognitive industrial internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2272–2281, 2017.

[12] P. Segura, O. Lobato-Calleros, A. Ramírez-Serrano, and I. Soria, "Human-robot collaborative systems: Structural components for current manufacturing applications," *Advances in Industrial and Manufacturing Engineering*, vol. 3, p. 100060, 2021.

[13] M. Blatnickỳ, J. Dižo, J. Gerlici, M. Sága, T. Lack, and E. Kuba, "Design of a robotic manipulator for handling products of automotive industry," *International Journal of Advanced Robotic Systems*, vol. 17, no. 1, p. 1729881420906290, 2020.