A learning from demonstration approach for DORA: A Dexterous Robot Assistant

Bachelor's Thesis in Electrical Engineering

Lucas Björnquist

Hannes Dewrang

Christopher Lindell

Omanilaye Ogbowuokara

Victor Törnsjö

Supervisors:

Yasemin Bekiroglu, Ahmet Ercan Tekden

Department of Electrical Engineering

Chalmers University of Technology

Gothenburg, Sweden 2025

Abstract

Mobile manipulators are robotic systems with a mobile base and a robotic arm. They are increasingly useful for tasks performed in unstructured and dynamic environments. This project enhances the capabilities of a mobile manipulator called *DORA*, by incorporating *imitation learning*.

Preliminary experiments are conducted to assess *DORA's* existing capabilities in both simulated and real-world environments. These tests confirm that *DORA* can successfully execute tasks based on its current functionality.

Imitation learning, specifically Learning from Demonstration (LfD), enables *DORA* to learn new tasks by replicating the movements of a human demonstrator. A teleoperation setup, consisting of an *AprilTag* and *MoCap gloves*, is used to collect four dimensional data capturing the robot's 3D position and open/close state of the gripper.

The collected 4D data is used to train an *imitation learning* algorithm in a simulation environment. The comparison between the Dynamic Time Warping (DTW) and Fréchet distance (FD) scores for the training and test data shows that the trained *imitation learning* model generalizes well to the demonstrated motion patterns. In conclusion, this thesis lays a solid foundation for further improving *DORA's* dexterity through *imitation learning*.

Table of Contents

Terminology

DORA	The Dexterous Robot Assistant, located at Chalmers, is a mobile manipulator that performs tasks autonomously.
End-effector	The device at the end of a robotic arm, designed to interact with the environment.
Imitation learning	The process of acquiring new skills or behaviors by observing and replicating the actions of others.
MoCap glove	A motion capture glove used to track hand and finger movements.
AprilTag	A fiducial marker system that uses patterns that enable the robot to visualize and navigate its surroundings.

1 Introduction

As robotics advances, mobile manipulators are vital in bridging the gap between industrial automation and real-world applications. Robotic systems that combine a mobile base, a multifunctional manipulator arm, and an *end-effector*, are designed to move and interact with objects to perform complex and dexterous tasks. In this context, dexterity refers to the robot manipulators ability to perform fine motor tasks with high precision. Their dual capabilities of mobile motion and precise manipulation make them particularly suited for applications in domestic environments, such as household maintenance, where they must operate in dynamic and unstructured environments [1]. This thesis employs a mobile manipulator to perform dexterous tasks.

The *DORA* robot is a mobile manipulator capable of performing various tasks in indoor environments; its name is an abbreviation for Dexterous Robot Assistant. This project evaluates a supervised machine learning technique called Learning from Demonstration (LfD) as a method to add further dexterity to *DORA*. LfD is a subset of *imitation learning* and is a framework that enables robots to acquire new skills by mimicking actions made by human demonstrators.

The framework leverages human demonstration data for training. An operator collects demonstration data through teleoperation by remotely controlling the system. Within this project's scope, the team carries out teleoperation using *MoCap gloves* [2] and *April-*

Tags [3]. Both tracking tools enable teleoperation: *MoCap gloves* are wearable devices that use sensors to track hand movements, while *AprilTags* are typically used to estimate the 3D pose of marked objects in a robot's environment. This project uses *AprilTags* for tracking the teleoperator wrist position.

The LfD method offers several advantages over standard programming approaches, which often rely on manual programming, preprogrammed tasks or optimization-based programming [4]. Instead, LfD enables the robot to learn tasks directly from human demonstrations by tracking the wrist pose and capturing natural movement patterns. In this project, by using *MoCap gloves* and *AprilTags*, the robot merges multiple tracking systems to improve the precision of motion execution.

Implementing the LfD method in combination with teleoperation increases the robot's generality, enabling *DORA* to adapt more effectively to varying environments and tasks. This adaptability is especially useful in residential settings where the non-expert user can demonstrate an action later learned by replication, making *DORA* practical for everyday life.

1.1 Aim

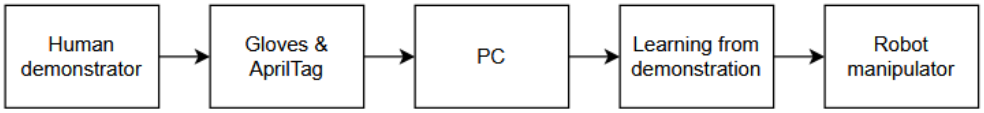


Figure 1: Visualization of the system outline.

The project aims to explore the use of *imitation learning* to enhance *DORA's* dexterity, enabling it to perform new tasks. This process is done with an experimental teleoperation setup using *AprilTags* and *MoCap gloves* as Figure 1 shows. An LfD method is incorporated in simulation testing to generate trajectories learned from the demonstrated trajectories. Finally, real-world experiments validate and integrate the code.

1.2 Implementation

The main objective of this project is to develop an LfD framework that enables a robot to learn and perform tasks. These tasks consists of movement in three dimensions, and a fourth dimension which is the open/close state of the gripper. Thee framework are to be integrated into the *DORA* robot. To achieve this goal, the project is divided into four key phases that integrate *imitation learning* with motion tracking, as illustrated in Figure 2.

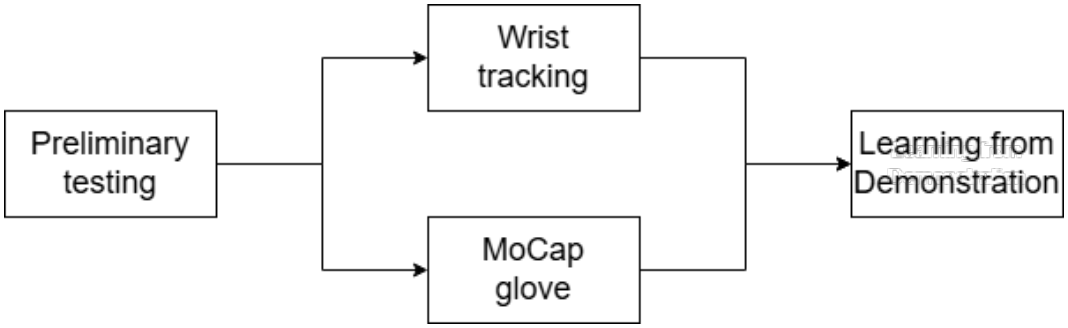


Figure 2: Flowchart of the project

1.2.1 Preliminary testing

DORA could not accurately model the effects of gravitational force. The simulation was adjusted and refined through testing and experimentation to obtain a functional simulation environment.

1.2.2 MoCap glove Integration

The hand pose tracking sub-task focuses on tracking hand and finger movements. The project is divided into two parts for this task: integrating *MoCap gloves* and *AprilTag* tracking.

A Python script is produced to extract and track finger movements using *MoCap gloves*. The script streams data over Wi-Fi to a Redis server to detect open/close hand states.

1.2.3 Wrist Pose Tracking using AprilTags

Code is developed for *AprilTag*-based wrist tracking. 3D movements are visualized and tested in Rviz, a visualization software, and another simulation environment to ensure accurate spatial localization.

1.2.4 Learning from Demonstration

The *MoCap glove* and *AprilTag* systems are combined to record 4D demonstration data with x, y and z positions and the open/-close state of the gripper.

A machine learning algorithm is trained on multiple datasets while adapting code from a research study to fit the specific use case [5]. First, Python code is developed to pre-process the demonstrated data. A neural network based on demonstration data for modeling first-order 4D dynamical systems is then trained. For testing, motion is simulated based on the initial states to evaluate if our method generates correct motion. The models are evaluated using average Dynamic Time Warping (DTWD) [6] and Fréchet Distance (FD) [7].

Finally, teleoperation using *AprilTag* tracking and *MoCap gloves* is deployed on the real robot. This step tests the synchronization of the human teleoperator's motion, including grasping movement, with the *DORA* robot's gripper.

1.3 Research question

The following research questions are addressed in this report:

- How effective is the integration of *MoCap gloves* and *AprilTag*-based teleoperation in Learning from Demonstration for *DORA*?
- How does incorporating an LfD framework improve the dex-

terity of the *DORA* robot in performing fine-motor tasks?

1.4 Delimitations

This thesis was a simulation and real-world experimental study conducted over 16 weeks and with a 5,000 SEK budget. Due to the constraints imposed by the budget and time frame, the report has several limitations regarding the method and type of data collection.

A key limitation of this project, due to time constraints, is the limited amount of demonstration data. This restriction reduces *DORA's* dexterity and generality capabilities.

Experiments use teleoperated demonstrations and a controlled indoor environment to increase the replicability of this study. Human operators perform teleoperation in a single motion sequence and the same surface, lighting and hardware/software setup - which minimizes variation in external factors that could influence the results.

The hardware in *DORA* is restricted to the existing robot arm and sensors. This limitation confines the research to a singular hardware configuration and removes the opportunity to investigate the efficiency of alternative hardware solutions for executing dexterous tasks.

2 Background

This project aims to build new skills for *DORA*, a mobile manipulator. As shown in Figure 3, the system setup consists of an MiR200 as the mobile platform, with a 6-DoF UR10 robotic arm as the manipulator, and a two-finger OnRobot gripper as the *end-effector*. The mobile base has a sensing system comprising of two 3D-cameras, four ultrasonic sensors and two laser scanners.

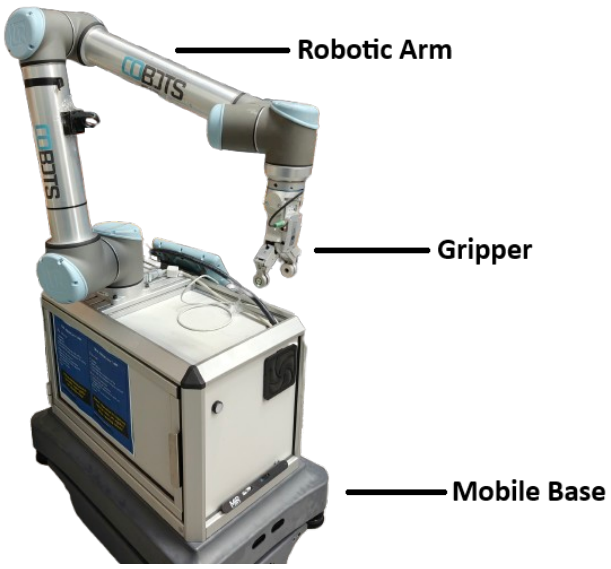


Figure 3: The *DORA* robot: A mobile manipulator comprising an MiR200 mobile base, UR10 arm manipulator, and OnRobot gripper [8]

A Force-Torque (FT) sensor is integrated into *DORA*, enabling it to measure both linear and rotational forces applied at the *end-effector*. This sensor plays a critical role in force-feedback control, which is a control system methodology where the resulting force is measured and used as a feedback parameter within the control

loop to adjust the robot's actions.

2.1 System components

This section introduces the different tools used in the project. It also presents the key concepts related to this thesis and ethical considerations.

2.1.1 ROS

Robot Operating System (ROS) is a standard for robot programming and development that utilizes middleware and an open-source software framework for robotic applications [9]. ROS facilitates communication between multiple parts of the robotic system and provides a wide range of toolboxes and libraries for building and managing robotic applications; hence, it plays a key role in enabling LfD techniques on real robots.

At the file system level, ROS organizes its software into packages, which handle specific functionalities or hardware components of the robot. The next level in the ROS hierarchy is the computation graph level, a data-processing and communication system connected by nodes (individual programs). Nodes interact with each other by publishing and subscribing to streams of messages called topics. This architecture supports modularity, allowing sensors, actuators and controllers to be easily integrated and reused for different tasks.

The ROS community network enables remote users to share tools

and packages, access tutorials, receive support and contribute to the ROS infrastructure by improving its codebase and features. This project utilizes ROS Noetic Ninjemys, which is distributed on Ubuntu 20.04, a Linux-based operating system.

2.1.2 Simulation and visualization software

The software used in this project includes CoppeliaSim and ROS visualization (RViz). CoppeliaSim is a GUI-based simulation program that uses 'scenes' to model and test robotic systems, whereas RViz is a 3D visualization tool in ROS. RViz supports multiple camera perspectives through plugins that subscribe to ROS messages. In simulation and robot testing, the program allows users to analyze the completed robot tasks performed in the simulation environment.

To prevent unexpected behavior or accidents, RViz and CoppeliaSim ROS simulation software visualize the robot's movements prior to execution with the real robot using ROS motion planning capabilities. Both simulation software packages are well-supported by the Ubuntu/Linux operating system, and Python code written in Visual Studio and Jupyter Notebook IDEs is used alongside the simulations to step through the robot program interactively. These IDEs allow for easy testing, debugging, and visualization of the robot's behavior during development.

2.1.3 MoCap gloves

MoCap gloves are soft wearable sensors that provide high-precision tracking of hand and finger movements. In this project, the Rokoko Motion Capture System (Rokoko) is used to collect and analyze motion data. Each glove contains 7 sensors utilizing an Inertial Motion Unit (IMU) based system and Electromagnetic Field (EMF) sensors to track and record hand movements [10]. The IMU unit integrates an accelerometer and gyroscope, which capture data on speed, acceleration and rotation [11]. This attribute facilitates *imitation learning* by enabling the robot to learn both positional and dynamic variations in real time, rather than relying solely on pre-recorded datasets.

The motion data is livestreamed via Rokoko Studio Plus, which has a rotational dynamic accuracy of $\pm$ 1 degree, an internal framerate of 400 fps and an external framerate of 100 fps [12]. These specifications ensure high positional accuracy of hand movements for LfD tasks. Livestreaming motion data from *MoCap gloves* to external programs requires a subscription to Rokoko Studio Plus, which was funded through the project's budget.

2.1.4 AprilTag tracking

One critical component of LfD is ensuring that the robot can accurately track movement. Visual trackers, such as *AprilTag*, are particularly effective for this purpose. *AprilTag* is an open-source fiducial marker system consisting of square-shaped reference tags [3]. A camera detects these tags and interprets them

using marker-based algorithms, allowing robots to estimate object positions and orientations in 3D space.

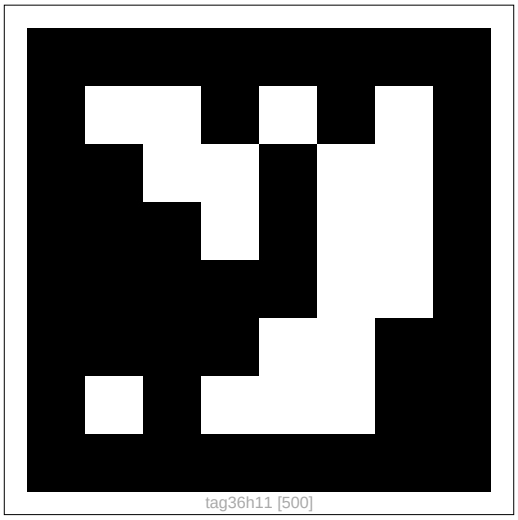


Figure 4: AprilTag marker

AprilTag markers, as shown in Figure 4, are encoded in a black-and-white grid representing a binary pattern [3]. This unique design offers several advantages. First of all, it contains redundant bits that allow for reconstruction of unclear markers [13]. Additionally, *AprilTag* uses a lexicode-based encoding system to ensure a minimum Hamming distance between different tags, reducing false positives (incorrect detection of a marker when none is present) and improving detection accuracy.

In teleoperated LfD, *AprilTags* are often attached to the camera or the demonstrator's hands. When a human moves, the camera system captures the movements of the fiducial markers, allowing the robot to learn the corresponding motion task through demonstration. A study comparing the efficacy of differ-

ent fiducial marker systems highlighted the limitations that must be accounted for while using *AprilTag* [13]. The key takeaway is to minimize occlusion; since the *AprilTag* tracking algorithm is based on edge recognition, the edges of the marker must remain visible. Even partial occlusion can significantly impact accurate detection.

2.2 Related Work

Robot manipulation in contact-rich environments requires adaptive strategies to ensure safe robot interaction with both objects and surroundings [14, 15]. While prior work on the *DORA* robot [16] focused on trajectory planning, the integration of an FT-sensor on the *end-effector* and velocity control, Learning from Demonstration (LfD) offers a data-driven alternative to increasing environmental adaptability.

Recent advances in LfD leverage advanced deep machine learning methods to reduce the required demonstrations. For example, Tekden *et al.* used neural fields and shared embeddings to jointly model 3D scenes and motion trajectories, enabling efficient *imitation learning* in novel scenes [17]. Another study applied Convolutional Neural Networks (CNNs) to vision-based data specifically for cleaning tasks [18]. However, this approach was not well suited to the *DORA* project due to its reliance on processing image and spatial data rather than temporal data e.g., from a Motion Capture setup.

A journal article by Pérez *et al.* explored LfD using Contrastive Learning (CL) with Deep Neural Networks (DNNs) [5]. DNNs are a collection of computer algorithms that extract patterns at multiple levels of abstraction. This paper bases the Convergent Dynamics from Demonstrations (CONDOR) framework on first-order and second-order dynamical systems, which model position and velocity-based behaviors, respectively. CONDOR aligns well with continuous trajectory MoCap data, so this project adopts its LfD algorithms for simulation experiments.

Additional studies have explored alternative approaches, such as motion and vision-based tools, for collecting demonstration data for LfD teleoperation. [19] combines a wearable IMU-based and Optical Motion Capture system along with statistical and optimization methods to optimize the trajectory, whereas [20], relied solely on non-wearable IMU devices to complete spatial human tasks. In contrast, this thesis investigates a hybrid approach that combines motion tracking and vision tracking to address occlusion issues, reduce dependency on FT sensors and improve usability for human demonstrators.

2.3 Ethical considerations

Improved object manipulation skills in robots can help automate monotonous tasks. This advancement can have positive implications because repetitive physical tasks often result in poor ergonomics for humans and can cause injuries over time. However, these advancements in robot dexterity comes with risks. For ex-

ample, the deployment of robots in everyday settings increases the risk of impact or crushing-related injuries, particularly among vulnerable groups such as children and pets.

Despite these ethical concerns, the LfD method implemented in this project is limited to domestic environments and requires further work to be integrated into the more complex realities of the industrial and service sectors.

3 Method

The following method section introduces the theoretical foundations underlying the door opening task, multi-modal motion tracking, data processing, and learning from demonstration algorithms, along with other implementation details.

3.1 Velocity-based controller in the door-opening task

The mobile manipulator performs door-opening tasks using a velocity-based controller that relies on real-time feedback from force–torque (FT) sensors, with *DORA* using the Robotics Toolbox for Python to compute the Jacobian for joint velocity control [21, 22]. Table 2 lists the key parameters used in the control algorithm.

Table 2: Definitions of the variables used in the velocity-controller

Variable	Definition
$J(q)^+$	Pseudoinverse of the Jacobian matrix of the manipulator
q	Joint configuration
v_{ref}	Reference velocity
v_{linear}	Linear velocity
v_{angular}	Angular velocity
$s(\hat{r})$	Skew-symmetric matrix of the unit vector $\hat{r}$
v_d	Desired *end-effector* velocity
α	Positive control gain
ω	Angular velocity
$\hat{r}$	Unit vector representing direction from *end-effector* to estimated door hinge location

$\hat{p}_o(t)$	Estimated door hinge location
p_e	Robot *end-effector* position
F	Measured force
F_r	Radial force
F_{rd}	Desired radial force
β	Positive control gain
v_f	Force feedback term
$\Delta F_{\hat{r}}$	Difference between actual and desired radial force
$I(\Delta F_{\hat{r}})$	Integral of the difference between actual and desired radial force
$\dot{p}_o$	Change in the estimated hinge location
γ	Hinge update law constant
α_f	Force feedback in hinge location update constant
α_i	Integral feedback in hinge location update constant
v	*End-effector* velocity

The system updates joint velocity measurements derived from Equation 1, which are then used to control the *end-effector's* motion:

$$\dot{\mathbf{q}} = \mathbf{J}(\mathbf{q})^{+}\mathbf{v}_{ref} \tag{1}$$

where $\mathbf{J}(\mathbf{q})^{+}$ represents the pseudoinverse of the *end-effector* Jacobian matrix at a joint configuration $\mathbf{q}$, and $\mathbf{v}_{ref}$ is the reference velocity vector.

As shown in Equation 2, the reference velocity is a combination of linear and angular components:

$$\mathbf{v}_{ref} = \begin{bmatrix} \mathbf{v}_{linear} \\ \mathbf{v}_{angular} \end{bmatrix} = \begin{bmatrix} -s(\hat{\mathbf{r}})\mathbf{v}_d + \alpha\hat{\mathbf{r}}\mathbf{v}_f \\ \mathbf{0}_{2\times 1} \\ \omega \end{bmatrix} \tag{2}$$

The key elements include a unit vector $\hat{\mathbf{r}} = \frac{\hat{\mathbf{p}}_o(t) - \mathbf{p}_e)}{\|\hat{\mathbf{p}}_o(t) - \mathbf{p}_e\|}$, where $\hat{\mathbf{p}}_o(t)$ is an estimate of the hinge position, $\mathbf{p}_e$ is the *end-effector* position, $s(\hat{\mathbf{r}}) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \hat{\mathbf{r}}$ shows the generated tangential direction of the door and the angular velocity $\omega = \frac{\|\mathbf{v}_{linear}\|}{\|\hat{\mathbf{p}}_o(t) - \mathbf{p}_e\|}$.

The force compensation term $\mathbf{v}_f$ is detailed in Equation 3 incorporates integral control:

$$\mathbf{v}_f = \Delta\hat{F}_r + \beta\mathcal{I}(\Delta\hat{F}_r) \tag{3}$$

where $\Delta\hat{F}_r = F_r - F_{r_d}$ represents the radial force error, with $F_r = \hat{\mathbf{r}} \cdot \mathbf{F}$ computed from raw measurements $\mathbf{F}$.

In Equation 4, the hinge position estimate is updated according to:

$$\dot{\mathbf{p}}_o = \gamma\|\hat{\mathbf{r}}\|^{-1}((\alpha_f\Delta\hat{F}_r + \alpha_i\mathcal{I}(\Delta\hat{F}_r)))\mathbf{v} \tag{4}$$

where $\mathbf{v} = \mathbf{J}\dot{\mathbf{q}}$ is the *end-effector* velocity, and γ, α_f, α_i are constants.

The control goals are summarized in Equation 5 as follows:

$$\hat{\mathbf{r}} \to \mathbf{r}, \quad \mathbf{v} \to s(\hat{\mathbf{r}})v_d, \quad \int \Delta F_{\hat{r}}\, dt \to 0, \quad F_r \to F_{r_d} \tag{5}$$

These objectives ensure that the estimated hinge location, *end-effector* velocity and applied radial converge to their real values and the radial force error integrates to zero.

3.1.1 Coordinate transformations

Mobile manipulators have many data inputs from sensors such as grippers and cameras, and each part has its unique orientation in space. ROS TF2 is a transformation library that organizes the relative positions and orientations of robot components over time using rigid body transformations between coordinate frames [23]. As seen in Figure 5, rigid body transformations use frames, which consist of an origin and an orthogonal 3D coordinate axes, for the robotic workspace [24]. TF2 transforms the sensor data from the sensor's frame (a global coordinate system) into the robot's base frame (the robot's central reference point attached to its base) by multiplying it with a transformation matrix, as expressed in Equation 6.

$$\mathbf{F}_{\text{base}} = \mathbf{T}_{\text{base}\leftarrow\text{space}} \cdot \mathbf{F}_{\text{space}} \tag{6}$$

3.2 Demonstration data collection

The first phase of demonstration data acquisition involves creating a virtual simulation setup to effectively model tasks. This is achieved using a python-based robot visualizer, which enables real-time visualization of the *DORA* in a browser via a localhost interface.

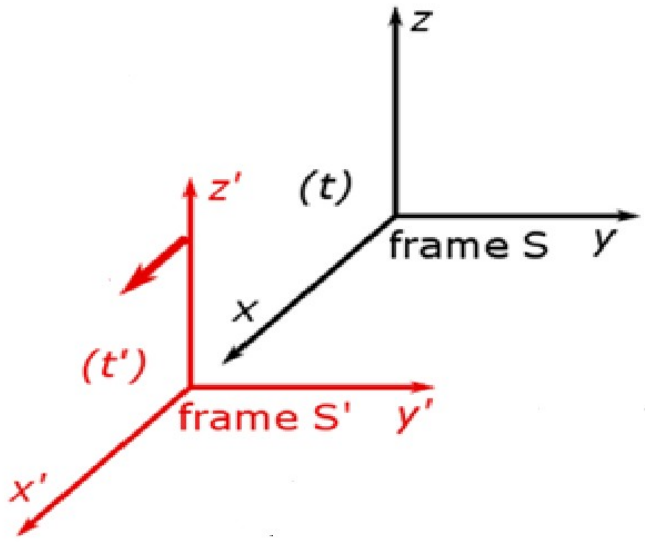


Figure 5: v Graphical representation of the transformation of sensor frame S, to base frame, S', by multiplying with a rotation matrix. [25]

Our framework, based on *Mocap gloves* and *AprilTags*, is to collect demonstration data with a ZED 2 camera.

3.2.1 Hand kinematics extraction

Redis (Remote Dictionary Server) is used as the data handling system for glove data. It is a relational database that accesses data directly from RAM and stores values on the disk. Key-value systems map to data structures such as lists and strings, with keys for easier and quicker access and lower latency [26].

This project uses the JavaScript Object Notation (JSON) module to store kinematic data. Its architecture supports the flow of kinematic data and a total of 21 joint angles per hand through the computer browser, server and database, allowing other robot

systems to also access real-life data [27].

A Python script parses the JSON file and extracts the relevant data, which is then used to compute the Cartesian distances between the fingertips and the wrist using Equation 7.

$$d_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{p}_{\text{wrist}} - \mathbf{p}_{i,\text{tip}}\| \tag{7}$$

where $N = 5$ corresponds to the five fingers. This average is then normalized using min-max scaling:

$$d_{\text{norm}} = \frac{d_{\text{avg}} - d_{\text{min}}}{d_{\text{max}} - d_{\text{min}}}, \quad d_{\text{norm}} \in [0, 1] \tag{8}$$

The normalized open/close state value d_{norm} in Equation 8 is used as a real-time, scalar indicator of the hand state.

3.2.2 AprilTag pose estimation algorithms

Figure 6: *MoCap glove* and *AprilTag* setup

Different sizes of *AprilTag* markers are printed and evaluated for the best performance. A training environment is implemented with the ZED 2 camera mounted on top of a metal fixture and monitoring the marker's upward, downward, horizontal, diagonal and rotational movements.

A 10×10 cm *AprilTag* is used as the final marker size, as it offers improved depth perception and more consistent tracking at higher speeds and greater distances from the camera, without compromising ergonomics or ease of use for the operator.

A custom support structure is constructed to mount the *AprilTag* onto the teleoperator's wrist as shown in Figure 6. *AprilTags* are used to track visual fiducial markers. The tag position is recorded and used as a reference pose upon initial detection. For example, if the tag is subsequently moved 10 cm along the x-axis, the robot

will mirror this motion by moving 10 cm along its x-axis [28].

$$\min_{\mathbf{rvec},\mathbf{tvec}} \sum_i \left\| \mathbf{x}_i - \pi \left(\mathbf{K}[\mathbf{R} \mid \mathbf{t}] \mathbf{X}_i^{(h)} \right) \right\|^2 \tag{9}$$

$\mathbf{x}_i$	2D image coordinates
π	The projection function: converts a 3D point in camera coordinates to a 2D point
$\mathbf{X}_i^{(h)}$	3D world coordinates
$\mathbf{R}$	Rotation matrix
$\mathbf{t}$	Translation vector
$\mathbf{K}$	Camera intrinsic matrix
rvec	Rotation vector to be optimized
tvec	Translation vector to be optimized

AprilTag tracking estimates the pose of a tag relative to the camera by comparing the detected *AprilTag* corners in the image with their known 3D reference coordinates and finding the rotation and translation vectors that defines the pose of the *AprilTag* with respect to the camera frame. This is achieved through minimizing the reprojection error described in Equation 9.

3.3 Teleoperation

In the general teleoperation setup, all tracking tools (motion capture, vision systems) and ROS platforms are synchronized in order to record demonstration data and perform real-life experiments.

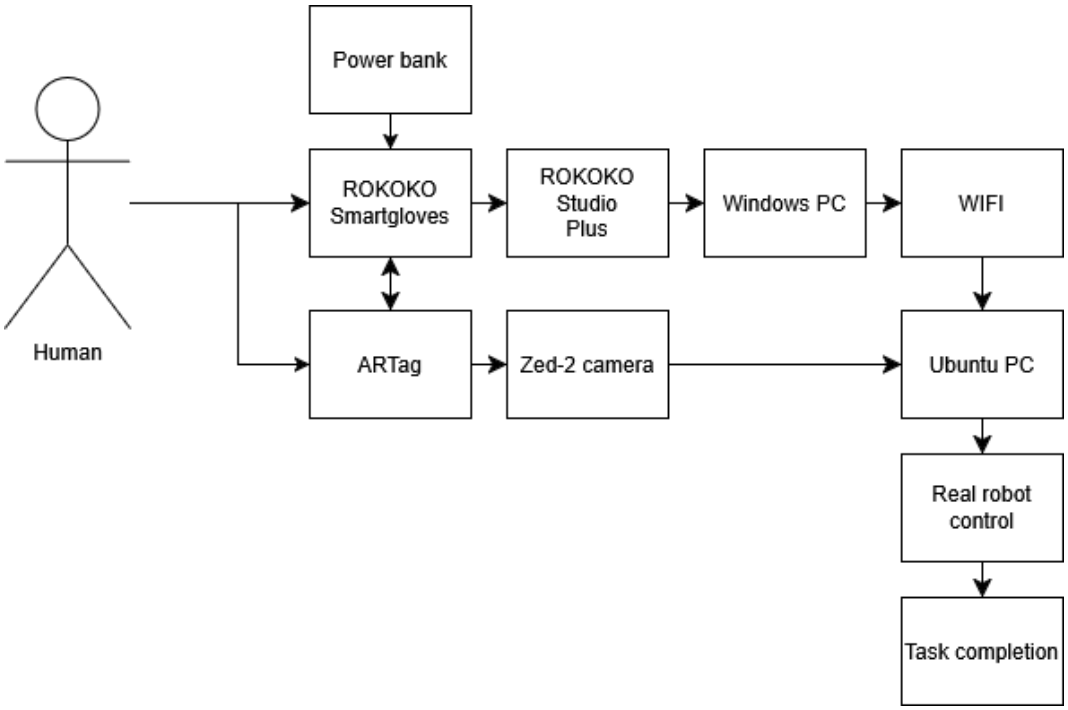


Figure 7: Overview of the teleoperation setup. The Ubuntu PC runs the *AprilTag* tracking system and processes data for robot control, while the Windows PC runs the Rokoko motion capture software. Both machines are connected via a local Wi-Fi network to enable real-time data synchronization across platforms

As shown in Figure 7, the system was configured using an Ubuntu PC as the main computer which is responsible for running the *AprilTag* tracking program. Since the Rokoko software is not compatible with Linux, a separate Windows PC was used to run it. The two computers were connected via a closed Wi-Fi network using a router.

The Windows PC streamed motion capture data from Rokoko Studio, which was transmitted in real time as a JSON file. This data was received by a Python script running on the Ubuntu PC, which processed the glove data for further use in the robot control system. This setup enabled the integration of both systems, effectively bypassing the compatibility issues between Rokoko and Linux.

3.4 Learning from Demonstration

Learning from Demonstration involves the following steps: recording demonstrations, pre-processing data, extracting key features such as hand poses, training the neural model on demonstration data and finally, evaluating the model's performance.

3.4.1 CONDOR framework

The CONDOR method models motions as time-invariant dynamical systems or motion primitives, described by:

$$\dot{x} = f(x)$$

where $x \in \mathbb{R}^n$ is the robot's state (e.g. position, velocity), and f is a nonlinear, continuous and differentiable function [5]. The dynamical system and robot system are transferable through low-level controllers.

A motion is stable if the robot always reaches the goal state x_g from any initial state, i.e.,

$$\lim_{t\to\infty} \|x - x_g\| = 0.$$

with the time derivative at the goal state being zero.

With a given demonstration $\mathcal{D}$ and trajectories, τ_i, CONDOR learns and models a parametrized dynamical system $f_\theta^{\mathcal{T}}$ that imitates the forward divergence between demonstration and robot statistical trajectory distributions p_θ.

In the process of finding the parametrized vector θ^*, $f_\theta^{\mathcal{T}}$ also ensures convergence to x_g. Put together, the optimization problem becomes:

$$\theta^* = \arg\min_\theta \; D\left(p^*(\tau) \| p_\theta(\tau)\right), \qquad \text{s.t.} \quad \lim_{t\to\infty} \|x_t - x_g\| = 0.$$

where x_t is the discrete time step.

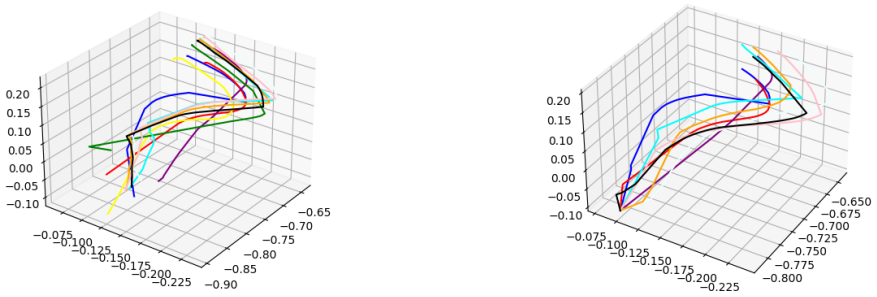
There are two ways to use CONDOR for robot control. In the first method with online control, the robot measures its current state x_t at each time step and uses CONDOR's learned dynamical system $f_\theta^{\mathcal{T}}$ to compute the desired velocity/acceleration $\dot{x}_t$. As a concluding step, it sends $\dot{x}_t$ to a low-level controller for execution. This method is suitable for dynamic environments since it can react to disturbances in real-time.

The second method uses offline control for generated trajectories

and assumes that tracking is perfect. CONDOR generates a full trajectory by recursively applying $f_\theta^{\mathcal{T}}$ from an initial state x_0. The trajectory is then stored and tracked by a buffer. This method is advantageous for motions such as writing tasks, where the exact shape of the trajectory, such as the curves and strokes of letters, is critical.

3.4.2 Data pre-processing

In LfD, human demonstrations can exhibit endpoint variance due to non-uniform speed and acceleration and inconsistent start and end positions. Such variability is a source of data-driven noise for the LfD algorithm. Therefore, a pre-processing strategy is employed in this project.



(a) Before processing **(b)** After processing

Figure 8: Datasets before and after being processed

Initially, the collected data and corresponding 3D plots, shown in Figure 8a, are examined to identify and remove outlier trajectories. Linear interpolation is then applied to each dataset. This process involves calculating the shared endpoint, which is the average of all the endpoints across the trajectories. Each original trajectory is trimmed to 75% of its length, and the remaining

25% is reconstructed by interpolating between the last retained point and the shared average endpoint. Each trajectory is also trimmed at the beginning, with 30 to 50 initial data points removed to ensure a smooth start The trimmed and interpolated segments are then concatenated to form the complete trajectory, which can be seen in Figure 8b, and the code is used to compute the average endpoint across all demonstrations.

This approach is theoretically motivated by the *imitation learning* model, where dynamical systems are modeled as movement primitives with attractors (i.e. goal states) [5]. These models assume that movements are governed by differential equations that converge toward an attractor. Hence, the data pre-processing step emulates this attractor behavior by explicitly guiding each trajectory toward a common endpoint.

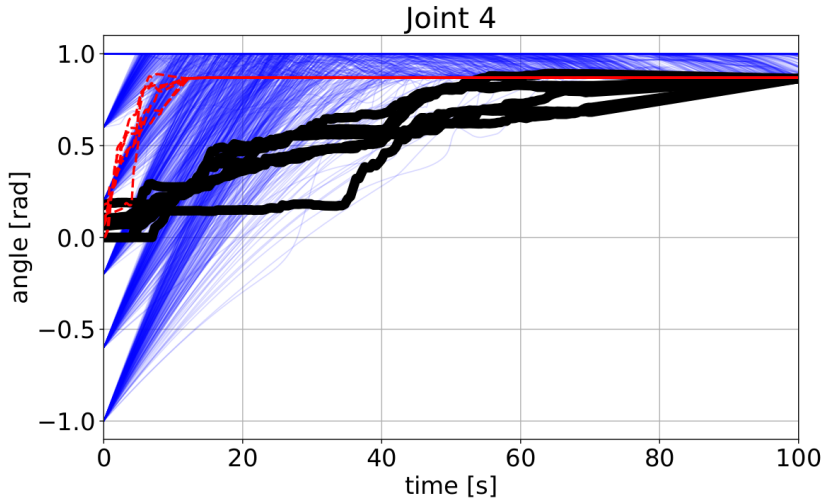


Figure 9: Simulation trajectories of joint 4 over time, shown as a function of the gripper state. Blue trajectories are the models under different initial conditions, the black trajectories are the demonstration data and the red point is the convergence point.

This is confirmed by examining Figure 9: variance in the endpoint of the gripper state is reduced in the refined dataset, which improves training consistency and results.

3.4.3 Evaluation metrics

To evaluate the model's performance after training with the LfD algorithms, demonstration trajectories are compared with simulation trajectories. Dynamic Time Warping Distance (DTWD) and Fréchet Distance (FD) are evaluation metrics that capture both the temporal and spatial aspects of motion [6, 7, 29].

DTWD is particularly suited for scenarios where temporal inconsistencies exist, for example, demonstrations executed at varying speeds [6]. DTWD measures the similarity between two sequences and uses a discrete algorithm that allows non-linear alignments of sequences by "warping" the time axis to find the best match between points. While it is robust to time shifts, the DTW distance remains sensitive to outliers and noisy data.

On the other hand, FD is a continuous spatio-temporal measure that compares the shapes of trajectories by preserving the order and location of points along the curves [7]. It finds the smallest possible maximum distance between two points on the trajectories, after adjusting the speed at which each trajectory is traversed. FD is useful for tasks where the spatial characteristics of the demonstration data are important, such as the path taken

through space. However, the metric is sensitive to temporal alignment and noise, especially outliers.

Together, DTWD and FD are complementary metrics on trajectory similarity, facilitating a comprehensive evaluation that accounts for time variations and spatial accuracy.

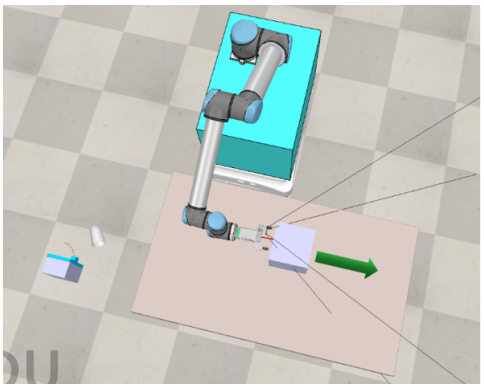
4 Results

This section provides an overview of the setup procedures, the final experimental setup designed for collecting demonstration data as well as the algorithms and virtual environments used for LfD.

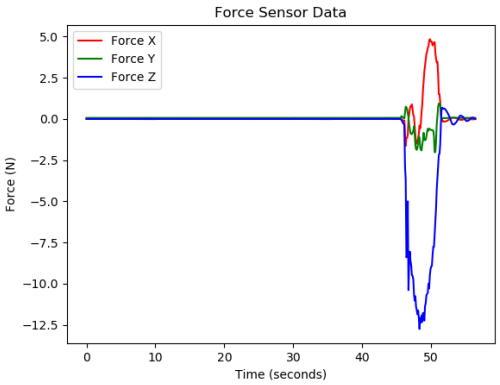
4.1 Preliminary door opening task

The project started with investigating *DORA's* ability to perform a basic door opening manipulation task. The problems encountered were gravity-related simulation errors and an undefined gripper base frame. The initial step of evaluating *DORA's* ability to perform the door opening task involves simulation testing in CoppeliaSim. When gravity is enabled in CoppeliaSim, the robot arm collapses onto the table in the simulation. Due to time constraints, it is deferred to future work.

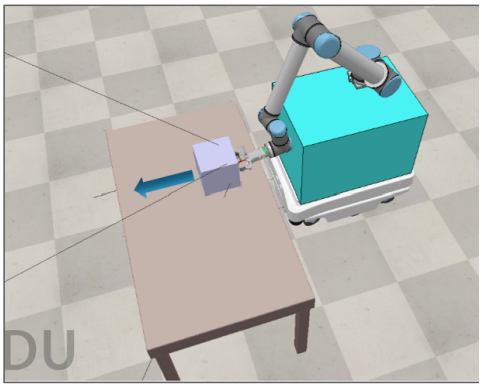
Simulation tests in both CoppeliaSim and RViz are conducted to determine where the robot's base frame operates from. The objective is to plot the forces exerted by the robot in the x, y and z directions during the motion and analyze the results to identify the robot's base frame. In this setup, *DORA* pushes a cuboid object in CoppeliaSim using the motion planning function in RViz, while a Python script generates a subsequent force-time diagram.



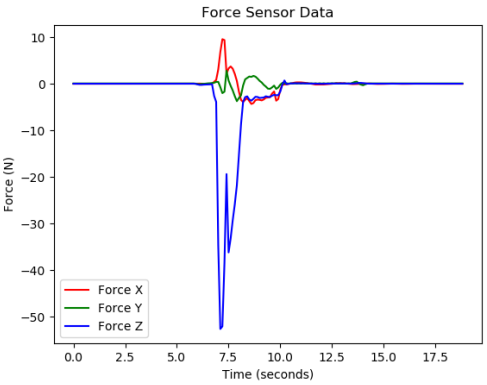
(a) Gripper pushing forward, robot arm pushing left



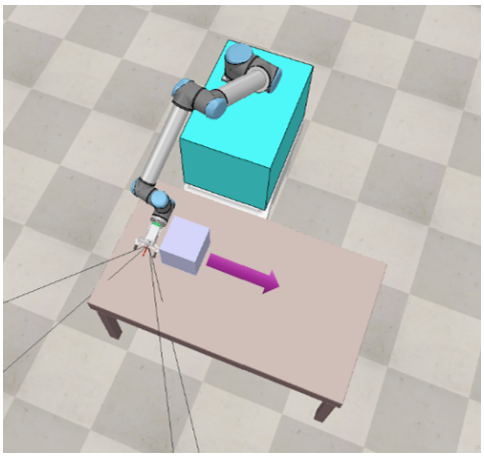
Force Sensor Data, (a)



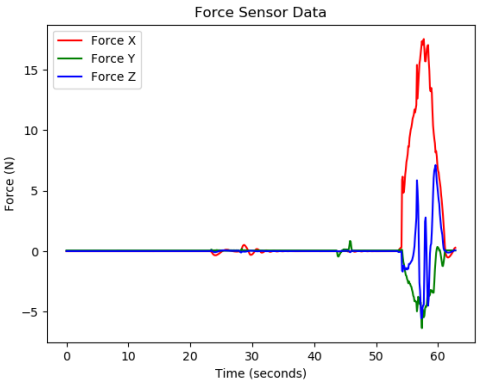
(b) Gripper pushing forward, robot arm pushing forward



Force Sensor Data, (b)



(c) Gripper pushing left, robot arm pushing left



Force Sensor Data, (c)

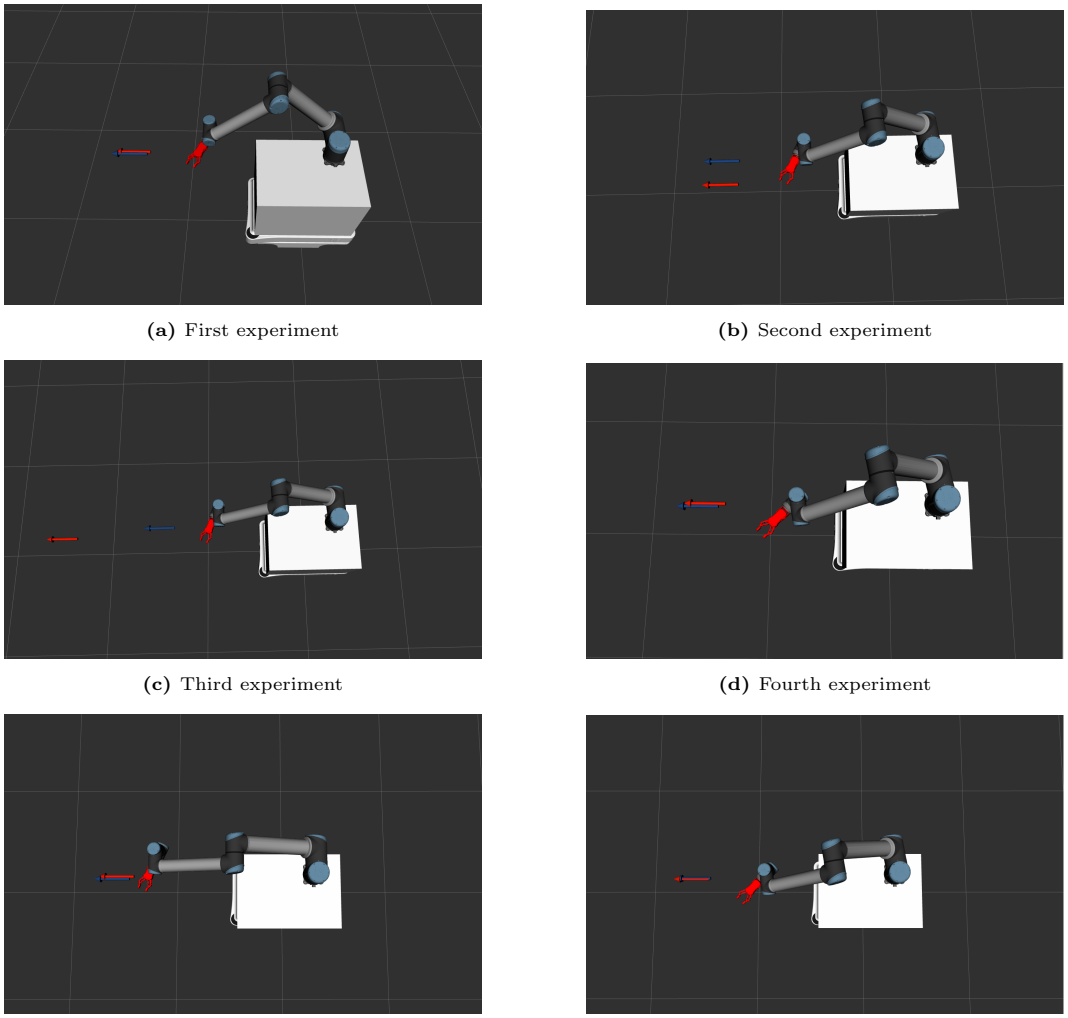
Figure 10: Comparison of force profiles from different pushing directions simulated in RViz.

From Figures 10b and 10c, it is apparent that forward motion produces a force along the negative z-axis, while leftward movements results in a positive resultant force in the x-direction.

In Figure 10a, the robot arm pushes left and the gripper pushes forward relative to the sensor frame, creating a force in the negative z-direction. This confirms that the force frame is oriented from the gripper's perspective.

In real-world testing, the manipulation task is to open a small cabinet placed on a table from various angles, requiring *DORA* to adopt different poses and force-torque values to open the door successfully.

Figure 11 shows the door opening sequence in RViz, with blue arrows representing the estimated force vectors and red arrows indicating the actual trajectory. *DORA* achieved a 60% success rate in 10 real-world tests, demonstrating a smooth motion path in successful attempts to open the door. However, the robot encountered issues, including excessive torque application and stopping midway in the experiment. A hypothesis is that the force-torque sensors yield incorrect values after extended use, leading to excessive twisting or weakened grip while completing the door-opening task.



(a) First experiment

(b) Second experiment

(c) Third experiment

(d) Fourth experiment

(e) Fifth experiment

(f) Sixth experiment

Figure 11: Door opening simulations visualized in RViz. Red and blue arrows represent force vectors.

4.2 Hand pose tracking

This section presents the results from testing the implemented code for the *MoCap gloves* and the *AprilTag* camera system.

In the first phase of data acquisition, two virtual environments are developed to support more effective task simulation: a Simple

Robot Visualizer as well as a PyTorch-based setup for the LfD framework. Figure 12 shows the robot visualizer which provides real-time visualization of *DORA*.

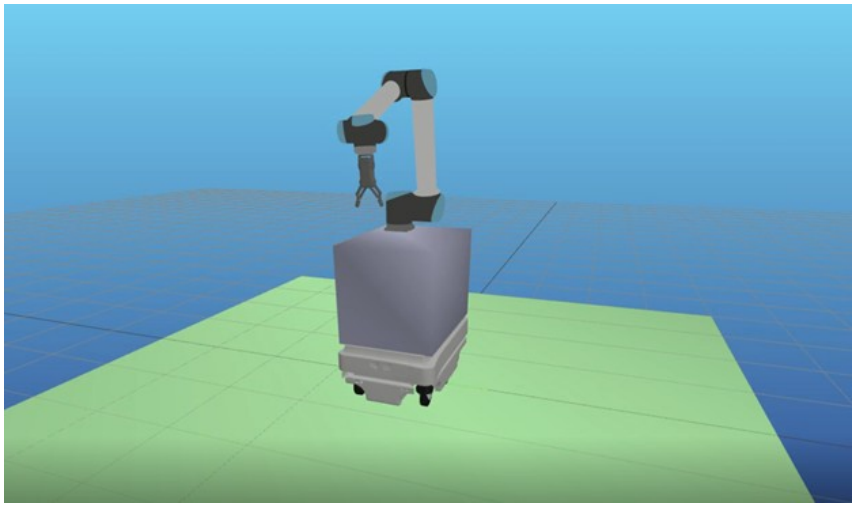


Figure 12: The robot visualizer environment

In the second phase of data collection, a total of 20 demonstrations: 10 sets of 2 distinct S-shaped trajectories and C- shaped trajectories are performed by a human demonstrator and recorded in a training environment with a ZED 2 stereo camera.

4.2.1 MoCap gloves

The glove system is tested for accuracy and precision in detecting hand open/close state through a series of test cases designed to assess its responsiveness and reliability across various hand configurations. These test cases include individual finger movements in various positions, dynamic finger movements and transitions between a fully open hand and a fully closed fist. The gradual progression from open to closed hand postures was examined to ensure the system could accurately capture intermediate config-

urations.

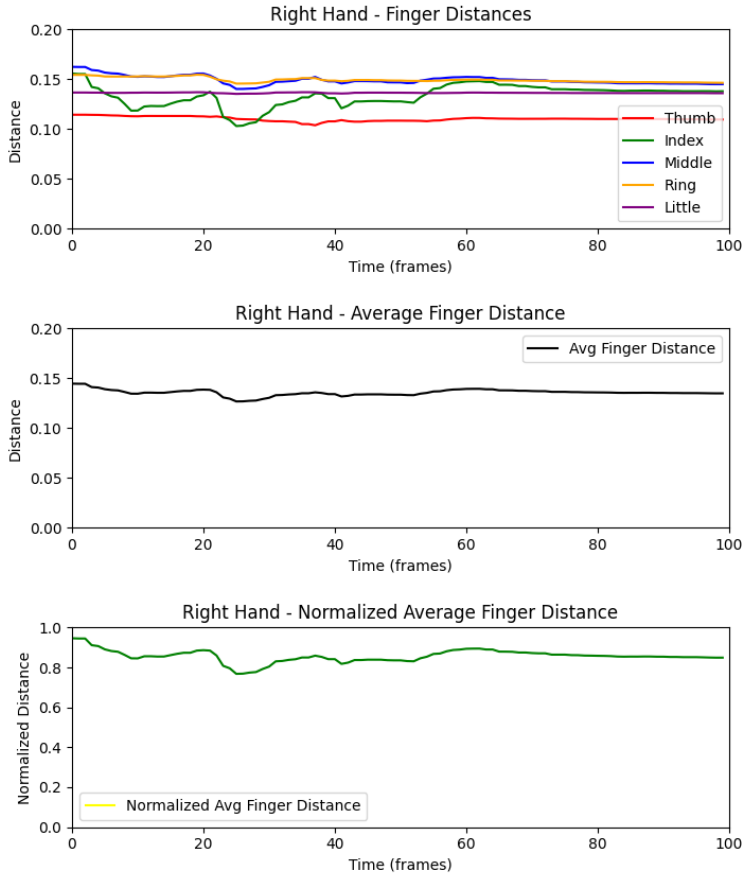


Figure 13: Plots showing the finger distances for each finger, the average finger distance and a normalized average finger distance.

The resulting plots in Figure 13 visualize the sensor readings during a sequence of hand movements. The first sub-plot displays the distance between each finger and the wrist, with different colors representing each finger. The second sub-plot shows the average distance across all five fingers, providing an overall indication of hand open/close state. Lastly, the third sub-plot illustrates the

normalized distance values, which were used as input to trigger gesture-based commands.

The normalized values were also printed in real time in the computer terminal for comparison and debugging purposes. These readings helped verify that the glove system captured relative hand configurations accurately, although minor delays and fluctuations were observed during rapid movements.

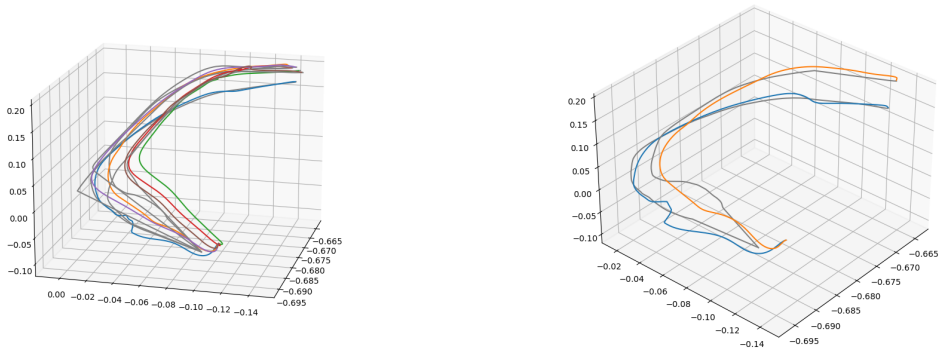
4.2.2 Pose tracking using AprilTag

Figure 14: The pose of the *AprilTag* relative to the camera shown both in the visualization environment and in the real-world.

As shown in Figure 14, RViz can show the *AprilTag* and also store the active position in relation to the ZED 2 camera. There is small amount of buffering at small distances between the *AprilTag* and the ZED 2 camera, but it tracks the position smoothly. During larger distances, it buffers and can lose track of the tag. The pose,

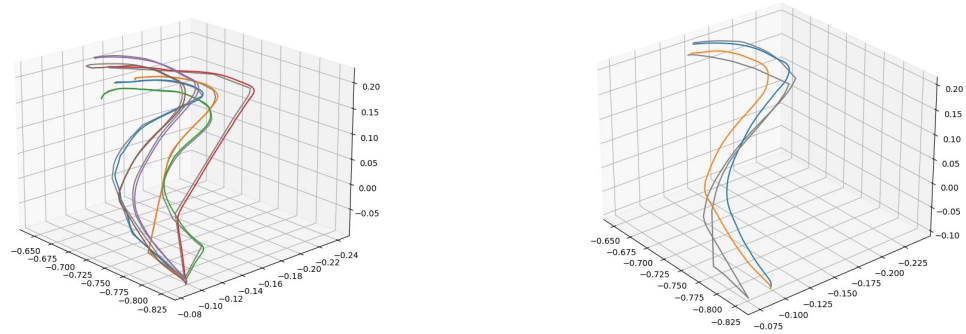
which includes both the position and rotation, of the *AprilTag* is tracked successfully.

4.3 Simulation experiments



(a) Training data **(b)** Test data

Figure 15: Three dimensional plot comparing the C-shape trajectories of the training and test datasets.



(a) Training data **(b)** Test data

Figure 16: Three dimensional plot comparing the S-shape trajectories of the training and test datasets.

As shown in Figures 15 and 16, two types of motion were recorded: S-shape and C-shape. A total of 8 trajectories were recorded for each motion type. Six trajectories were used for training the model, and the remaining two were reserved for testing, achiev-

ing an 75/25 training-test split.

In Figures 15a and 16a, the collected training data is shown in color, alongside the model-generated training trajectories shown in grey. Similarly, in Figures 15b and 16b, the same model is applied to the test dataset.

Table 3: DTWD and FD scores by motion type and data split

S-motion	**DTWD**	**FD**
Training data	**0.044**	**0.153**
Test data	0.053	0.239
Error	-0.009	-0.086

C-motion	**DTWD**	**FD**
Training data	0.075	0.345
Test data	**0.023**	**0.095**
Error	0.052	0.250

On average, the DTW distance is 0.049 while the Fréchet distance is 0.208.

Table 3 displays the DTWD and FD scores for two datasets representing S-shapes and C-shape motions. The C-motion dataset achieves the lowest DTWD and FD scores overall , with particularly low values on the test set (0.023, 0.095). In contrast, the S-motion dataset shows higher DTWD and FD scores, with slightly better performance on the training dataset compared to test set (0.044, 0.15).

4.4 Validation of the teleoperation setup

The final test validates the teleoperation code's applicability to the physical *DORA* robot in a real-world scenario. The existing codebase from the simulation environment is modified to work with the real robot, allowing for direct control via the *MoCap gloves* and the *AprilTag* system.

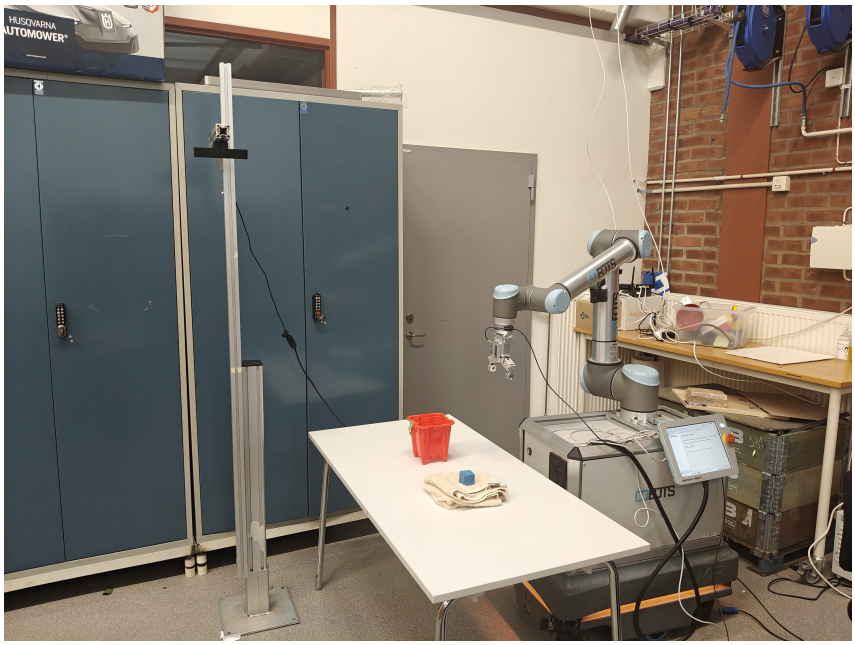
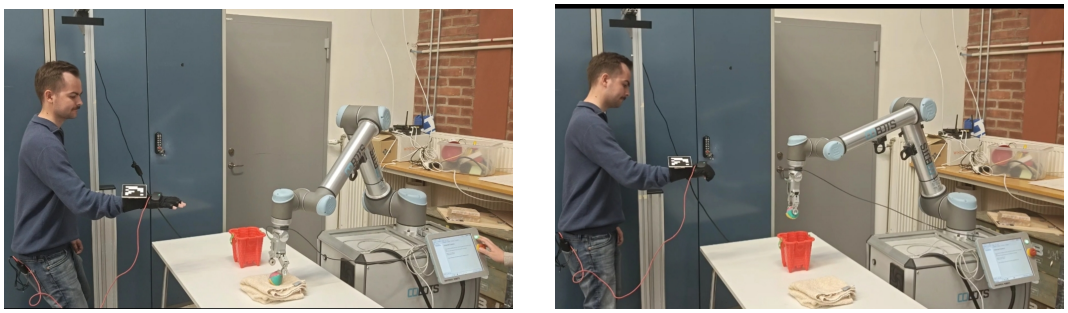


Figure 17: The *DORA* robot setup for the real-world experiment.

As shown in Figure 17, a table is positioned in front of the robot, with a soft ball placed on it. A towel beneath the ball helps prevent potential damage to the robot, while a bucket is placed beside it for the robot to drop the ball into. Additionally, a camera is positioned in front of the table to allow the robot to track and mirror the operator's hand movements.



(a) Robot grasping the ball. **(b)** Robot dropping the ball in the bucket.

Figure 18: The real-world control of the *DORA* robot.

During the tests, as seen in Figure 18, the robot is controlled to grasp the ball and place it into the bucket. Due to the limited range of motion of the user's arm compared to the robot's workspace, a scaling multiplier is applied to the motion data. This technique enables the robot to perform larger movements and reach the necessary positions for task completion.

DORA was able to perform basic grasping and placing motions, and successfully completed two out of five grasping tasks repetitions. Two attempts are aborted due to the robot movement speed either being too fast or too slow, prompting multiple adjustments to the scaling multiplier. A third attempt failed due to a system crash.

5 Discussion

The implementation of *MoCap gloves* and *AprilTag* proved to be valuable tools for providing learning from demonstration data to *DORA*.

During real-world testing using live streaming of the demonstration data information from the *MoCap gloves* and *AprilTags*, *DORA* could mimic the human operator's arm and hand movements seamlessly. However, the *AprilTag* implemented in this project is fixed on a flat surface, limiting *DORA's* ability to perform rotational movements. To further improve the dexterity of *DORA* by enabling rotational tracking, a cube with *AprilTags* placed on multiple faces of a cube could be implemented.

The camera placement also posed limitations, as it frequently lost track of the *AprilTag* when the arm was out of view or far away, leading to interruptions in the tracking data. Additionally, the physical support structure of the camera obstructed the teleoperator, restricting wider arm movements and leading to less natural movement during the experiment.

Furthermore, the ergonomics of the data collection setup for the teleoperator was unsatisfactory, which may have affected the consistency and quality of the recorded movements. The positioning of the equipment and the physical strain during repeated tests contributed to operator discomfort and potential variability in

the input data.

The codebase for the gloves also showed room for improvement, particularly in the real-time performance of the plotting feature. A noticeable delay of approximately 1–2 seconds was observed between the actual hand movement and the corresponding plot update. This delay was not reflected in the terminal output, where the normalized values were updated much more quickly, indicating that the lag originated from the plotting process itself. Moreover, the plotting feature significantly slowed down the PC's performance running the code, making the system less responsive. Disabling the plotting functionality was time-consuming, further reducing the efficiency of the testing process.

Imitation learning was successfully applied using data from the *MoCap gloves* and *AprilTag* system. The resulting evaluation metrics indicate that the model performs well, with generated trajectories closely matching the recorded motions.

For the S-motion dataset, the model achieves lower error on the training data compared to the test data, which is expected since the test scenarios were not seen during training and reflects effective generalization. Interestingly, for the C-motion dataset, both DTWD and FD scores are lower on the test data. This may be due to simpler or more consistent C-motions in the test set, or possibly due to overfitting.

Due to time constraints, the quantity and variability of collected trajectories were limited, which likely affected overall model performance in some simulation cases.

Another limitation caused by time constraints was that the project did not include experiments with LfD models trained on the real robot. As a result, no data collection was conducted with the real robot.

6 Conclusion

This project explores the application of LfD in mobile manipulators to teach new manipulation skills. It demonstrates that *MoCap gloves* and *AprilTag* tracking are valid tools for collecting data and performing teleoperation. in an LfD-based method for the *DORA* robot. However, time constraints limited the ability to evaluate whether the LfD framework improves the dexterity of the *DORA* robot in real-world experiments.

Further improvements can be made by optimizing the ergonomics of the teleoperator, modifying the motion capture gloves' tracking code to eliminate the 1-2 second delay and disabling the 3D plotting functionality.

Simulation results from the preliminary door opening task establishes force frame as oriented from the gripper's perspective and real-world testing confirms the current functionalities of the *DORA* mobile manipulator.

An *imitation learning* model utilized the collected data to model, test and create new trajectories. When comparing the generated trajectories to the demonstrated ones, the average Dynamic Time Warping (DTW) distance of 0.049 and Fréchet distance of 0.208 were obtained. Increasing the quantity and variability of the trajectories will likely improve the quality of the generated trajectories.

The teleoperation setup validated the teleoperation system that was developed in this project. Improvements to this section include tracking rotational hand movement and reorganizing camera setup allowing a greater variation of movement.

Results collected in this project demonstrates that the dexterity of *DORA* can be improved through an LfD-based method. Making *DORA* a more dexterous robot has enabled multiple new use cases for *DORA* to be employed in performing different tasks.

7 Future Work

7.1 Future Work

Due to time constraints, real- world testing with the trajectories created by the *imitation learning* model on the *DORA* robot was not feasible. The project's limited time frame also restricted data collection and limited the variety of recorded movements. To address these limitations in future work, additional time should be allocated towards increasing the quantity of collected data and recording a broader range of movements. Implementing data collection with the real *DORA* robot would also be an essential next step.

Other robotic systems have already been able to use tools similar to *MoCap gloves* and *AprilTags* to mimic a human operator's arm and hand movements [19, 20]. Due to the successful integration of these tools into *DORA*, there is the possibility of practical real-world applications. One example of a real-world implementation of this added functionality in *DORA* is remote manipulation of the robot's arm and gripper to perform tasks where human intervention is too dangerous. To be able to make *DORA* even more useful in these situations, future work could focus on integrating its new capabilities with the ability to remotely manipulate the robot's mobile base.

Future work could explore alternative camera setups to address user limitations, further enhancing the system's functionality.

Furthermore, in future work, *DORA* could benefit from integrating real-time *end-effector* force-torque sensor data into the learning framework. This would be particularly important for contact-rich tasks such as surface-wiping, where force-feedback could enable the robot to adapt its trajectories based on the forces experienced throughout the motion.

References

[1] C. Ling, X. Xing, L. Liu, L. Zhang, X. Bai, and C. Liu, "Design and research of the multifunctional mobile manipulator based on ros," in *2020 IEEE 3rd International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, pp. 86–90, November 2020.

[2] B.-S. Lin, I.-J. Lee, S.-Y. Yang, Y.-C. Lo, J. Lee, and J.-L. Chen, "Design of an inertial-sensor-based data glove for hand function evaluation," *Sensors*, vol. 18, p. 5, May 2018.

[3] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *2011 IEEE International Conference on Robotics and Automation*, (Shanghai, China), pp. 3400–3407, IEEE, May 2011.

[4] A. Barekatain, H. Habibi, and H. Voos, "A practical roadmap to learning from demonstration for robotic manipulators in manufacturing," *Robotics*, vol. 13, p. 100, July 2024.

[5] R. Pérez-Dattari and J. Kober, "Stable motion primitives via imitation and contrastive learning," *arXiv preprint arXiv:2302.10017*, June 2023.

[6] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, pp. 43–49, February 1978.

[7] A. Driemel, S. Har-Peled, and C. Wenk, "Approximating the fréchet distance for realistic curves in near linear time," *arXiv preprint arXiv:1003.0460*, January 2012.

[8] Y. Bekiroglu and A. Tekden, "DORA: Dexterous Robot Assistant," 2025. [Unpublished image], used with permission.

[9] N.A., "Ros/concepts - ros wiki." https://wiki.ros.org/ROS/Concepts, 2025. Accessed: Apr. 07, 2025.

[10] N.A., "Smartgloves - affordable quality finger and hand motion capture." https://www.rokoko.com/products/smartgloves, 2025. Accessed: May 8, 2025.

[11] F. S. Hover and M. S. Triantafyllou, "10.8: What does an inertial measurement unit measure?." https://eng.libretexts.org/Bookshelves/Mechanical_Engineering/System_Design_for_Uncertainty_(Hover_and_Triantafyllou)/10%3A_Vehicle_Inertial_Dynamics/10.8%3A_What_Does_an_Inertial_Measurement_Unit_Measure%3F, 2025. Accessed: May 9, 2025.

[12] N.A., "Rokoko smartgloves – faqs." https://support.rokoko.com/hc/en-us/articles/20912004920593-Rokoko-Smartgloves-FAQs, 2025. Accessed: Apr. 21, 2025.

[13] A. Sagitov, K. Shabalina, L. Sabirova, H. Li, and E. Magid, "Artag, apriltag and caltag fiducial marker systems: Comparison in a presence of partial marker occlusion and rotation," in *Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics*, pp. 182–191, 2017.

[14] M. Mayr, *Learning with Skill-based Robot Systems*. PhD thesis, Dept. of Computer Science, Lund University, Lund, Sweden, 2024. Online.

[15] C. Tsuji, E. Coronado, P. Osorio, and G. Venture, "Adaptive

contact-rich manipulation through few-shot imitation learning with force-torque feedback and pre-trained object representations," *IEEE Robotics and Automation Letters*, vol. 10, pp. 240–247, January 2025.

[16] F. Bramsved, L. Gunterberg-Klase, V. Hendriksen, and E. Schuchert, "DORA – Dexterous Robot Assistant," bachelor's thesis, Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden, 2024. https://odr.chalmers.se/items/54623223-e9dd-499f-a7d8-2eb51001b742.

[17] A. Tekden, M. P. Deisenroth, and Y. Bekiroglu, "Neural field movement primitives for joint modelling of scenes and motions," *arXiv preprint arXiv:2308.05040*, August 2023.

[18] J. Kim and S. Joe, "Enhancing service robotics for cleaning tasks: A deep learning approach integrating learning from demonstration and cross-robot knowledge transfer," in *2024 13th International Conference on Control, Automation and Information Sciences (ICCAIS)*, pp. 1–6, November 2024.

[19] X. Liu, Z. Wang, J. Li, A. Cangelosi, and C. Yang, "Demonstration learning and generalization of robotic motor skills based on wearable motion tracking sensors," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–15, 2023.

[20] A. Rodriguez-Liñan *et al.*, "An approach to acquire path-following skills by industrial robots from human demonstration," *IEEE Access*, vol. 9, pp. 82351–82363, 2021.

[21] Y. Karayiannidis, C. Smith, F. E. Viña, P. Ögren, and D. Kragic, "'Open sesame!' Adaptive Force/Velocity Control for Opening Unknown Doors," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4040–4047, October 2012.

[22] A. Aronson, P.Balaji, M.Berg, T.Casparsson, J.Cheng, J.Hu, and Q.Yang, "Dexterous robot assistant: Door opening with a mobile manipulator.".

[23] N.A., "tf2 - ROS Wiki." https://wiki.ros.org/tf2, 2025. Accessed: May 06, 2025.

[24] Northwestern University, "Chapter 3 Autoplay – Modern Robotics." https://modernrobotics.northwestern.edu/nu-gm-book-resource/chapter-3-autoplay/#department, 2025. Online video, accessed: May 06, 2025.

[25] Stigmatella, "Galilean and spacetime coordinate transformations." https://openverse.org/image/8b6ff952-bc0a-4a1e-8af9-466ca0a9d2fc, 2025. Image, accessed May 6, 2025.

[26] Fireship, "Redis in 100 Seconds." https://www.youtube.com/watch?v=G1rOthIU-uo, 2025. Online video, accessed: May 08, 2025.

[27] j96w, "DexCap: redis_glove_server.py." https://github.com/j96w/DexCap/blob/main/STEP1_collect_data/redis_glove_server.py, 2025. Accessed: Apr. 27, 2025.

[28] RIVerlab, "apriltags_ros." https://github.com/RIVeR-Lab/apriltags_ros, 2025. Accessed: Apr. 22, 2025.

[29] H. Su, S. Liu, B. Zheng, X. Zhou, and K. Zheng, "A survey of trajectory distance measures and performance evaluation," *The VLDB Journal*, vol. 29, pp. 3–32, Jan. 2020.