AE4324 Assignment

Eunkyung Cho
Miłosz Pluciński
Miantao Zhao

1 Mobility and Workspaces

This report starts with an overview of the robot considered for this assignment, with detailed specifications outlined in subsection 1.1. It then covers the robot's mobility in subsection 1.2, followed by a discussion of the robot's workspace in subsection 1.3.

1.1 Robot Specification

The robot analyzed in this assignment is a generic four-degree-of-freedom (DoF) robotic arm, with its visualization shown in Figure 1.

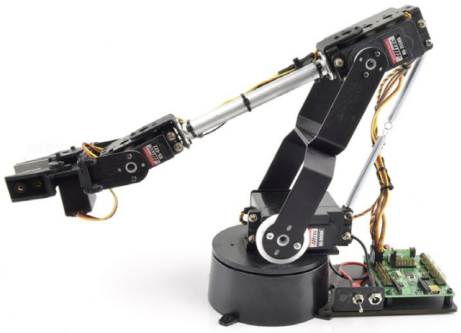
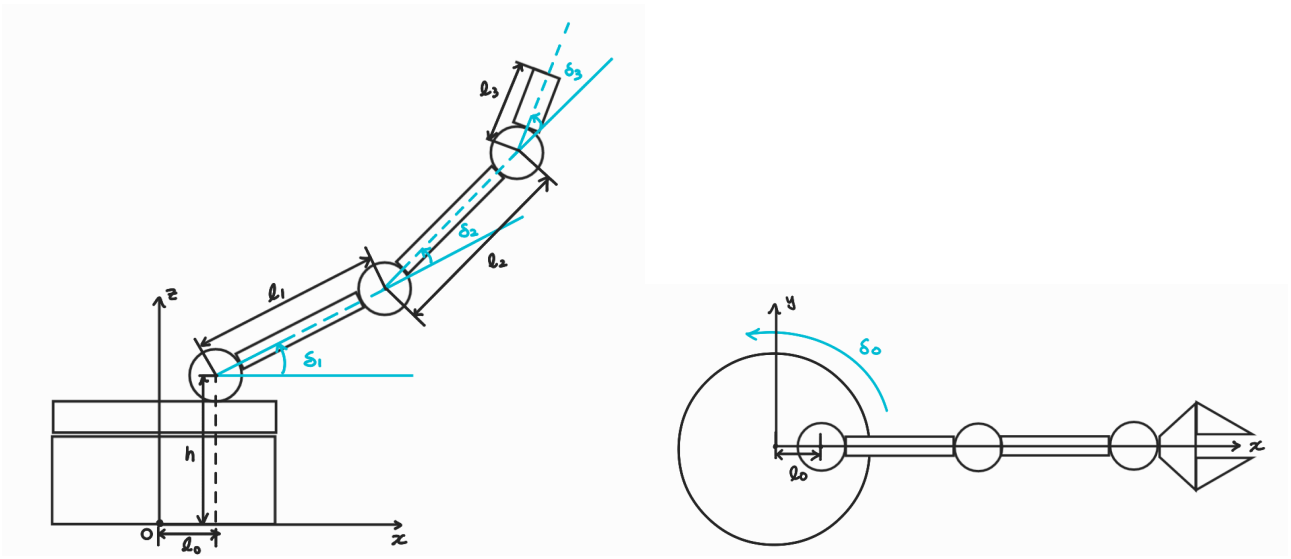


Figure 1: A generic 4-DoF robotic manipulator

For the later analysis, the link lengths and angles are defined as visualised in Figure 2a and Figure 2b



(a) Side view of the robotic arm

(b) Top view of the robotic arm

The length of each link is determined by measuring the distance between the coordinate points of its corresponding joints. Table 1 summarizes the link lengths.

Table 1: Link Length

Link	Length [m]	Symbol
0	0.015	l_0
1	0.094	l_1
2	0.107	l_2
3	0.055	l_3

1.2 Mobility

Mobility refers to the number of independent parameters needed to define the configuration of a mechanism. The general mobility of a mechanism is given by the Grübler-Kutzbach equation [?]:

$$M = d(n - g - 1) + \sum_{i=0}^{g} f_i \tag{1}$$

where:

- d is the mobility factor (6 for spatial mechanisms, 3 for planar mechanisms),
- n is the total number of links, including the ground link,
- g is the total number of joints,
- f_i is the degree of freedom of joint i.

Since the robot operates in a spatial mechanism, $d = 6$. Given that there are six links ($n = 6$) and five joints ($g = 5$), with each joint providing one degree of freedom, the total sum of joint freedom is:

$$\sum_{i=0}^{g} f_i = 5. \tag{2}$$

Substituting these values into Equation 1 gives the general mobility of 5.

1.3 Workspace

The workspace of a robot is the set of positions its end-effector can reach. To analyze this workspace, the forward kinematics of the robot arm was first derived. Forward kinematics describes the end-effector's position and orientation as a function of the joint angles. In three-dimensional space, the position and orientation are given by:

$$x = (l_1c_1 + l_2c_{12} + l_3c_{123}) \cdot c_0 \tag{3}$$
$$y = (l_1c_1 + l_2c_{12} + l_3c_{123}) \cdot s_0 \tag{4}$$
$$z = l_1s_1 + l_2s_{12} + l_3s_{123} \tag{5}$$
$$\phi = 0 \tag{6}$$
$$\theta = \delta_1 + \delta_2 + \delta_3 \tag{7}$$
$$\psi = \delta_0 \tag{8}$$

Where x, y, z represent the three-dimensional position of the robot arm, ϕ is the roll angle of the end effector, θ denotes its to pitch angle and ψ denotes its yaw or heading angle. The terms $s_{ij...k}$ and $c_{ij...k}$ are defined as:

$$s_{ij...k} = \sin(\delta_i + \delta_j + ... + \delta_k) \tag{9}$$
$$c_{ij...k} = \cos(\delta_i + \delta_j + ... + \delta_k) \tag{10}$$

The joint angle limits, which define the physical constraints of the robot's movement, are summarized in Table 2.

Table 2: Joint Angle Limits

Joint	**Min Joint Angle** [°]	**Max Joint Angle** [°]
0	-95	95
1	-14	150
2	-167	0
3	-95	100

To visualize the workspace, a 3D simulation was generated using Python, considering two scenarios:

- **Unconstrained workspace**: No joint limits were imposed.
- **Constrained workspace**: The actual joint constraints were applied.

This analysis provides insights into the robot's reachable space under practical conditions. The visualization of the unconstrained case is shown in Figure 3, Figure 4 and Figure 5.

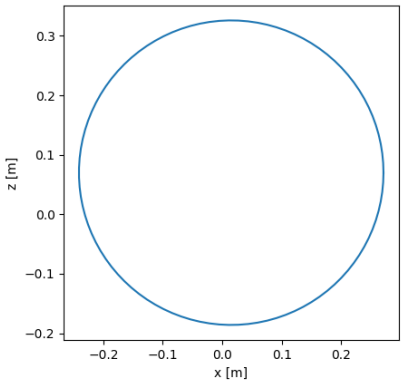


Figure 3: Side view of robot workspace with $\delta_0 = 0$ [rad] without joint constraints

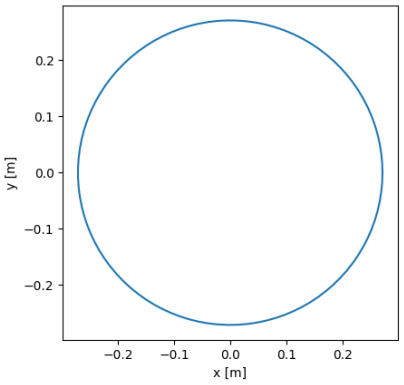


Figure 4: Top down view of robot workspace without joint constraints

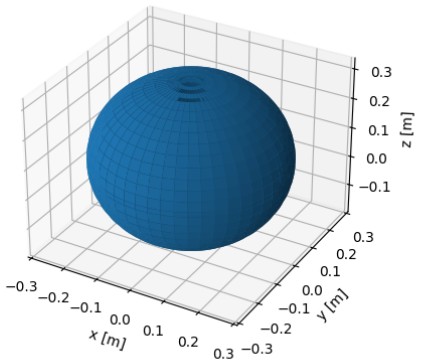


Figure 5: 3D plot of robot workspace without joint constraints

The constrained workspace is depicted in Figures 6, 7, and 8.

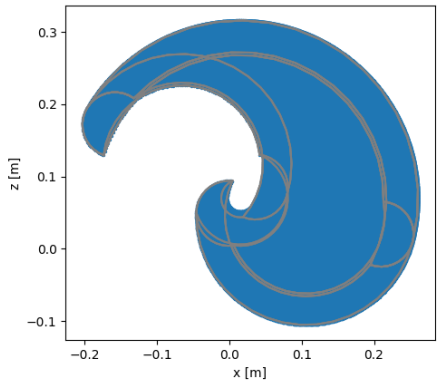


Figure 6: Side view of robot workspace with $\delta_0 = 0$ [rad] with joint constraints

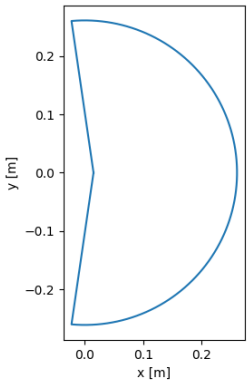


Figure 7: Top down view of robot workspace with joint constraints

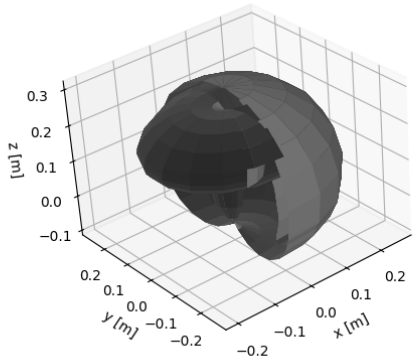


Figure 8: 3D plot of robot workspace with joint constraints

2 Inverse Kinematics

Inverse kinematics (IK) refers to the process of determining the joint angles required for a robotic arm to reach a specified end-effector position. A Python-based IK function was developed and tested to validate this process. To assess the accuracy and feasibility of the implementation, a simple validation was carried out to determine whether selected end-effector positions were within the robot's reachable workspace. The positions evaluated are listed below.

i. Position 1: (0.1, 0.1, 0.1)

ii. Position 2: (0.2, 0.1, 0.3)

iii. Position 3: (0.0, 0.0, 0.3)

iv. Position 4: (0.0, 0.0, 0.07)

Among these, Positions 1 and 3 were found to be reachable, whereas Positions 2 and 4 were outside the robot's operational workspace. This limitation is attributed to the physical constraints of the robotic arm, which restrict its range of motion and prevent the end-effector from attaining certain locations.

For the feasible positions, the IK function was used to control the robot in both simulation and real-world environments. Given that a 4-DoF robot can yield multiple valid IK solutions for a single end-effector pose, both elbow-up and elbow-down configurations were computed, visualized, and evaluated.

2.0.1 Position I – [0.1, 0.1, 0.1]

In simulation, both the elbow-down and elbow-up solutions were successfully computed and executed, as illustrated in Figure 9 and Figure 18.

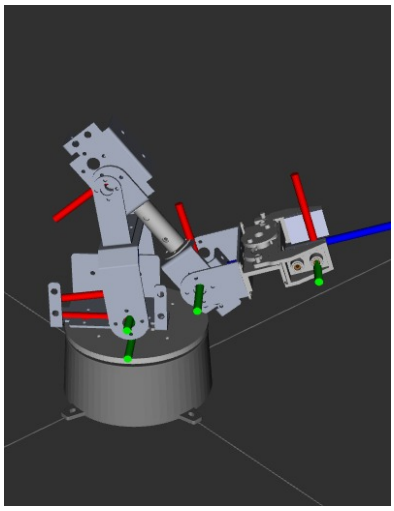


Figure 9: Position I – elbow-down (simulation)

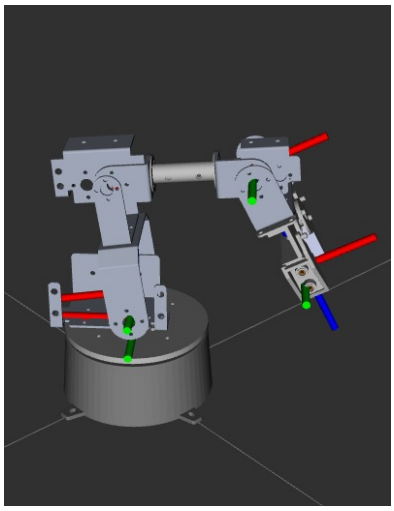


Figure 10: Position I – elbow-up (simulation)

On the physical robot, both configurations were also executed successfully:

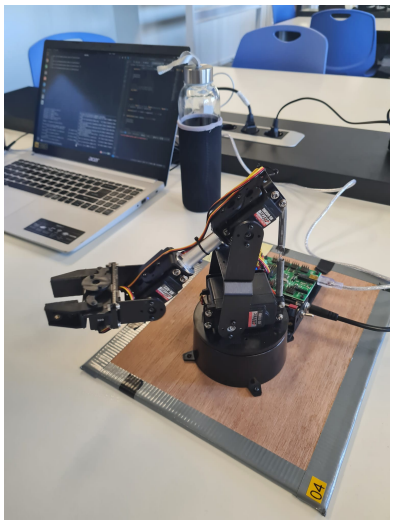


Figure 11: Position I – elbow-down (physical space)

Figure 12: Position I – elbow-up (physical space)

2.0.2 Position III – [0.0, 0.0, 0.3]

For Position III, both elbow-down and elbow-up configurations were successfully computed and executed, as shown in Figure 13 and Figure 14.

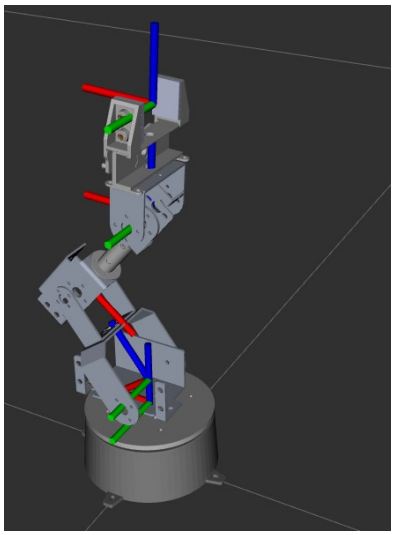


Figure 13: Position III – elbow-down (simulation)

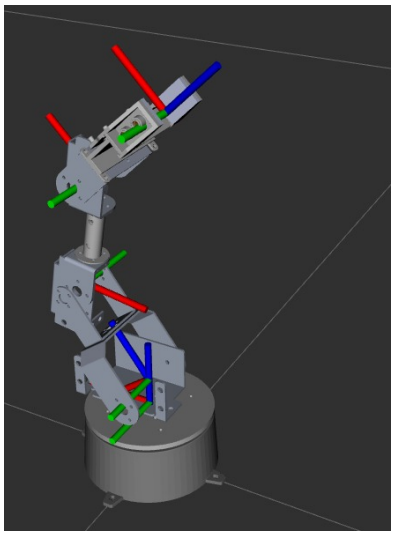


Figure 14: Position III – elbow-up (simulation)

The real-world executions of these configurations are shown in Figure 15 and Figure 16:

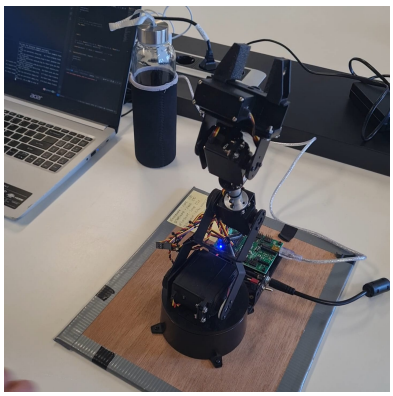


Figure 15: Position III – elbow-down (physical space)

Figure 16: Position III – elbow-up (physical space)

Following the successful validation of individual end-effector positions, the inverse kinematics function was extended to a dynamic scenario, enabling the end-effector to follow a continuous trajectory. A time-parameterized path was defined in three-dimensional space, and corresponding joint configurations were computed for each point along this path. These joint trajectories were then executed on the physical robot, allowing the end-effector to trace the desired shape.

In this demonstration, the target shape was the Prometheus flame from the TU Delft logo. A bicycle light was attached to the end-effector (gripper), and a long-exposure photograph was taken during execution. The result, along with the simulated trajectory, is shown below.

Figure 17: Long exposure image of TU Delft logo

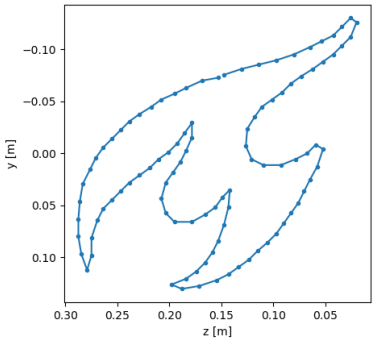


Figure 18: Trajectory of flame in simulation environment

3 Jacobian

The Jacobian matrix describes the relationship between the joint velocities and the end-effector velocities. Mathematically, this relationship is expressed as:

$$\dot{\vec{x}} = J\dot{\vec{q}} \tag{11}$$

where $\dot{\vec{x}}$ denotes the time derivative of the end-effector position and orientation, and $\dot{\vec{q}}$ is the vector of joint velocities.

The partial derivative of an end-effector defines each element of the Jacobian matrix coordinate with respect to a joint variable:

$$J_{ij} = \frac{\partial x_i}{\partial q_j} \tag{12}$$

where the vectors $\vec{x}$ and $\vec{q}$ are given as:

$$\vec{x} = \begin{pmatrix} x \\ y \\ z \\ \theta \\ \phi \\ \psi \end{pmatrix}, \quad \vec{q} = \begin{pmatrix} \delta_0 \\ \delta_1 \\ \delta_2 \\ \delta_3 \end{pmatrix} \tag{13}$$

Here x, y, z denote the position of the end effector whereas ϕ, θ, ψ denote orientation of the end effector in the roll, pitch and yaw directions respectively. The end-effector position and orientation components are functions of the joint angles. Thus their derivatives can be found to construct the Jacobian matrix for the robot, which is shwon in subsection 1.1:

$$J = \begin{bmatrix} -(l_1c_1 + l_2c_{12} + l_3c_{123}) \cdot s_0 & (-l_1s_1 - l_2s_{12} - l_3s_{123}) \cdot c_0 & (-l_2s_{12} - l_3s_{123}) \cdot c_0 & (-l_3s_{123}) \cdot c_0 \\ (l_1c_1 + l_2c_{12} + l_3c_{123}) \cdot c_0 & (-l_1s_1 - l_2s_{12} - l_3s_{123}) \cdot s_0 & (-l_2s_{12} - l_3s_{123}) \cdot s_0 & (-l_3s_{123}) \cdot s_0 \\ 0 & l_1c_1 + l_2c_{12} + l_3c_{123} & l_2c_{12} + l_3c_{123} & l_3c_{123} \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{14}$$

As $\phi = 0$ and x, y and ψ are coupled, a simplified square version of J can be constructed by eliminating the bottom row, which corresponds to ϕ. This leads to the square Jacobian matrix $J_\square$ and the reduced end effector space vector $\vec{x}_r$, which contains the position x, y, z and the pitch.

$$J_{\square} = \begin{bmatrix} -(l_1c_1 + l_2c_{12} + l_3c_{123}) \cdot s_0 & (-l_1s_1 - l_2s_{12} - l_3s_{123}) \cdot c_0 & (-l_2s_{12} - l_3s_{123}) \cdot c_0 & (-l_3s_{123}) \cdot c_0 \\ (l_1c_1 + l_2c_{12} + l_3c_{123}) \cdot c_0 & (-l_1s_1 - l_2s_{12} - l_3s_{123}) \cdot s_0 & (-l_2s_{12} - l_3s_{123}) \cdot s_0 & (-l_3s_{123}) \cdot s_0 \\ 0 & l_1c_1 + l_2c_{12} + l_3c_{123} & l_2c_{12} + l_3c_{123} & l_3c_{123} \\ 0 & 1 & 1 & 1 \end{bmatrix}, \quad (15)$$

$$\dot{\vec{x}}_r = J_{\square}\dot{\vec{q}}, \qquad \vec{x}_r = \begin{pmatrix} x \\ y \\ z \\ \theta \end{pmatrix} \quad (16)$$

The benefit of using the square Jacobian $J_{\square}$ is that it is invertible, provided it is not singular. This allows the vector of joint velocities to be computed from the motion of the end effector as follows:

$$\dot{\vec{q}} = J_{\square}^{-1}\dot{\vec{x}_r} \quad (17)$$

In the case of infeasible solutions, the Jacobian becomes undefined and non-computable because no joint configuration $\mathbf{q}$ exists such that the forward kinematics $\mathbf{x} = f(\mathbf{q})$ satisfies the specified task-space constraint. As a result, the partial derivative in Equation 12 cannot be evaluated. This has direct implications where without a valid Jacobian, one cannot compute differential relationships between joint and end-effector velocities. Consequently, it is not possible to generate joint velocity commands that achieve the desired end-effector motion, rendering the task physically unachievable for the robot.

In practice, the Jacobian was used to compute the joint velocities required to track a constant end-effector velocity trajectory. This trajectory was executed on the real robot, and a video of the execution is included in the submission package.

4 Pick and place

This section explores the practical capabilities and limitations of a robotic arm through a series of manipulation tasks that progressively increase in complexity and precision. The experiments are designed to evaluate core functionalities such as pick-and-place operations, gentle object handling, and contact-based tasks like surface wiping.

4.1 Task 4.1

One of the most common tasks in robotics is the Pick and Place operation, where a robot picks up an object from one location and places it in another. In the setup, the robot was programmed to pick up a rigid object from location A, place it at location B, return to its home position, and then repeat the task in reverse. A demonstration video of this Pick and Place task is included in the submission files.

A key challenge observed during this task was the absence of position sensing or feedback control. This meant the object had to be placed manually and precisely in a fixed position for the robot to successfully grasp it. Any minor misalignment could result in a failed pickup, reducing the system's robustness and adaptability. Additionally, the lack of velocity control led to jerky and inconsistent movements, which could be problematic in real-world applications requiring smooth, precise motion.

Recommended Modifications:

- Integrate computer vision or proximity sensors for real-time object detection and localization
- Implement closed-loop position and velocity control to enable smoother, more accurate motion
- Add motion planning algorithms that can dynamically adjust trajectories based on object location

4.2 Task 4.2

For Task 4.2, a similar motion sequence as in subsection 4.1 was executed, but with an added requirement of delicacy: picking up a soft object (a berry) and placing it into a cup. Since no real berries were available, a sponge was used as a substitute.

The primary challenge here was the lack of force or tactile feedback in the gripper. This made it impossible to control the gripping force, which would be essential if a real berry were used — excessive pressure could easily bruise or crush it. The same issues from subsection 4.1 regarding lack of sensing and velocity control also applied.

Recommended Modifications:

- Equip the gripper with force or pressure sensors to allow adaptive gripping of delicate objects.
- Incorporate compliant or soft robotic grippers to reduce the risk of damaging fragile items.

4.3 Task 4.3

In this task, the robot was tasked with cleaning up a water spill using a sponge. The sponge was picked up by the gripper and used to wipe the surface until it was visibly clean. A major issue observed was the misalignment between commanded and actual joint angles, particularly at large deflections. Specifically, the motor angle δ_1 showed a deviation of several degrees when reaching far from the robot's base, impacting accuracy and repeatability. Moreover, due to the absence of force feedback at the joints, the robot was unable to maintain constant pressure on the surface, which is an essential requirement for effective wiping.

Recommended Modifications:

- Calibrate the joint control system to reduce positional errors, especially for extended reaches.
- Add joint torque sensors or a force-torque sensor at the end-effector to enable constant contact force.
- Integrate impedance or admittance control strategies for better surface interaction.

5 Appendix

Implementation acessible through github at https://github.com/Taouuw/robotics2