Door and Handle Motion Execution via Robotic Arm (PART B)

Dimitrios Karatis 10775, *Electrical and Computer Engineering, AUTH*

***Abstract*—This project extends the work from Part A by implementing the simulated door and handle motion using a 6-DOF industrial robotic arm (UR10). The task involves generating joint velocity commands to reproduce the predefined trajectory of the door handle and door opening sequence within a 5-second interval. The robot maintains a rigid grasp on the handle, preserving the relative end-effector pose throughout the motion. The motion planning ensures continuity in joint positions and velocities, utilizing the Jacobian-based inverse differential kinematics of the UR10. MATLAB implementation with Peter Corke's Robotics Toolbox enables real-time simulation, including 3D visualization of the robot, the end-effector's absolute and relative trajectories, and joint position and velocity profiles.**

***Index Terms*—robotics, differential kinematics, Jacobian, UR10, joint velocity control, trajectory tracking, handle manipulation, rigid grasp, MATLAB simulation**

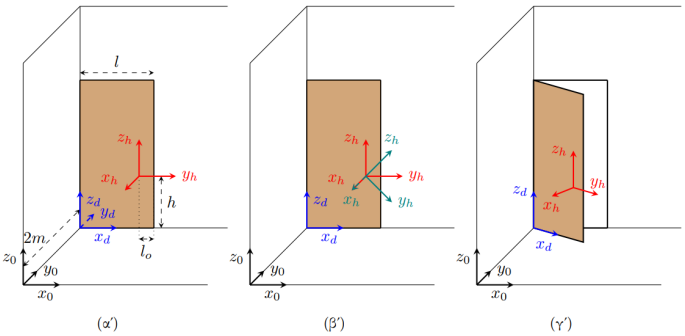

Fig. 1: The 3 Poses from part A

I. Introduction

This part of the project focuses on controlling a 6-DOF robotic arm (UR10) to perform the door opening sequence simulated in Part A. The robot starts from an initial joint configuration:

$$\mathbf{q}_0 = [-1.7752, -1.1823, 0.9674, 0.2149, 1.3664, 1.5708]^\top \text{ rad}$$

while maintaining a fixed end-effector pose relative to the door handle frame $\{H\}$. The UR10's base frame $\{B\}$ is located at position $\mathbf{p}_B = [1, 1, 0]^\top$ with identity orientation.

The objective is to generate joint velocity commands $\dot{\mathbf{q}}_r \in \mathbb{R}^6$ such that the end-effector follows the reference trajectory for the handle, ensuring a rigid grasp and smooth motion over $T = 5$ seconds with sampling period $t_s = 0.01$ seconds. The control approach is based on differential kinematics using the arm's Jacobian, ensuring that joint trajectories are continuous and physically feasible.

II. Theoretical Background

A. Differential Kinematics

Differential kinematics describes how small changes in joint space produce corresponding changes in the task space (i.e., the position and orientation of the end-effector). The key relation is:

$$\dot{\mathbf{x}} = J(\mathbf{q})\dot{\mathbf{q}}$$

where $\dot{\mathbf{x}} \in \mathbb{R}^6$ is the spatial velocity (twist) of the end-effector, $\dot{\mathbf{q}} \in \mathbb{R}^n$ is the vector of joint velocities, and $J(\mathbf{q}) \in \mathbb{R}^{6\times n}$ is the Jacobian matrix evaluated at the joint configuration $\mathbf{q}$. The top three elements of $\dot{\mathbf{x}}$ correspond to the linear velocity of the end-effector, while the bottom three describe its angular velocity (in the base frame or end-effector frame, depending on convention).

The Jacobian provides an instantaneous linear mapping between joint velocities and end-effector twist. As such, it plays a critical role in motion control and inverse kinematic analysis. For the UR10 robot used in this project, the Jacobian is computed via:

```
J = ur10.jacobe(q);
```

which returns the geometric Jacobian expressed in the base frame.

B. Inverse Velocity Kinematics

In many robotic tasks, the desired velocity of the end-effector is known, and the goal is to compute the corresponding joint velocities that realize that motion. This inverse problem can be solved using the pseudoinverse of the Jacobian:

$$\dot{\mathbf{q}} = J^\dagger(\mathbf{q})\dot{\mathbf{x}}$$

where $J^\dagger$ denotes the Moore-Penrose pseudoinverse of J. This method provides a least-squares solution when the system is either redundant (more joints than task dimensions) or underdetermined (fewer joints than task dimensions).

For square and full-rank Jacobians, $J^\dagger = J^{-1}$. In the case of the 6-DOF UR10 robot, which has a square Jacobian, the pseudoinverse often suffices.

The pseudoinverse guarantees the minimum-norm joint velocity vector that achieves the desired end-effector velocity. However, it is sensitive to singularities and ill-conditioned Jacobians.

This document was prepared by Dimitrios Karatis as part of the second assignment for the Robotics course at Aristotle University of Thessaloniki.

C. Rigid Grasp Constraint

A fundamental assumption in this project is that the robot's end-effector is rigidly attached to the door handle throughout the motion. This implies that the relative pose between the end-effector frame $\{E\}$ and the handle frame $\{H\}$ remains fixed for the entire duration:

$$g_{EH}(t) = \text{constant}$$

Mathematically, this means the transformation from the end-effector to the handle, denoted $g_{EH} = g_{OE}^{-1} g_{OH}$, does not vary with time. As such, the motion of the end-effector is entirely dictated by the trajectory of the handle frame:

$$g_{OE}(t) = g_{OH}(t) \cdot g_{HE}$$

This constraint simplifies the control objective: by ensuring the end-effector tracks the handle trajectory while maintaining the fixed grasp transformation g_{HE}, we guarantee that the robot performs the intended door-opening sequence.

This rigid constraint also simplifies the evaluation of task performance — any deviation from constancy in $g_{EH}(t)$ indicates an error in tracking or loss of this rigidity.

III. Implementation in MATLAB

A. Initialization

The simulation begins by initializing the UR10 robot model using the Robotics Toolbox:

```
ur10 = ur10robot();
```

The robot operates over a time horizon of $T = 5$ seconds with a sampling interval of $\Delta t = 0.01$ seconds. This leads to $N = 501$ discrete time steps:

$$\text{time vector: } t \in [0, 0.01, \ldots, 5.0]$$

The robot starts from a predefined joint configuration:

$$\mathbf{q}_0 = [-1.7752, -1.1823, 0.9674, 0.2149, 1.3664, 1.5708]^T \text{ rad}$$

which aligns the end-effector with the initial handle pose from Part A. Two matrices, `q_traj` and `qd_traj`, are initialized to store joint positions and joint velocities respectively over all time steps.

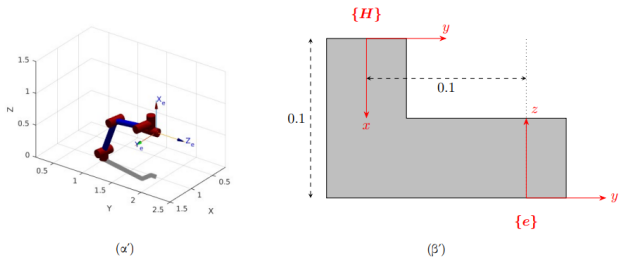

Fig. 2: UR10 Robot with Base Frame $\{B\}$ and End-Effector Frame $\{E\}$

B. Loading Trajectories

The homogeneous transformation trajectories computed in Part A are loaded from:

```
partA_oh_traj.mat
partA_od_traj.mat
```

These trajectories represent the pose of the handle $\{H\}$ and the door $\{D\}$ with respect to the inertial frame $\{0\}$ over time. Each entry in the 3D matrix `g_oh_traj(:,:,k)` contains the 4×4 transformation matrix for the handle at time t_k.

C. Rigid Grasp Definition

The robot maintains a rigid grasp on the handle throughout the door's opening sequence. This is modeled by a constant transformation g_{HE} from the handle frame $\{H\}$ to the end-effector frame $\{E\}$:

$$g_{HE} = \begin{bmatrix} 0 & 0 & -1 & 0.1 \\ 0 & 1 & 0 & 0.1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This implies the end-effector is offset from the handle by 10 cm along both the x_H and y_H axes, and rotated such that the axes align appropriately. The inverse transform $g_{EH} = g_{HE}^{-1}$ is computed in case it is needed later.

D. Control Loop

At each timestep k, the following control strategy is applied:

1) The desired end-effector pose $g_{OE}^{\text{des}}(t_k)$ is computed as:
$$g_{OE}^{\text{des}}(t_k) = g_{OH}(t_k) \cdot g_{HE}$$
2) The current pose of the end-effector is calculated via forward kinematics:
$$g_{OE}^{\text{curr}} = \texttt{ur10.fkine}(q)$$
3) The pose error $\Delta\xi$ between the current and desired poses is computed using:
$$\Delta\xi = \texttt{tr2delta}(g_{OE}^{\text{curr}}, g_{OE}^{\text{des}})$$
which returns a 6D vector encoding both translational and rotational differences.
4) The geometric Jacobian of the robot in the base frame is obtained by:
$$J = \texttt{ur10.jacobe}(q)$$
5) The joint velocities are computed using the pseudoinverse of the Jacobian:
$$\dot{\mathbf{q}} = J^{-1} \frac{\Delta\xi}{\Delta t}$$
6) Joint positions are updated using first-order Euler integration:
$$\mathbf{q}_{k+1} = \mathbf{q}_k + \dot{\mathbf{q}}_k \cdot \Delta t$$
7) Both $\mathbf{q}_k$ and $\dot{\mathbf{q}}_k$ are stored for analysis and visualization.

E. Simulation and Animation

Once the trajectory is computed, the robot's motion is animated in a 3D figure. The `ur10.animate()` function displays the UR10's configuration at regular intervals (every 4 time steps). Coordinate frames for the door, handle, and end-effector are also depicted using `trplot` to visually track their orientation and position.

To enhance realism, two custom helper functions are used:

- `plotHandle()` – draws a cylindrical model of the door handle aligned with the frame $\{H\}$.
- `plotDoor()` – creates a rectangular patch simulating the door surface attached to frame $\{D\}$.

These visualization elements help verify that the robot correctly manipulates the handle while keeping the relative pose fixed.

Additionally, the base frame and axis lines are also plotted for reference, and all visual elements update dynamically to reflect the robot's state throughout the simulation.

IV. Results

A. Joint positions $\mathbf{q}(t)$

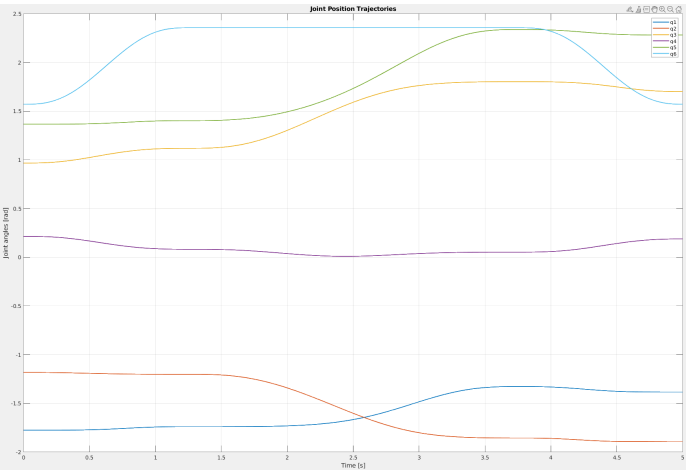

Fig. 3: Joint position trajectories of the UR10 robot over time

The figure above shows the evolution of the six joint angles q_1 through q_6 of the UR10 robot throughout the 5-second task execution. Each trajectory is smooth and continuous, with no abrupt discontinuities, which confirms the effectiveness of the inverse differential kinematics and Euler integration scheme used in the control loop.

The variation in each joint angle reflects the coordinated effort of the robot to follow the desired end-effector trajectory while respecting the rigid grasp constraint. Notably, joint q_2 and q_5 exhibit the most significant changes, indicating that these joints contribute heavily to the handle's spatial redirection during the door-opening motion. In contrast, joint q_4 remains relatively constant, suggesting a stabilizing role or minimal contribution to orientation adjustment in this particular task.

This plot also provides insight into the redundancy and workspace capabilities of the UR10. The variation among joints indicates the robot's ability to exploit multiple joint configurations to achieve the same end-effector pose, which is crucial for avoiding singularities or joint limits in more complex tasks.

B. Joint Velocities $\dot{\mathbf{q}}(t)$

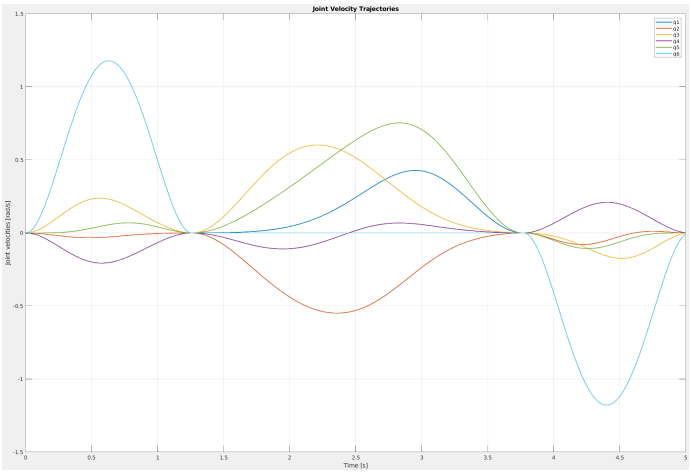

Fig. 4: Joint velocity trajectories of the UR10 robot over time

Figure 4 illustrates the time evolution of the joint velocities $\dot{q}_1$ through $\dot{q}_6$ for the UR10 robot during the execution of the door opening task. Each curve corresponds to one of the six joints, and all trajectories exhibit smooth and continuous profiles, with no sharp discontinuities or jumps.

The joint velocities are zero at both the beginning ($t = 0$) and end ($t = 5$) of the motion, which aligns with the boundary conditions imposed in Part A — namely, that the end-effector motion should begin and end at rest.

The plot shows distinct peaks and valleys in different joints at different times, reflecting the complex coordination needed among joints to track the end-effector trajectory. For instance, joints q_2 and q_6 reach the highest velocity magnitudes, suggesting that these joints provide the majority of rotational and translational compensation during handle manipulation. Conversely, joints like q_4 exhibit lower overall velocity magnitudes, indicating a more supportive role in maintaining orientation.

C. End-Effector Position Trajectory

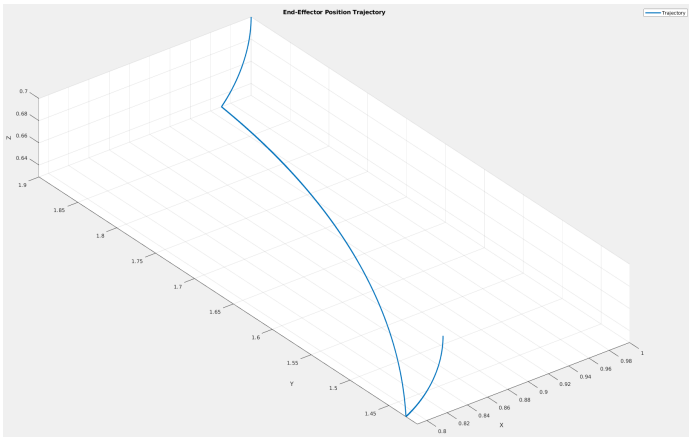

Fig. 5: 3D position trajectory of the UR10 end-effector in the base frame

The figure above presents the spatial path of the robot's end-effector in the base frame throughout the 5-second simulation. This 3D trajectory captures the full motion of the end-effector as it follows the desired path imposed by the handle's motion, based on the kinematic planning from Part A.

The curve shows a smooth, continuous trajectory without abrupt direction changes, which validates the correctness of the inverse differential kinematics controller and the interpolation used in the original handle trajectory. The overall motion begins with a downward and inward curve, corresponding to the initial rotation of the handle, followed by a lateral arc that aligns with the door's opening. The final segment returns to a more vertical displacement as the handle resets. The smooth curvature also demonstrates that the motion planning ensures zero initial and final velocities, avoiding jerky or unrealistic transitions.

This 3D trajectory confirms that the end-effector closely tracks the reference poses with high spatial fidelity, ensuring the integrity of the rigid grasp throughout the task.

D. End-Effector Orientation

Figure 6 shows the evolution of the end-effector's orientation throughout the 5-second door-opening task, expressed in unit quaternion form as $\mathbf{q} = [q_0, q_1, q_2, q_3]^T$, where q_0 is the scalar component and $[q_1, q_2, q_3]^T$ is the vector part.

Using unit quaternions to represent orientation ensures smooth interpolation and avoids several issues which are common in Euler angle representations. The plot indicates that all quaternion components evolve continuously and smoothly, without discontinuities, further confirming the effectiveness of the trajectory planning and control strategy.

The orientation transitions reflect the three-phase motion executed by the handle: initial rotation, door opening, and final realignment. Notably, q_0, the scalar part, remains positive and relatively stable, indicating that the orientation changes occur within a consistent hemisphere of quaternion space. The vector components exhibit curved trends that correspond to the rotational transformations applied during the simulation.

Overall, this plot confirms that the robot not only maintains a rigid grasp on the handle in position space, but also precisely tracks the desired rotational trajectory.

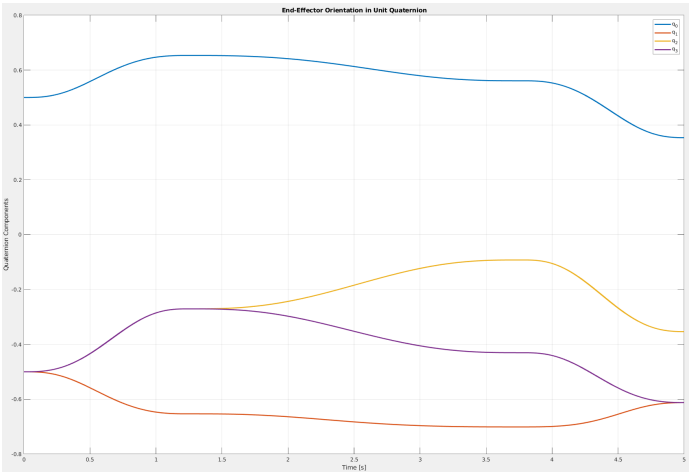

Fig. 6: End-effector orientation represented in unit quaternion form over time

E. Relative Position of End-Effector w.r.t. Handle

Figure 7 displays the relative position trajectory of the robot's end-effector with respect to the handle frame $\{H\}$ throughout the motion. This visualization serves as a validation of the rigid grasp constraint, under which the spatial relationship between the end-effector and the handle must remain fixed at all times.

The plot reveals that the relative position fluctuates only within a margin—on the order of 10^{-5} meters—due to floating-point precision and accumulated numerical errors. This variation is insignificant and confirms that the robot maintained a rigid connection with the handle throughout the entire 5-second simulation.

The consistency of the relative pose confirms the accuracy of the control scheme based on inverse differential kinematics and also verifies that the joint movements obeyed the handle motion as intended. Any large deviation in this plot would have suggested an error in rigid coupling, or a mistake in pose tracking.

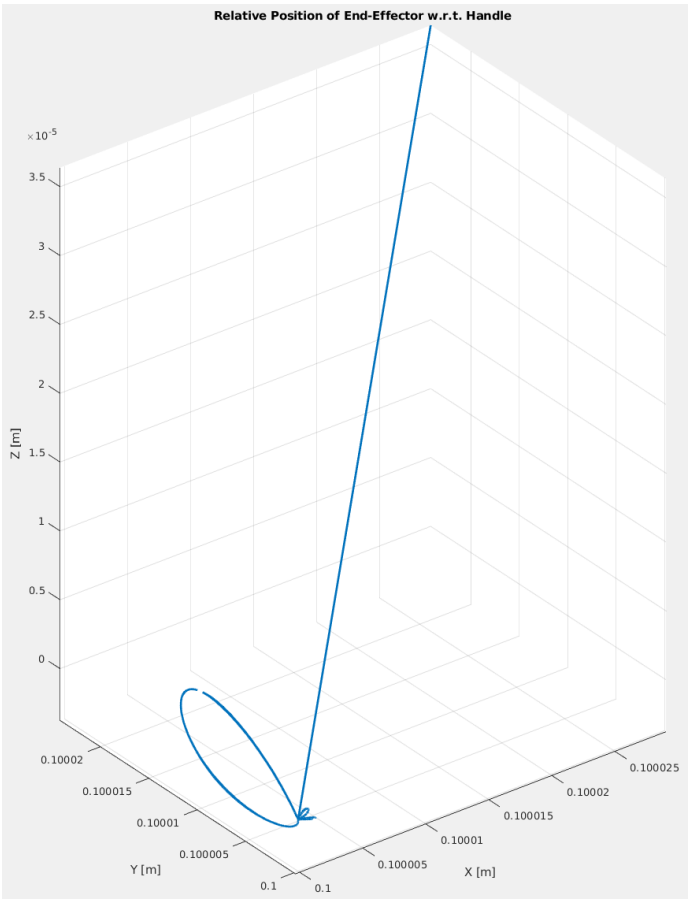

Fig. 7: Relative position of the end-effector with respect to the handle frame over time

F. Relative Orientation of End-Effector w.r.t. Handle

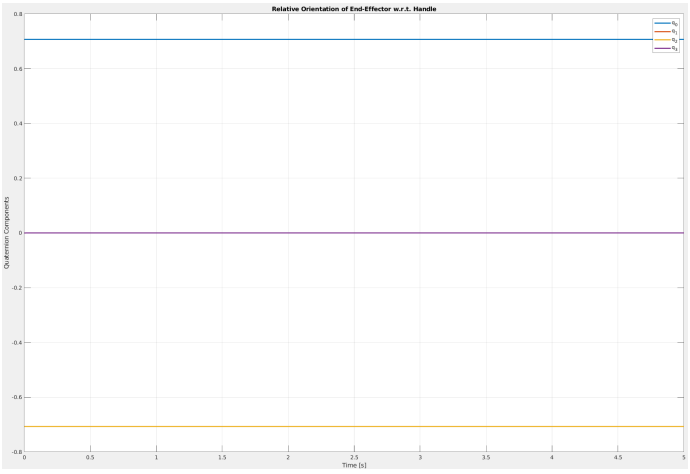

Fig. 8: Relative orientation of the end-effector with respect to the handle, represented in unit quaternion

Figure 8 illustrates the relative orientation of the robot's end-effector frame with respect to the handle frame over time, expressed as a unit quaternion $[q_0, q_1, q_2, q_3]^T$. The flat horizontal lines for all quaternion components clearly demonstrate that the orientation remains perfectly constant throughout the 5-second simulation.

This outcome confirms that the rigid grasp assumption was strictly maintained not only in position but also in orientation. The fixed quaternion values match the initially defined transformation between the handle and the end-effector (g_{HE}), which specifies a static relative orientation during the task.

Together with the relative position results, this plot reinforces that the simulation achieved a truly rigid body constraint between the robot and the handle.

G. Animation Screenshot

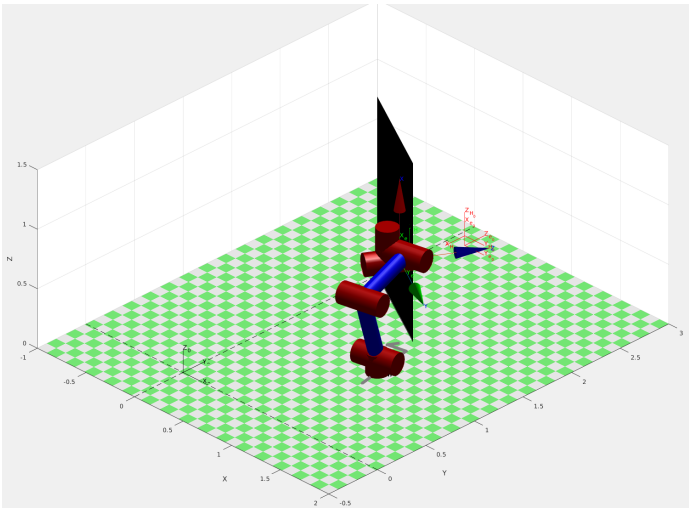

Fig. 9: Final frame of the UR10 simulation showing the door, handle, and robot in 3D space

Figure 9 presents a snapshot from the final frame of the animated simulation. It shows the UR10 robotic arm in its final pose after completing the door opening sequence. The visual scene includes the robot, the door represented as a black vertical patch, the handle visualized as a grey cylindrical object, and the environment grid for spatial reference.

Coordinate frames for the base $\{0\}$, handle $\{H\}$, and end-effector $\{E\}$ are plotted using color-coded axes. This frame confirms the successful execution of all three motion phases: handle rotation, door swing, and handle reset. The end-effector remains rigidly attached to the handle, and its final orientation and position accurately reflect the expected result.

V. Conclusion

In this project, the door and handle opening sequence planned in Part A was successfully executed using a UR10 robotic arm through inverse differential kinematics. The robot maintained a rigid grasp on the handle throughout the motion, with joint velocities computed from the end-effector trajectory using the Jacobian pseudoinverse.

The results demonstrated smooth and continuous joint position and velocity profiles, accurate tracking of the end-effector trajectory, and constant relative pose between the end-effector and the handle—validating the rigid grasp constraint. The use of unit quaternions ensured stable orientation tracking without singularities.

Overall, the simulation confirmed the effectiveness of the control approach and its suitability for executing constrained tasks in a real-life scenario.

REFERENCES

1) P. Corke, *Robotics Toolbox for MATLAB*.