ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

SCHOOL OF ENGINEERING

DEPARTMENT of
ELECTRICAL, ELECTRONIC AND INFORMATION ENGINEERING
"Guglielmo Marconi"
DEI

MASTER'S DEGREE IN AUTOMATION ENGINEERING

Master Thesis
in
Autonomous and Mobile Robotics M

Imitation Learning From Teleoperation-Based Demonstrations Using Gaussian Mixture Regression For A Dual-Arm Robot

CANDIDATE:
Luca Barbieri

SUPERVISOR:
Prof. Gianluca Palli

CO-SUPERVISOR:
Dr. Roberto Meattini
Alex Pasquali
Kevin Galassi

Academic Year
2022/2023
I Session

Abstract

In recent years, a significant technological advancement has determined the rising of collaborative robotics. While a couple of decades ago robots were primarily used in industrial applications to replace human labour, today they have found their way into various everyday activities in close co-operation with humans. This era is also characterised by the remarkable progress and development of Artificial Intelligence (AI), which enables robots to exhibit greater flexibility and adaptability in dynamic and highly complex environments. Consequently, modern applications allow collaborative robots to work with humans in a shared environment and even interact one with the other. Data-based algorithms using a machine learning approach can be exploited to train robots to perform tasks autonomously. Specifically, robotic manipulators can acquire skills through human demonstration or imitation, using the so-called Programming by Demonstration (PbD) paradigm, via kinesthetic teaching, teleoperation, or an abstract idea of the task. This thesis focuses on employing real-time intuitive teleoperation, used as a tool to enforce PbD. Through this approach, a human operator can intuitively perform uni- or bi-manual telemanipulation. In this thesis work, teleoperation was indeed exploited to provide multiple demonstrations of a given task (or skill) to collect a dataset of trajectories from different initial poses. The task was demonstrated to the dual-arm Baxter robot with the specific goal of grasping a bottle on a table with one arm and pouring its content into a glass held by the other arm. Using Gaussian Mixture Regression (GMR) it was possible to embed the skill described by the different demonstration trajectories into a more general probabilistic description, which also exploits the concept of dynamical systems for the generation of online trajectories, called Dynamical Movement Primitive (DMP), which can autonomously adapt to variable conditions. Finally, the GMR-based PbD approach was implemented and tested by means of grasping and pouring experiments with the real Baxter robot.

Abstract

Negli ultimi anni, si è verificato un notevole avanzamento tecnologico che ha portato alla nascita della robotica collaborativa. Mentre in passato i robot venivano impiegati principalmente nell'ambito dell'industria per sostituire la mano d'opera umana, oggi essi hanno trovato applicazione in una vasta gamma di attività quotidiane, collaborando strettamente con gli individui. Quest'era è altresì caratterizzata da considerevoli progressi e sviluppi nel campo dell'Intelligenza Artificiale (IA), che consentono ai robot di dimostrare una maggiore flessibilità e adattabilità in ambienti dinamici e complessi. Di conseguenza, le moderne applicazioni consentono agli esseri umani di operare in un ambiente condiviso con robot collaborativi, e persino di interagire l'uno con l'altro. Gli algoritmi basati sui dati, che utilizzano un approccio di machine learning, possono essere sfruttati per addestrare i robot a svolgere compiti in modo autonomo. In particolare, i manipolatori robotici possono acquisire abilità tramite dimostrazione o imitazione umana, mediante un paradigma noto come programmazione per dimostrazione. Tale approccio si avvale di modalità di insegnamento cinestetico, teleoperazione o di una concezione astratta del compito. Questa tesi si focalizza sull'utilizzo della teleoperazione intuitiva in tempo reale, impiegata come strumento per raggiungere la programmazione per dimostrazione. Mediante tale approccio, un operatore umano può manipolare in modo intuitivo il sistema robotico tramite telemanipolazione, sia con una sola mano che con entrambe. Nella presente tesi, la teleoperazione è stata impiegata per fornire multiple dimostrazioni di uno specifico compito, al fine di raccogliere un insieme di dati riguardanti le traiettorie da diverse posizioni iniziali. In particolare, il compito dimostrato consiste nell'afferrare una bottiglia posta sopra un tavolo con un braccio del robot e versarne il contenuto in un bicchiere tenuto dall'altro braccio. Attraverso l'utilizzo della Gaussian Mixture Regression (GMR), è stato possibile incorporare l'abilità espressa dalle diverse traiettorie dimostrative in una descrizione probabilistica più generale, che si avvale anche del concetto di movimenti dinamici per la generazione online di traiettorie. Tale concetto è noto come Dynamical Movement Primitive (DMP) e permette un adattamento autonomo a condizioni variabili. Infine, questo approccio è stato implementato e testato attraverso una serie di esperimenti di presa e versamento utilizzando il robot Baxter.

Contents

List of Figures

Chapter 1

Introduction

Until a decade ago, robots were merely used to indicate an industrial automatic machine able to perform heavy and repetitive jobs. Nowadays, with the rise of Artificial intelligence (AI), robots have become more flexible and intelligent, providing the possibility of even working in contact with humans. Collaborative Robots, also called Cobots, are anthropomorphic robots equipped with a precise set of sensors, such as force and proximity sensors, which make them able to interact with humans safely. Technology development has made possible not only human replacement, but also the usage of robots in Activities of Daily Living (ADL), collaborating with humans in various environments and sectors, such as industries, working places, hospitals, houses, restaurants, and so on. State-of-the-art applications permit collaborative robots not only to work near humans but also to collaborate with them and be controlled in real-time by humans remotely. For example, robots can be teleoperated to perform remote jobs, as in the case of robotic surgery, or they can be used to navigate across dangerous paths for exploration or research and rescue tasks. In the prosthetic and rehabilitation field, they can be utilised to help humans in everyday life with exoskeletons to assist paralysed people or with prosthetic hands or feet in case of amputated people. Differently, in industrial applications, robots have always been designed to operate via model-based control. Starting with complete knowledge of the dynamical model of the robot, the objective is to compute a control law to make it follow a desired trajectory. Even if this control technique is very efficient and robust to uncertainty and disturbances, it may be not sufficient in everyday applications, since it can require a complete design change even for small changes in the task specification. To overcome this problem, research is going towards the exploration of data-based algorithms using a Machine Learning approach, which allows robots to learn by demonstration or human imitation. A possible data-based approach is the programming-by-demonstration

PbD also known as Learning from Demonstration.

This framework consists of the initial collection of a set of data *training set* obtained from the demonstration of the assignment to perform and replicate by the robot. The demonstration can happen via Kinesthetic teaching, in which the human moves the robot along a particular trajectory, via primary-secondary teleoperation, via human imitation using a tracking system to recognise movements of the human limbs, or via an abstract description of the task. Then, from the demonstration, an ML-based algorithm performs the so-called Learning Control, extracting a feasible and robust control law. In such a way, the robot is able to learn and perform that task with complete autonomy.

1 IntelliMan Project

This thesis work is related to the activity of the European project IntelliMan funded by the European Union and coordinated by the University of Bologna. IntelliMan is concentrating on the question of “How a robot can efficiently learn to manipulate in a purposeful and highly performant way” [1]. To achieve that, IntelliMan is operating on designing and controlling AI-Powered Manipulation systems for advanced robotic services in the manufacturing and prosthetic field. The proposed solutions are characterized by persistent learning capabilities regarding the interaction and manipulation of the surrounding in a purposeful and efficient way. The robotic system will be capable of learning unique manipulation skills from human demonstration or from an abstract description of a task suitable for high-level planning, making it capable of performing the task with complete autonomy and also detecting any possible failure.
Citing the official website [1], IntelliMan has set the following specific objectives

- Platform-independent knowledge transfer between different domains and systems;
- Manipulation task structure and hierarchy learning from sensory measures and human cues, as well as new tasks planning;
- Reduction of training samples need;
- Guaranteed performance, safety and fault detection.

2 Learning From Demonstration And Gesture Reproduction Via Intuitive Teleoperation Of A Humanoid Bimanual Robot

This thesis project will propose two different setups for the interactive telemanipulation of a humanoid bimanual robot and for the development of basic actions and skills (e.g grasping). The solution developed will be based on ROS, an open-source operating system for robotics, and the Vicon Tracker, a tracking system based on infrared sensors, to track the position and orientation in the space of the human upper limbs. In particular, a Baxter robot made by Rethink Robotics will be controlled. Lastly, teleoperation will be used as a tool to achieve Learning from Demonstration, a technique for extracting a control law from the demonstration of a task performed by a human. The robotic system will be then able to learn how to replicate that task in an autonomous and robust manner, insensitively to disturbances of the environment and with possible different starting or arrival points. This concept inspired multiple research during the last decades, in particular, Calinon et al. [2], propose several methods based on the generation of a Dynamical Movement Primitive (DMP), a unidimensional trajectory that represents the shape of the movement. This can be used as a reference velocity trajectory computed via a Machine learning algorithm. For the computation of the DMP, in this thesis was used the Gaussian Mixture Regression (GMR), a stochastic method used to train a Gaussian Mixture Model (GMM) from a collected Training set, see Figure 1.1. A spring-damper system was then used to guarantee the asymptotic convergence of the trajectory to the target point. In such a way, the human operator is able to teach the robot a new task using teleoperation showing it the right movements to perform. All the used algorithms, such as GMM and GMR will be deeply explained in Chapter 5.

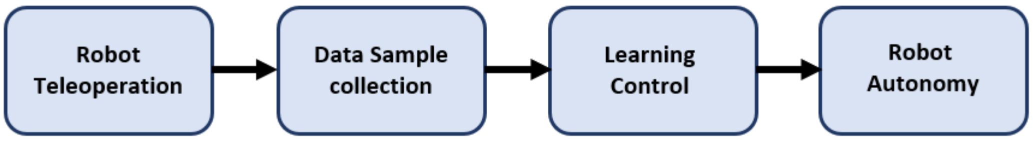


Figure 1.1: Robot learning by demonstration is divided into sub-tasks. First, the robot is teleoperated to make it perform a task N times and to collect a training set. After that, Learning control extracts a law from the demonstrations and allows the robot to learn the performing of that task autonomy.

Chapter 2

Robot Set-Up Description

This chapter will explain the setup used throughout the thesis, focusing on the used components, such as:

- Robot Operating System (ROS)
- Rethink Robotics Baxter Robot
- Vicon tracking system

Lastly, the communications across the entire experimental environment will be explained. In particular, the methods to control Baxter via ROS and the way to track the movement of the human upper limbs using the Vicon system for the robot teleoperation.

1 ROS

ROS [3], 'Robot Operating System', is an open-source meta-operating system designed for robotic applications. It provides the same functionalities and services as a standard Operating System, such as hardware abstraction, device driver control, inter-process communication, application management, and other commonly used functions. The idea of ROS is to generate a high-level control that can communicate with the lower level of the robot. The key idea of ROS is the distributed framework of processes that run concurrently. Each process, called "node", works independently from the others and represents a different module. The nodes are also capable of sharing information through a communication system based on the concept of "topic" and "messages".

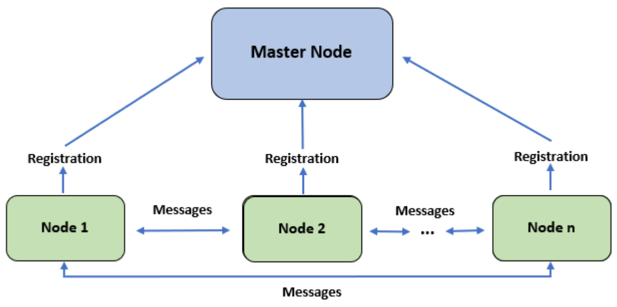


Figure 2.1: ROS communication scheme. Nodes are registered to a central master and can communicate with each other through messages, services and actions

1.1 Nodes

In ROS each executed program is denoted as a Node and it can be seen as a single entity running concurrently in the system. Each node can communicate with the others through a server-client communication architecture, in which each node can work as a client but also as a server. as illustrated in figure 2.1 Communication between nodes can happen in two three ways:

- ROS topics: publish/subscribe communication.
- ROS services: a node directly invokes another node service or action.
- ROS actions

A ROS−based robot control system is usually composed of many nodes, each one designed to have a short and specific task, the result of which should be exchanged with other nodes. The consequence is a network of data elaboration with a complex graph−like structure capable of solving demanding problems. This particular architecture provides many benefits. The biggest one is fault tolerance, as each node is an isolated part of the system, and reduces code complexity compared to monolithic systems.

1.2 ROS Topics

ROS topics can be seen as ”mailboxes” used by ROS nodes to exchange messages in the publish/subscribe communication. One ROS node would publish the ROS message to a certain ROS topic, while another ROS node would subscribe to that ROS topic and acquire the ROS message sent. Ros nodes can publish or subscribe to any Ros topics since they are all characterized by a unique name.

ROS Messages are described as *.txt* files inside a specific folder called *msg* under the ROS package folder structure. Each ROS message is described with a data structure that contains only primitive types (integers, floats, or booleans) or arrays of these types. A ROS message can also include another ROS message. creating a more complex and longer message. Since this type of communication is not direct, the topic is the intermediary between two nodes, the sender and the receiver keep their anonymity.

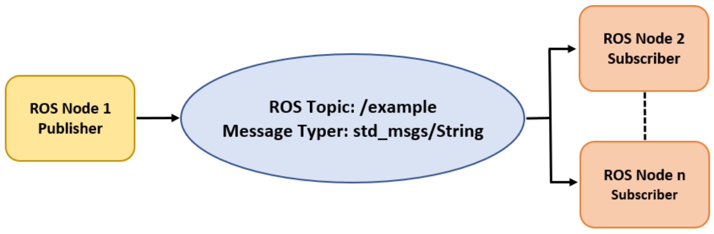


Figure 2.2: ROS Message communication scheme through a ROS Topic

1.3 Services

ROS services are another method for messages exchange as direct communication between nodes. By using the ROS services, the publish/subscribe mechanism is avoided and nodes can send requests and replies to each other directly using the so-called *srv*. Also in this case, the services have the target folder named *srv* in which a custom service can be defined. Each *srv* is a *.txt* file containing the request sent from the client node and the response from the Server node. Since the ROS services are a form of direct communication, they improve the performance of the system, however, they reduce the system′s decoupling.

1.4 Action

In the case that ROS services are operations requiring a certain amount of time to be executed, ROS provides Actions. For such a reason, ROS actions can be seen as particular types of ROS services. Those are useful if an event occurs before the communication of the response from the server takes place, changing the system condition. Therefore the execution of the service may be no longer needed. To overcome this problem, an action message is composed of three parts: Goal, Feedback and Result. The Goal, sent to the Action-

Server by an ActionClient, is a part of the message containing information about the objective to execute. E.g. a position or a series of parameters. During the execution, the ActionServer is able to tell an ActionClient about the incremental progress of the goal through some feedback. At last, the result is sent back to the ActionClient providing information about the correct execution of the task or some other feedback.

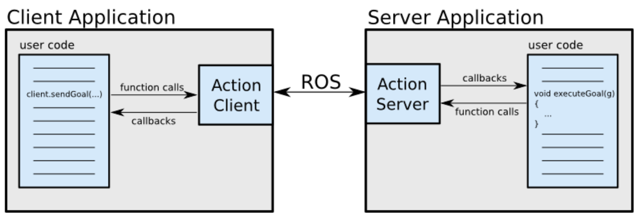


Figure 2.3: ROS Action working scheme. Image taken from [3]

1.5 Rviz

RVIZ is a ROS graphic interface that shows the robot model to the user. It uses data from the Robot sensors to create a representation as accurately as possible of how the robot is moving and what it is seeing or detecting [3]. In robotic related work, Rviz is useful to show the user the so-called TFs. As illustrated in Figure 2.4, each robot element, (E. g.links, end-effectors, base) can be represented as a reference frame. A ROS TF is a transformation between a fixed reference frame, called *Base*, and the considered object frame, in terms of position and orientation.

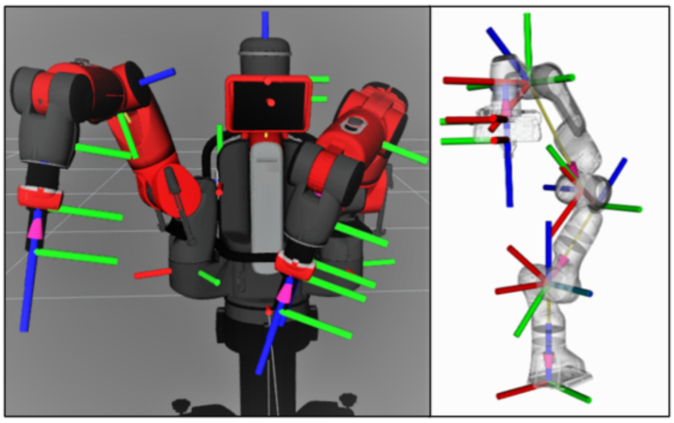


Figure 2.4: Robot Baxter TF visualization on RVIZ (left) Robot Panda TF visualization (right)

2 Baxter Robot

Baxter is a dual-arm anthropomorphic robot designed by Rethink Robotics as a research collaborative robot[4]. Each arm is characterized by 7 Degrees of Freedom as joint angles, as illustrated in figure 2.5, called S_0 and S_1 for the shoulder, E_0 and E_1 for the elbow and W_0, W_1 and W_2 for the wrist. Consequently, the End-Effector will be able to reach every position and orientation within the Robot workspace. The seventh degree of freedom allows the robot to extend its positioning capability and eventually perform motion avoidance algorithms. Another important characteristic of a collaborative Robot is the capability of operating in a non-isolated place. Due to a force sensor placed in each joint, the robot is able to detect contact with any other object or person and then stop the execution of the operation, avoiding hazards or dangerous situations.

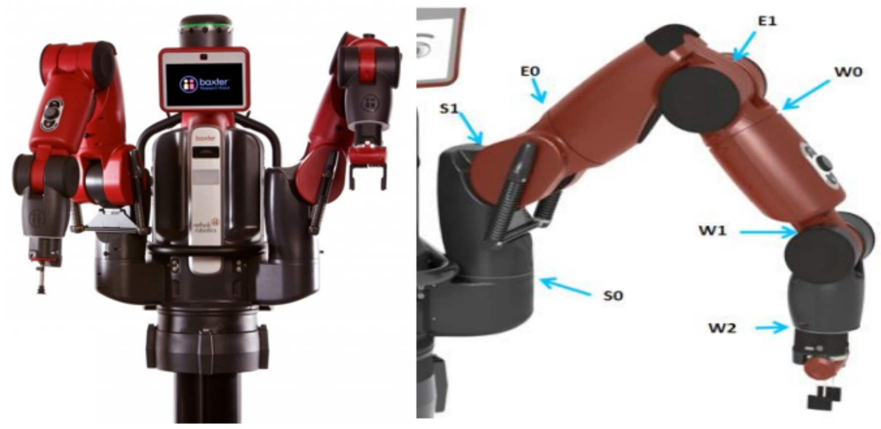


Figure 2.5: Baxter Humanoid Robot (Left) and Baxter's Arm and Joint Designations (Right).

2.1 Baxter Robot - Ros Interface

Taking as a reference the official website of the Baxter designer Rethink robotics [4], the interface between ROS and the Robot is immediate. Inside the Rethink Robot, there is an *Intera SDK* that provides a software interface allowing researchers to develop custom applications to run on the robot. As shown in Figure 2.6, the robot provides a stand-alone ROS Master to which any development workstation can connect via the various ROS APIs, simply through an Ethernet interconnection.

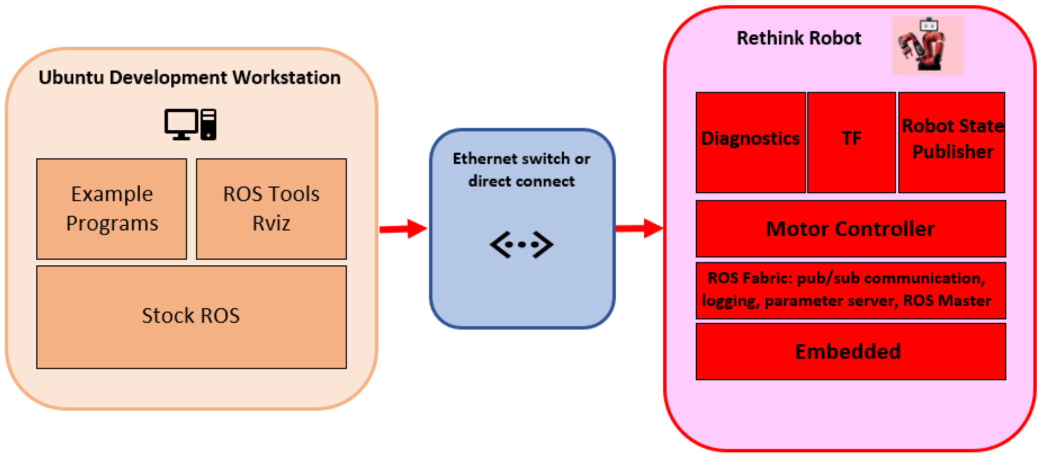

Ubuntu Development Workstation
Example Programs
ROS Tools Rviz
Stock ROS
Ethernet switch or direct connect
Rethink Robot
Diagnostics
TF
Robot State Publisher
Motor Controller
ROS Fabric: pub/sub communication, logging, parameter server, ROS Master
Embedded

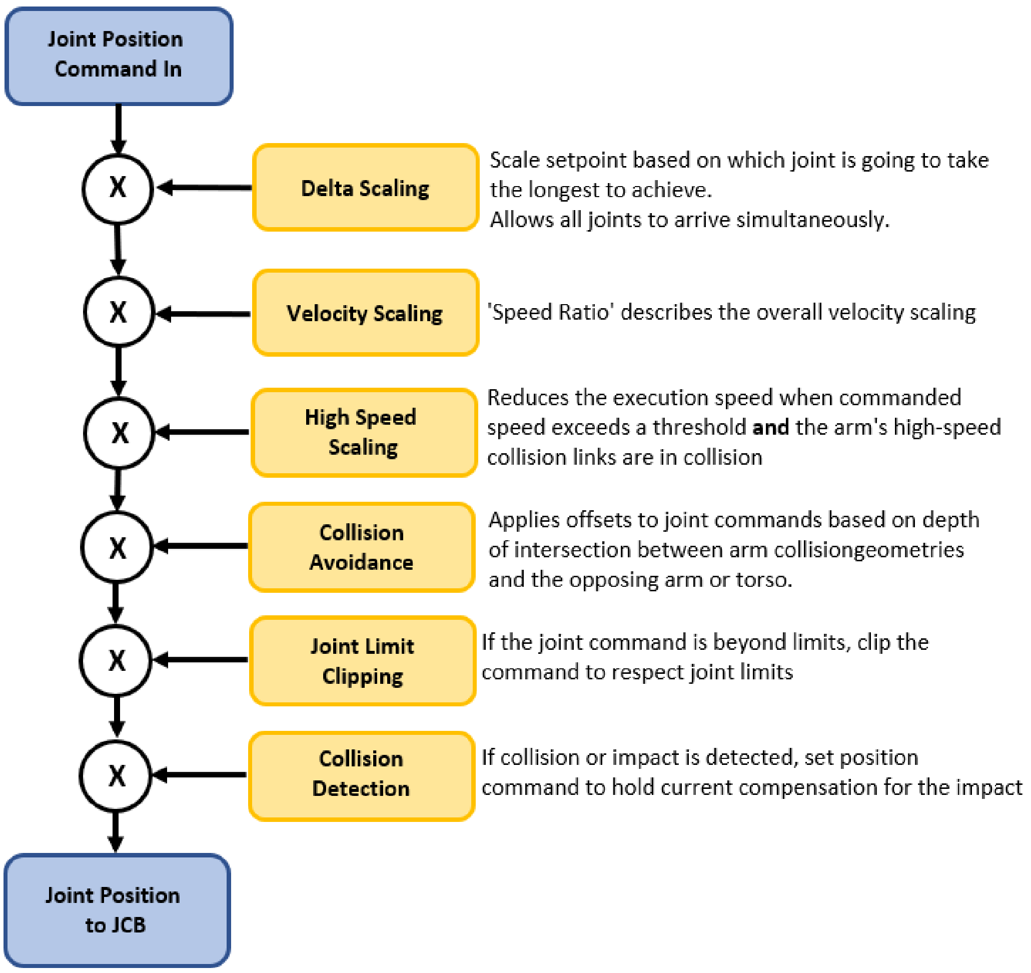
Figure 2.6: Rethink Robot ROS interface.

In such a way, from the Robot State Publisher, it is possible to access all the robot state data, such as joint angles, joint efforts, joint velocities, ROS transform system (TF) and so on. It is also possible to control each robot arm in four different modes:

- **Joint Position Control:** At every time step, it is specified the joint angles the robot should achieve, as a list of seven values, resulting in a full description of the arm configuration. This joint command is then subscribed to the Motor Controller, which processes the command ensuring safety (collision avoidance and detection) and expected behaviour by making the modifications illustrated in figure 2.7.

- **Raw Joint Position Control:** It is an advanced Control Mode. It is a control type similar to the Joint position one, but it can be more dangerous as it does not include Collision Avoidance and velocity scaling algorithm.

- **Joint Velocity Control:** At every time step, it is specified the joint velocities the robot should achieve, as a list of seven commanded joint velocities, which is then subscribed to the Motor Controller. As in the Joint Position Control, the motor controller fully processes this joint velocity command (applying collision avoidance, velocity clipping and detection).



Joint Position Command In
X
Delta Scaling
Scale setpoint based on which joint is going to take the longest to achieve.
Allows all joints to arrive simultaneously.
Velocity Scaling
'Speed Ratio' describes the overall velocity scaling
High Speed Scaling
Reduces the execution speed when commanded speed exceeds a threshold **and** the arm's high-speed collision links are in collision
Collision Avoidance
Applies offsets to joint commands based on depth of intersection between arm collisiongeometries and the opposing arm or torso.
Joint Limit Clipping
If the joint command is beyond limits, clip the command to respect joint limits
Collision Detection
If collision or impact is detected, set position command to hold current compensation for the impact
Joint Position to JCB

Figure 2.7: Processing executed by the motor controller to ensure safety in the Joint Position Control. Image taken from [4]

- **Joint Torque Control:** At every time step, it is specified the joint torques the robot should achieve, as a list of seven commanded joint torques, which is then subscribed to the Motor Controller. As it is shown in Figure 2.8, are performed only the gravity/spring compensation, the collision avoidance and the torque scaling algorithms, therefore this type of control can be dangerous.

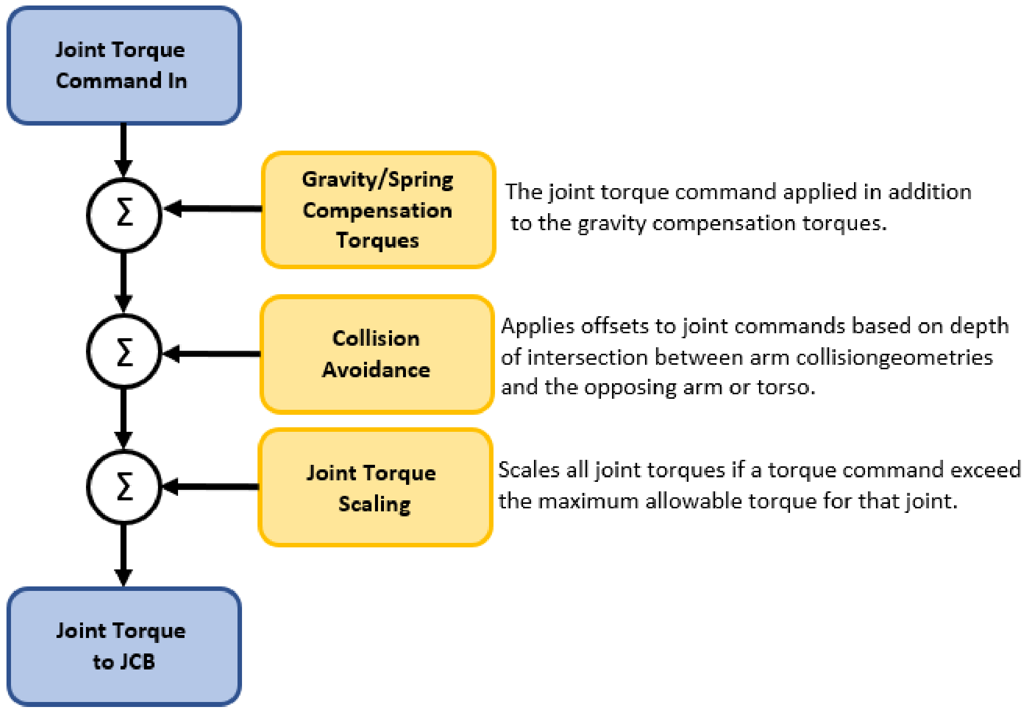

Joint Torque Command In
Σ
Gravity/Spring Compensation Torques
The joint torque command applied in addition to the gravity compensation torques.
Σ
Collision Avoidance
Applies offsets to joint commands based on depth of intersection between arm collisiongeometries and the opposing arm or torso.
Σ
Joint Torque Scaling
Scales all joint torques if a torque command exceed the maximum allowable torque for that joint.
Joint Torque to JCB

Figure 2.8: Processing executed by the motor controller in the Joint Torque Control. Image taken from [4]

3 Vicon Tracker

Vicon Tracker [5] is a motion capture system to track objects in space. The system is composed by multiple infrared camera capable to identify the position and orientation with respect to a base frame. Each object is composed by of three or more reflective passive markers, placed accordingly to a unique geometry, used to avoid mis-detection or confusion among them. Figure 2.9 shows an example of a tracked object, composed by seven reflective passive markers. The software is able to detect the position in space of each one of them, and therefore compute the position and orientation in space of its reference frame, with respect to the Vicon base frame.

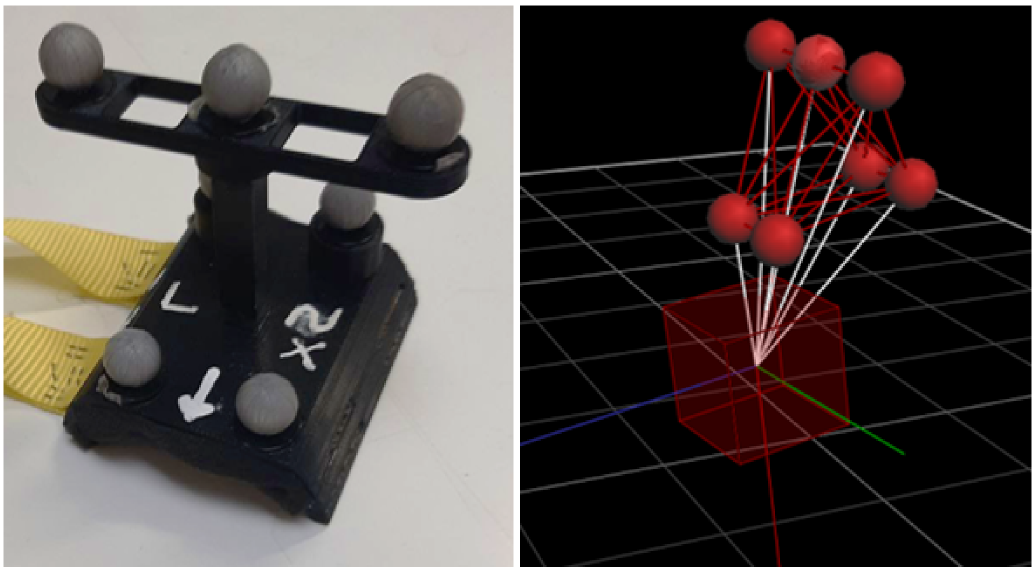


Figure 2.9: Example object composed of seven markers (Left) and same object tracked by the Vicon system (Right).

After an initial calibration of the cameras through some movements of the so-called calibration wand, explained in [5], all the objects are detected and tracked by eight Vicon Bonita cameras sampling at 100 Hz.

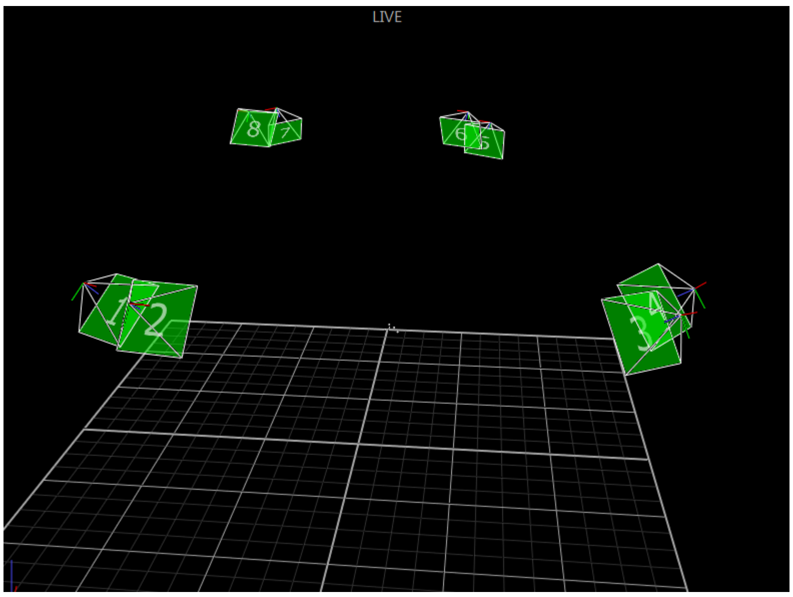


Figure 2.10: Vicon tracking cameras arrangement

3.1 Vicon Ros Bridge

To read the tracked data from Vicon it was created a ROS node that acts as a bridge and makes the tracked objects available as transformations between reference frames on ROS. ROS TF is the transform library, presented in [6], which lets the user keep track of multiple coordinate frames over time. Therefore, it is possible to obtain the position and orientation of any object, represented as a reference frame in the 3D space, which is expressed by a Python list in the following way:

$$POSE_{obj} = [x,\ y,\ z,\ q_x,\ q_y,\ q_z,\ q_w] \tag{2.1}$$

Where $x,\ y,\ z$ is the cartesian position, while $q_x,\ q_y,\ q_z,\ q_w$ the quaternion orientation, with respect to a fixed reference frame in the exact middle of the tracking room, called *vicon* of pose $[0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1]$

To obtain the relative TF between two objects it was implemented a ROS function that simply makes the difference between their TFs. In such a way it is possible to acquire the position and orientation coordinates of an object O_2 written in another object frame O_1.

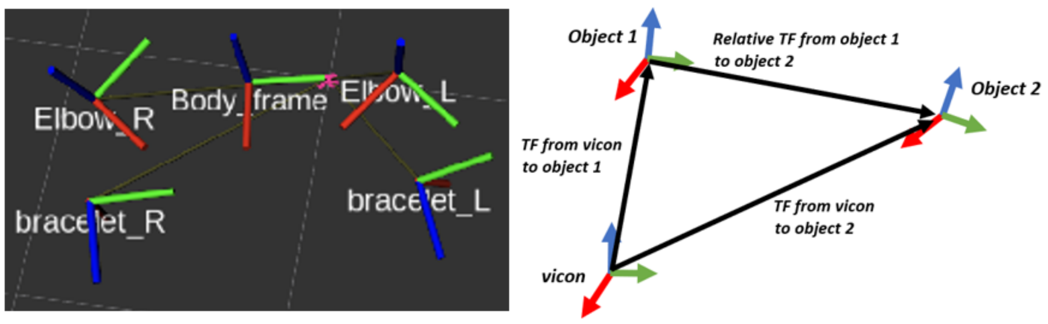

Elbow_R
Body_frame
Elbow_L
bracelet_R
bracelet_L
Object 1
Relative TF from object 1 to object 2
Object 2
TF from vicon to object 1
TF from vicon to object 2
vicon

Figure 2.11: TFs visualization on Rviz of the objects tracked from Vicon (left) and ROS relative transform between two different reference frames (right)

Chapter 3

Experimental Set Up

The teleoperation objective of this thesis work is the map between the movements of human's upper limbs and the robotic system's upper limbs. In such a way, the human operator would be able to intuitively control the Baxter robot in real-time by only moving the limbs, using visual feedback. To obtain this result, the human limbs' movements must be tracked. This chapter will present the experimental setup for the teleoperation of the robot arms. In particular, there will be explained the considered mathematical model and, after an initial calibration, the way it was exploited for the mapping between the human and the Robot's upper limbs. The teleoperation technique will be explained in chapter 4.

1 Mathematical Model Of The Human Arm

An important quality of human-Robot upper limb telemanipulation is the mapping between the human degrees of freedom and the robots. It may not be sufficient to make the End-Effector of the robot follow the trajectory in space of the human hand. In several applications, it is important to be coherent with the human upper limb positioning in space, paying attention not only to the hand but also to the elbow positioning. Consequently, it is necessary to analyse and describe the mathematical model of a human arm. Compared to a robotic arm, humans are characterized by a higher number of degrees of freedom. In literature exists multiple models that try to approximate human behaviour using a lower number of degrees of freedom. As represented in Figure 3.1 the human arm can be modelled, and thus mathematically described, according to [7], as seven degrees of freedom mechanism, in the following way.

- A spherical joint for the shoulder, composed by θ_1, θ_2 and θ_3

- A rotational joint for the wrist, composed by θ_4.
- A spherical joint for the wrist, composed by θ_5, θ_6 and θ_7.
- Arm and forearm length, L1 and L2

This simplified model [7] neglects scapula movement and forearm pronation, but it can be considered adequate for many applications. In fact, these two other degrees of freedom are relatively small movements and are difficult to replicate by a robotic arm.

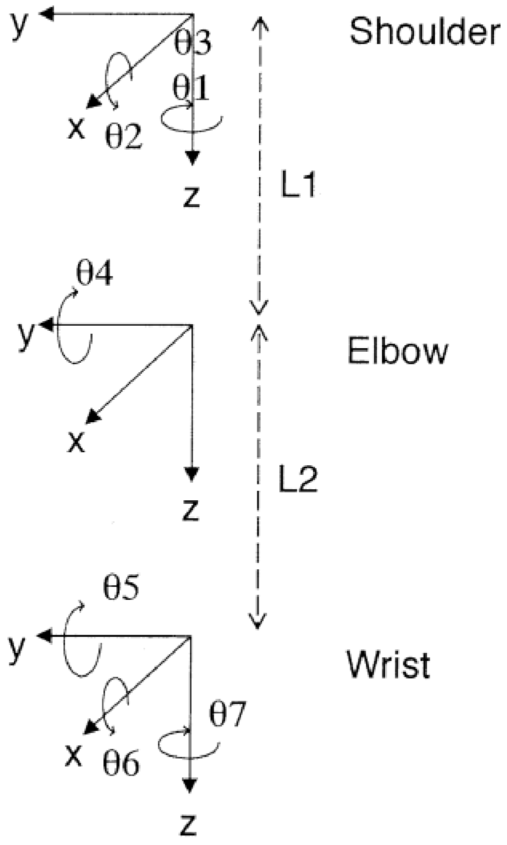


Figure 3.1: A simplified model of the human arm as seven degrees of freedom mechanism. Image taken from [7]

To use the presented model it can be useful to make the same consideration as in [8], in which the authors explain how the shoulder and the elbow angles,θ_1, θ_2, θ_3 and θ_4, provide the cartesian position in space of the human wrist, while θ_5, θ_6, θ_7 the hand orientation. As in the previous case, this is a simplification because, anatomically speaking, the wrist "rolling" angle, θ_7, is driven by the biceps and the triceps. Therefore, it is a degree of freedom belonging to the elbow.

2 Tracking Experimental Setup

To perform a complete tracking of the human upper limbs, five different objects have been created and each one is placed on a different human link. The first one is called *Body Frame* and, as illustrated in Figure 3.2, is placed on the human thorax guaranteeing a fixed reference frame. The remaining objects have the purpose of tracking the movements of the human upper limbs. Two of them are placed on the edge of the arm, just above the elbow, while the remaining two are on the edge of the forearm, in proximity to the wrist. Since they are placed on each link of the upper limb, they permit the tracking of the relative motion with respect to the base frame and the previous link. In Chapter 4, the manner in which this setup is utilized to compute the cartesian position and orientation of the human wrist in space will be explained alongh with the method to extract the human joint angles.

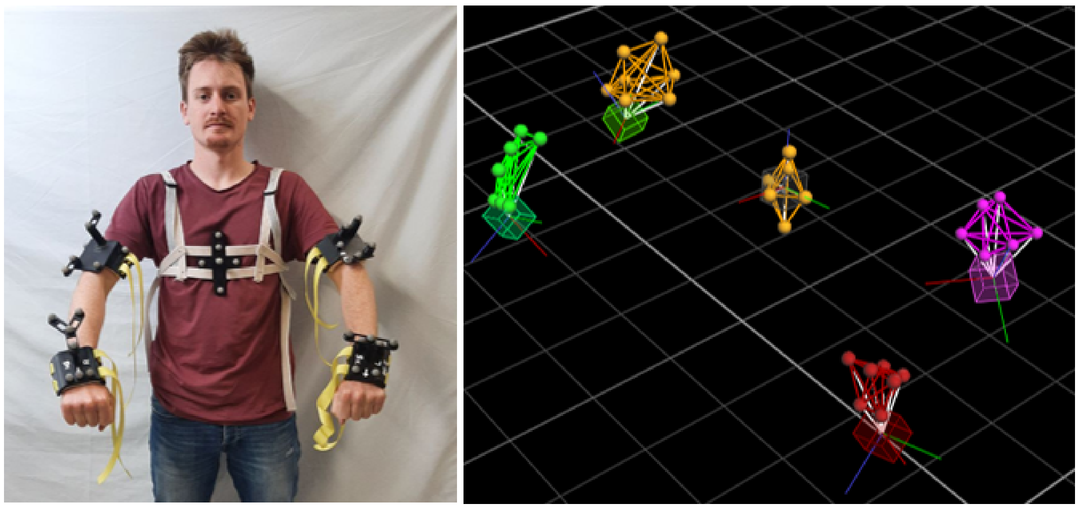


Figure 3.2: Placement of the objects on the human body (Left) and their tracking on Vicon software (right)

2.1 Digital Signal Processing On Vicon Measurement

In dealing with tracked objects by cameras, one of the main problems is the faulty detection of the pose of the object, due to disturbances in the environment. In the case of multiple tracked objects, there can be also confusion between two different objects, leading to an inaccurate stream of data.

To deal with these problems and obtain a better detection, the design of each single object must be unique, and with a redundant number of markers.

To avoid faulty detection of the pose of the object, the stream of data from the Vicon was filtered with a basic algorithm of digital signal processing. It is important to notice that faulty detection provides a non-continuous trajectory, which can lead to jumps in position and orientation. To cancel out any possible detection error, every element of the list from the TF explained in the previous section, is compared with the respective element of the previously sampled TF. If the difference between them is greater than a certain threshold, 10 *cm* for the positions and 0.1 *rad* for the orientations, the new data from Vicon is eliminated and substituted with the previously sampled data. Using this strategy, if a tracking error happens, the robot simply remains in a safe position. To make the trajectory smoother, a simple moving average (SMA) filter of five data points has been implemented, in the following way.

Let us consider a stream of data from Vicon of n data points in the form of Equation 2.1, as $(pose_1, pose_2, ..., pose_n)$. The SMA filter was performed for each coordinate of the TF pose, therefore considering a stream of n coordinate points $(p_1, p_2, ..., p_n)$.

The simple moving average filter is defined as:

$$SMA_k = \frac{p_{n-k+1} + p_{n-k+2} + ... + p_n}{k}$$

Therefore, considering $k = 5$:

$$SMA_5 = \frac{p_{n-4} + p_{n-3} + p_{n-2} + p_{n-1} + p_n}{5}$$

Since the filter outcome is an average value of K elements, to obtain a feasible orientation it is mandated to normalize the quaternions. In fact, these quantities are defined in [9] as a four-parameter representation $Q = (\eta, \epsilon)$, where:

$$\begin{aligned} \eta &= cos\frac{\theta}{2} \\ \epsilon &= cos\frac{\theta}{2}r \end{aligned} \tag{3.1}$$

$\eta \in \mathbb{R}$ is called the scalar part of the quaternion while $\epsilon = [\epsilon_x \ \epsilon_y \ \epsilon_z]^T \in \mathbb{R}^3$ is called the vector part of the quaternion. They are constrained by the following condition [9]:

$$\eta^2 + \epsilon_x^2 + \epsilon_y^2 + \epsilon_z^2 = 1 \tag{3.2}$$

Hence the name unit quaternion. Following the notation of Equation 2.1, calling $(q_x, \ q_y, \ q_z, \ q_w)$ the quaternion obtained from the SMA filter, the normalization is performed as follows.

$$\begin{aligned}
\epsilon_x &= \frac{q_x}{\sqrt{q_x^2 + q_y^2 + q_z^2 + q_w^2}} \\
\epsilon_y &= \frac{q_y}{\sqrt{q_x^2 + q_y^2 + q_z^2 + q_w^2}} \\
\epsilon_z &= \frac{q_z}{\sqrt{q_x^2 + q_y^2 + q_z^2 + q_w^2}} \\
\eta &= \frac{q_w}{\sqrt{q_x^2 + q_y^2 + q_z^2 + q_w^2}}
\end{aligned}$$

3 Initial Calibration

To develop user-friendly teleoperation available for everyone, an initial calibration is necessary to find the cartesian position of the shoulder and any possible misalignments between the elbow frame and the shoulder frame.

3.1 Cartesian Position Of The Shoulder

For the cartesian position of the shoulder, the user was asked to do some calibration movements with both arms, while the Vicon was tracking the relative position between the Body frame and the elbow objects. In particular, starting from the initial position with the upper limb parallel to the ground, Figure 3.3 (right), and the final position with the upper limb down (left).

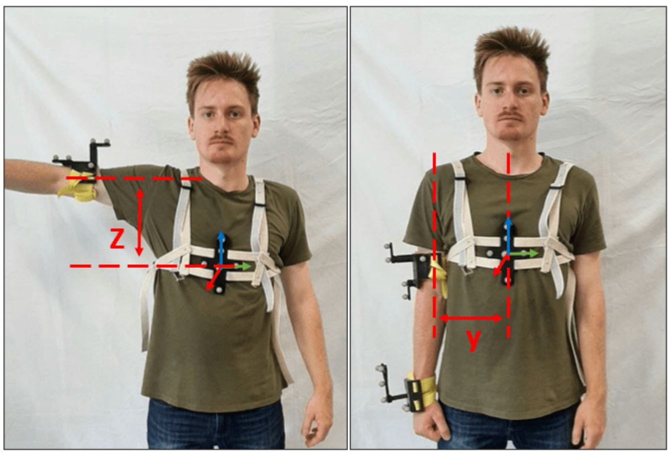


Figure 3.3: Shoulder calibration of a single arm. The operator alternates these two calibration positions to compute the distance between the base and the shoulder reference frame.

In such a way, by storing all data during the calibration time, plotted in figure 3.4, it is possible to find the following positions:

- X position as the average value of all the samples (red)
- Y position as the minimum value of all the samples (green)
- Z position as the maximum value of all the samples (blue)

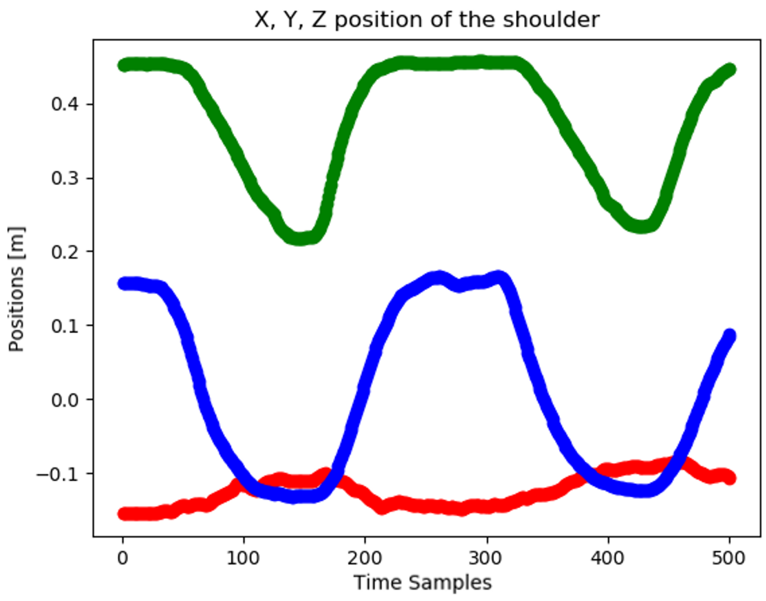


Figure 3.4: X, Y and Z distance between the base and the elbow reference frame. The X position is the average value of all the samples (red), the Y position is the minimum value of all the samples (green) and the Z position is the maximum value of all the samples (blue).

3.2 Elbow - Wrist Misalignment

For the relative orientation of the wrist frame with respect to the elbow frame, the user was asked to keep the arms steady, as in Figure 3.5, while the Vicon was tracking the relative orientation of the two respective objects.

Similarly to the previous case, the relative orientation was computed as the average value of all the samples for each quaternion orientation. After that, as explained in the previous section, the computation of the normalization of the quaternions is necessary. With those measurements, during the teleoperation, any possible misalignments were cancelled out merely by performing a homogeneous transformation as the inverse of the detected rotations from the calibration.

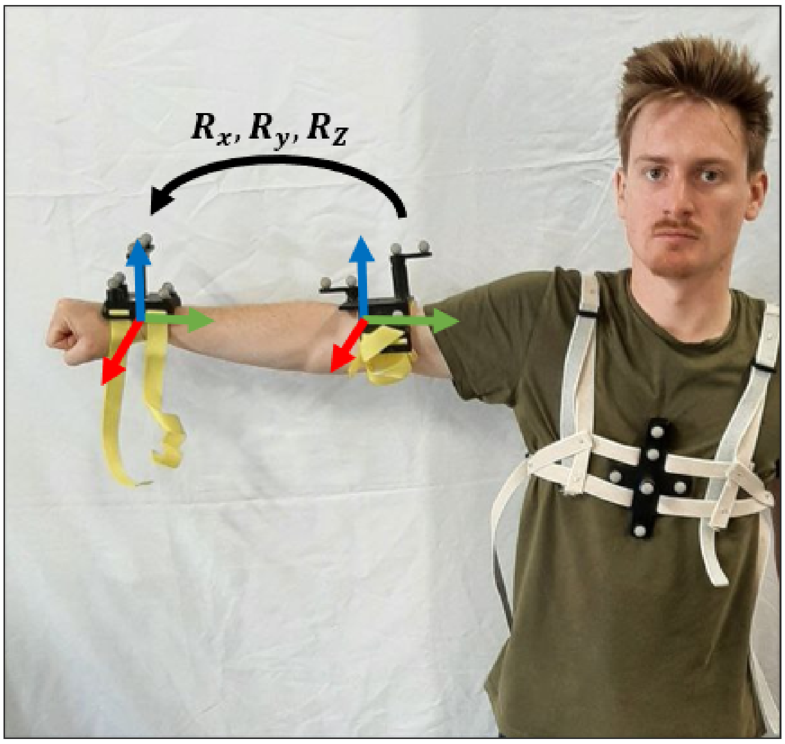


Figure 3.5: Elbow calibration of a single arm. The operator keeps the calibration position to compute any possible misalignment between the elbow and the wrist reference frame.

Chapter 4

Real-Time Dual-Arm Teleoperation

Teleoperation is generally used to indicate the remote control of a machine, called *secondary*, by a human operator who interacts with a local machine called *primary*. These elements exchange information through a communication channel, as illustrated in Figure 4.2. In this context, the operator must be able to command the machine by receiving feedback about the secondary's actions. It is important to guarantee some performance specifications such as a stable connection characterized by a delay as low as possible. Teleoperation systems are widely used in robotic applications. In industrial scenarios whether the task is repetitive or strenuous. In the case of robotic surgery, for example, the objective is to ensure better precision and eliminate any possible human generated from hand tremors, as in the case of the Da Vinci robot [10]. In the case the environment is dangerous for human beings, several robotic applications can be used for research and rescue tasks, as in Figure 4.1 using underwater manipulation or robotic dogs [11]. To enable the adequate operation of robots through humans, appropriate human-robot interfaces (HRIs) must be used, the most common example of interface available in literature are teach-pendants, joysticks and haptic devices.

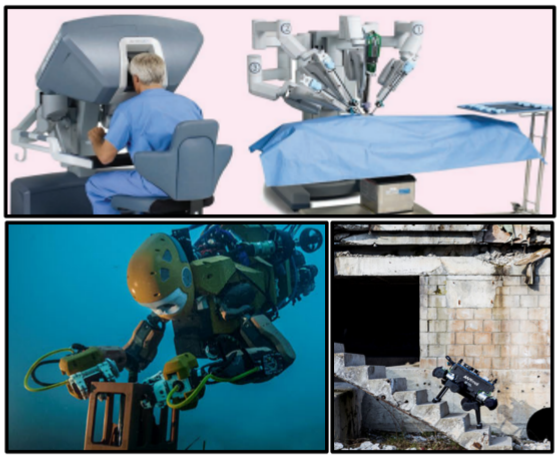


Figure 4.1: Some examples of robotic applications in which human teleoperation is used to achieve more profitable results. Da Vinci surgical robot (Up), an underwater robot performing research tasks (Down Left) and a robot dog (Down Right) for rescue tasks in a dangerous environment. Images taken from [10] and [11] .

Nonetheless, most of the existing teleoperation interfaces do not permit intuitive use and require unique expertise. It is desirable that the robotic system can be controlled naturally and intuitively by humans, providing a minimum dissimilarity between human operator control and robot execution. This thesis has implemented a real-time upper limb telemanipulation, transferring human motion into a humanoid robot.

Direct human-robot motion imitation may result impossible due to differences in their kinematics. To overcome this problem, two existing approaches of humanoid teleoperation are useful to control robotic arms, such as:

- Cartesian Telemanipulation
- Joint Mapping Telemanipulation.

The distinction is based on the way the human movement is replicated by the robot. The Cartesian Teleoperation has the objective of controlling the End Effector position and orientation of the robot within the workspace. The Joint mapping focuses on imitating the Human upper limb pose by controlling the robot arm directly in the joint space. This technique creates a direct mapping between the human and the robot joints' positions. In this work were developed both the Cartesian and the joint mapping telemanipulation. These techniques were later used as a tool to perform teaching-by-demonstration, as there will be explained in Chapter 5.

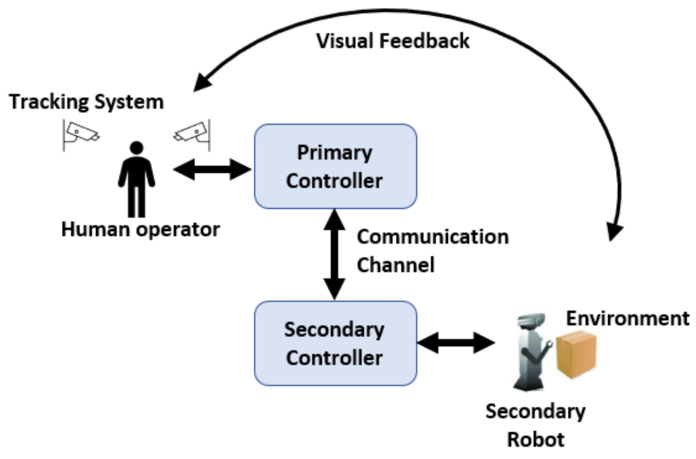


Figure 4.2: Teleoperation scheme used in this work. The human operator teleoperates the Robot and interacts with the environment using visual feedback.

1 Cartesian Teleoperation

During the cartesian teleoperation, the control was designed to create a mapping between the human reference position and the robot's end-effector. The control is depicted in figure 4.3. Starting from the two ROS TFs between the Base and the Wrist frame, it was performed the digital signal processing explained in Chapter 3, composed of a Moving Average filter followed by the normalization of the resulting quaternion.

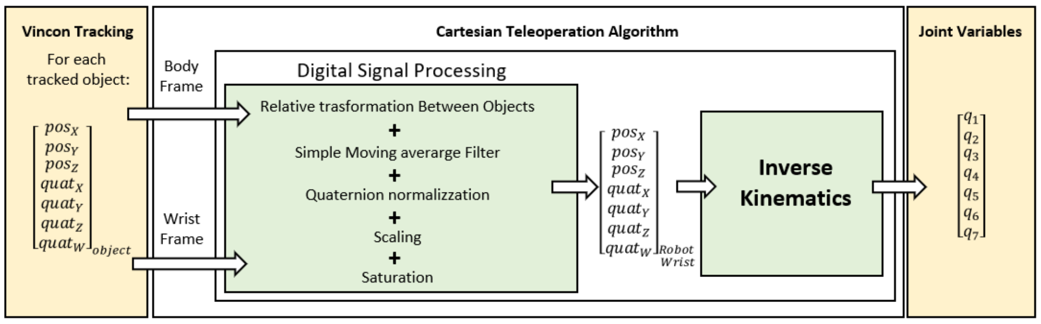

Vincon Tracking
For each tracked object:
$[pos_X, pos_Y, pos_Z, quat_X, quat_Y, quat_Z, quat_W]_{object}$
Cartesian Teleoperation Algorithm
Body Frame
Wrist Frame
Digital Signal Processing
Relative trasformation Between Objects
+
Simple Moving averarge Filter
+
Quaternion normalizzation
+
Scaling
+
Saturation
$[pos_X, pos_Y, pos_Z, quat_X, quat_Y, quat_Z, quat_W]_{Robot\ Wrist}$
Inverse Kinematics
Joint Variables
$[q_1, q_2, q_3, q_4, q_5, q_6, q_7]$

Figure 4.3: Cartesian teleoperation algorithm for each arm.

In such a way, for each time sample, the position and the orientation of the human wrist within the human workspace were obtained. To transform these TFs into a measurement valid for the robot, a scaling operation was executed as follows:

$$p_R = \frac{p_H \, R_{arm}}{H_{arm}}$$

where H_{arm} is the length of the human operator's arm, R_{arm} is the length of the robotic arm, p_H and p_R are respectively the cartesian position of the Human and the robot wrist, in terms of $[x, \ y, \ z]^T$ components. In addition, to avoid singularity configurations, a saturation in the Robot workspace was performed. Whenever the robot receives as an input a cartesian position outside its workspace, the input is saturated in X_{sat} Y_{sat} or Z_{sat} admissible values. Once the position and orientation of the end-effector are known, the *inverse kinematics problem* (IK) was solved, which consists on the determination of the robot joint position starting from a desired robot end-effector configuration. The solution to this problem is of fundamental importance in order to transform the motion specifications, assigned to the end-effector in the operational space, into the corresponding joint space motions allowing the execution of the desired motion. Since the considered Robot arm is a redundant manipulator, an infinite set of solutions of the IK exist. For this reason, every time step was performed the so-called *Inverse Differential Kinematics*, which is based on the Jacobian matrix **J**. Following the approach of [9], let us suppose a motion trajectory assigned to the end-effector in terms of v_e, and the initial conditions on position and orientation. The aim is to determine a feasible joint trajectory $(q(t), \ \dot{q}(t))$ that reproduces the given trajectory. the differential kinematics equation to consider can be formally written as:

$$v_e = J(q)\dot{q}$$

where $J(q)$ is the corresponding $(r \times n)$ Jacobian matrix.

In the case of $n = r$, the IK can be easily performed as follows:

$$\dot{q} = J(q)^{-1} v_e \tag{4.1}$$

However, whether the manipulator is redundant $(r < n)$, the Jacobian matrix is characterized by a higher number of columns than rows and infinite solutions exist. A viable solution method is to formulate the problem as a constrained linear optimization problem, that solves 4.1 and minimize the quadratic cost functional of joint velocities:

$$g(\dot{q}) = \frac{1}{2}\dot{q}^T\mathbf{W}\dot{q}$$

Where **W** is a suitable (n $\times$ n) symmetric positive definite weighting matrix. As in [9], solving via the Lagrange multipliers methods the optimal solution yield:

$$\dot{q} = W^{-1}J^T(JW^{-1}J^T)^{-1}v_e$$

If the initial joint configuration $q(0)$ is known, joint positions can be computed by integrating over time as follows:

$$q(t) = \int_0^t \dot{q}(\tau)d\tau + q(0)$$

Since the real-time teleoperation algorithm is performed in discrete time with a frequency $T = 50$ Hz, the integration must be done using numerical techniques. Considering the joint position at the previous time step as the initial condition $q(0)$, the joint angles are computed for each time samples using the Euler integration method. Knowing the joint positions $q(t_{k-1})$ and velocities $\dot{q}(t_{k-1})$ at the time sample $k-1$, it is possible to compute:

$$q(t_k) = q(t_{k-1}) + \dot{q}(t_{k-1})T \tag{4.2}$$

1.1 Cartesian Teleoperation Result

As illustrated in figure 4.4, the consequence of the cartesian teleoperation is a perfect robot achievement of the Cartesian position and orientation target inside the robot workspace, by simply tracking the human wrist position and orientation. The main advantage is the possibility of controlling directly the position and orientation of the end-effector of the robot, which may result to be convenient for the execution of some tasks such as the grasp of an object. Another important advantage is that, since we have the cartesian position, the recorded position can be used with different robotics platform with different kinematics. It will be necessary to update the inverse kinematics, but still we have the possibility to transfer the recorded trajetory to robotic platform. However, in case of an obstacle, the human operator does not have any type of control over the robot arm positioning within the workspace, particularly over the elbow position, making the task potentially unfeasible.

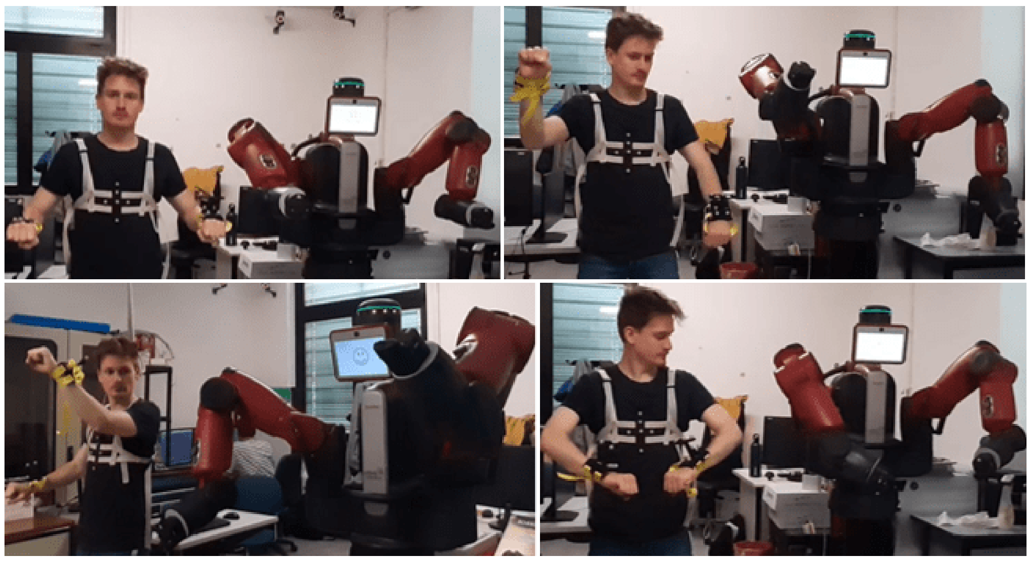


Figure 4.4: Dual-arm Cartesian teleoperation.

2 Direct Joint Mapping

A teleoperation alternative to the cartesian control is the so-called *Joint Mapping*, in which the robot arm is controlled to imitate the pose of the entire human upper limb. This teleoperation possibility is useful whether are present some obstacles in the environment. The human operator is then able to avoid any obstacle by simply using visual feedback (directly watching it or via webcam) resulting in a safer motion for the robot. The direct mapping of the robot's joint with the human body results also in a more 'natural' movement, which will be prone to have an higher human acceptance in collaborative environment. The mapping between the human body position and the robot position must be carefully studied. As presented in [8], to control the position of the human wrist, it may be only sufficient to control the first four angles of the upper limb kinematic chain (θ_1, θ_2, θ_3, θ_4,), which correspond to the shoulder angles and the elbow Flexione - Extension rotation.

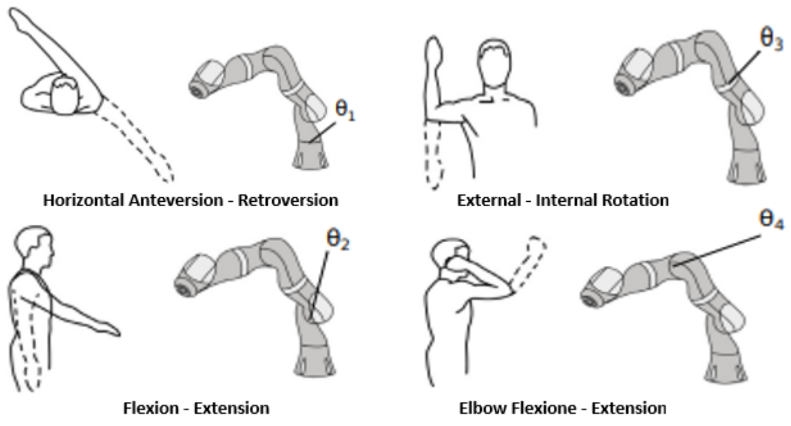

θ_1
Horizontal Anteversion - Retroversion
θ_3
External - Internal Rotation
θ_2
Flexion - Extension
θ_4
Elbow Flexione - Extension

Figure 4.5: Example of mapping between human joint into robotic arm joint. To control the position of the wrist, it is necessary to map only the first four joints of the kinematic chain, which are the shoulder joints and the elbow joint. Figure taken from [8]

In this thesis, the mapping was done by reading from the Vicon tracking system the respective Euler angle and then remapping the human angle interval into the robot angle interval. Using a saturation, it was also avoided any possible miscalculation or tracking misdetection. As will be explained in the next section, to control one more Degree of Freedom, it was mapped also the wrist Pronation-Supination rotation, corresponding to the joint θ_5.

The humna-robot joints mapped are

- Shoulder Joint: $\theta_1,\ \theta_2,\ \theta_3$;
- Elbow Joint θ_4: which control the position of elbowm usefull for imitiate the human limb pose;
- Wrist Joint: $\theta_5,\ \theta_6,\ \theta_7$.

In figure 4.6 is presented the general algorithm to perform the joint mapping teleoperation. Starting from the two ROS TFs between the Base and the Elbow frame, and between the Elbow and the Wrist frame, after the digital signal processing explained in Chapter 3, composed by the Simple Moving Average filter and the normalization of the quaternion, there were calculated the Euler angles for the joint mapping.

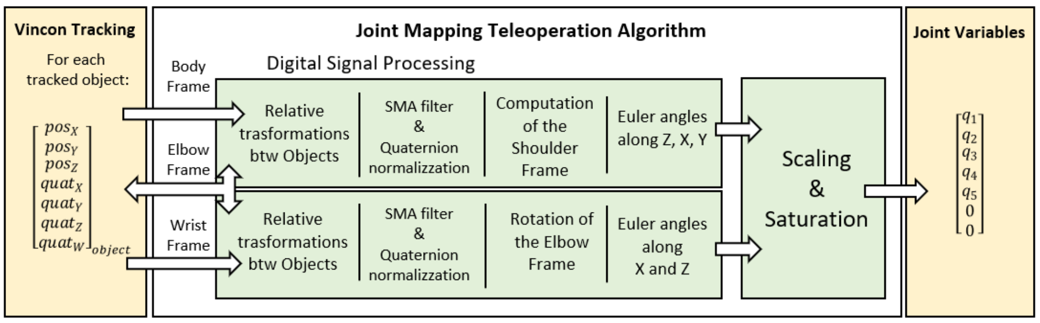

Vincon Tracking
For each tracked object:
$\begin{bmatrix} pos_X \\ pos_Y \\ pos_Z \\ quat_X \\ quat_Y \\ quat_Z \\ quat_W \end{bmatrix}_{object}$
Joint Mapping Teleoperation Algorithm
Digital Signal Processing
Body Frame
Elbow Frame
Wrist Frame
Relative trasformations btw Objects
SMA filter & Quaternion normalizzation
Computation of the Shoulder Frame
Euler angles along Z, X, Y
Relative trasformations btw Objects
SMA filter & Quaternion normalizzation
Rotation of the Elbow Frame
Euler angles along X and Z
Scaling & Saturation
Joint Variables
$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ 0 \\ 0 \end{bmatrix}$

Figure 4.6: Mapping between human joint into robotic arm joint. To control the position of the wrist, it is necessary to map only the first four joints of the kinematic chain, which are the shoulder joints and the elbow joint.

2.1 Shoulder Joint mapping

Let us call B, S and E the reference frames placed respectively on the thorax, considered ad base frame, on the shoulder and on the elbow. Let A_E^B, A_S^B and A_S^E be the Homogeneous transformation matrices respectively from frame B to frame E, frame B to frame S and frame E to frame S, following [9] notation.

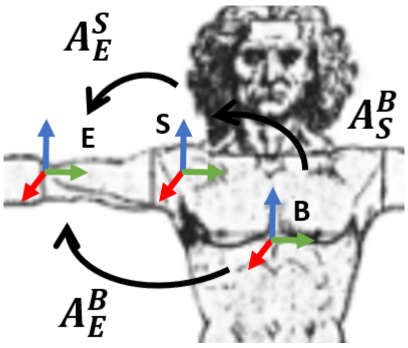


Figure 4.7: Trasformations among base frame, shoulder frame and elbow frame.

As is illustrated in Figure 4.7, the first tree angles of the kinematic chain, θ_1, θ_2 and θ_3 can be simply calculated using the Homogeneous transformation A_E^S as follows:

$$A_E^S = (A_S^B)^{-1} A_E^B$$

Where A_E^B is calculated for each time sample from the TF of the object Vicon "Elbow", as explained in the previous chapter. Moreover, A_S^B is a fixed translation of $p_S^B = [p_X,\ p_Y,\ p_Z]$, with the following form:

$$A_S^B = \begin{bmatrix} 1 & 0 & 0 & p_X \\ 0 & 1 & 0 & p_Y \\ 0 & 0 & 1 & p_Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The calculation of A_E^B from the TF read from Vicon is possible by transforming the TF list of 7 elements, pos_x, pos_y, pos_z, rot_x, rot_y, rot_z, rot_w, into a transformation matrix in the following form:

$$A_E^B = \begin{bmatrix} R_E^B & p_E^B \\ 0 & 1 \end{bmatrix}$$

where R_E^B is the rotation matrix and p_E^B is the translation vector from frame B to frame E. The first step in finding the rotation matrix is to transform the quaternion into the Euler angles, to point out the rotation along Z, X and then Y. Note that the order of the rotations is important because, in this applied case, the rotation along θ_2 depends on the rotation along θ_1, and consequently the one along θ_3 depends on θ_1 and θ_2. The resulting frame orientation is then obtained by the composition of rotations with respect to current frames, and it can be computed via postmultiplication of the matrices of elementary rotation [9] as follows:

$$R_E^B = R_z(\theta_1)R_x(\theta_2) * R_y(\theta_3) \tag{4.3}$$

The rotation matrix $R(\eta, \epsilon)$ corresponding to a given quaternion has the form:

$$\mathbf{R_E^B}(\eta,\epsilon) = \begin{bmatrix} 2(\eta^2+\epsilon_x^2)-1 & 2(\epsilon_x\epsilon_y - \eta\epsilon_z) & 2(\epsilon_x\epsilon_z+\eta\epsilon_y) \\ 2(\epsilon_x\epsilon_y+\eta\epsilon_z) & 2(\eta^2+\epsilon_y^2)-1 & 2(\epsilon_y\epsilon_z-\eta\epsilon_x) \\ 2(\epsilon_x\epsilon_z-\eta\epsilon_y) & 2(\epsilon_y\epsilon_z+\eta\epsilon_x) & 2(\eta^2+\epsilon_z^2)-1 \end{bmatrix} \tag{4.4}$$

Then, it is possible to extract the Euler angles in the following way. Referring with C_θ and S_θ respectively the cos_θ and the sin_θ, as in [9], according to equation 4.3, the resulting rotation matrix is the following:

$$\mathbf{R_E^B}(\eta,\epsilon) = \begin{bmatrix} C_{\theta1}C_{\theta3} - S_{\theta1}S_{\theta2}S_{\theta3} & -S_{\theta1}C_{\theta2} & C_{\theta1}S_{\theta3}+S_{\theta1}S_{\theta2}C_{\theta3} \\ S_{\theta1}C_{\theta3}+C_{\theta1}S_{\theta2}S_{\theta3} & C_{\theta1}C_{\theta2} & S_{\theta1}S_{\theta3}-C_{\theta1}S_{\theta2}C_{\theta3} \\ -C_{\theta2}S_{\theta3} & S_{\theta2} & C_{\theta2}C_{\theta3} \end{bmatrix}$$

Renaming $R_E^B(\eta,\epsilon)$ as:

$$\mathbf{R_E^B}(\eta,\epsilon) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{4.5}$$

The first three angles of the kinematic chain θ_1, θ : and θ_3 are computed as:

$$\begin{aligned} \theta_1 &= Atan2\left(-r_{12}, r_{22}\right) \\ \theta_2 &= Atan2\left(r_{32}, \sqrt{r_{12}^2+r_{22}^2}\right) \\ \theta_3 &= Atan2\left(-r_{31}, r_{33}\right) \end{aligned} \tag{4.6}$$

2.2 Elbow And Wrist Joint Mapping

As in the case of the Shoulder joint mapping, calling E and W the reference frames placed respectively on the elbow and the wrist, the Homogeneous transformation matrix between them is called A_W^E, and it is known any time samples from the Vicon TF.

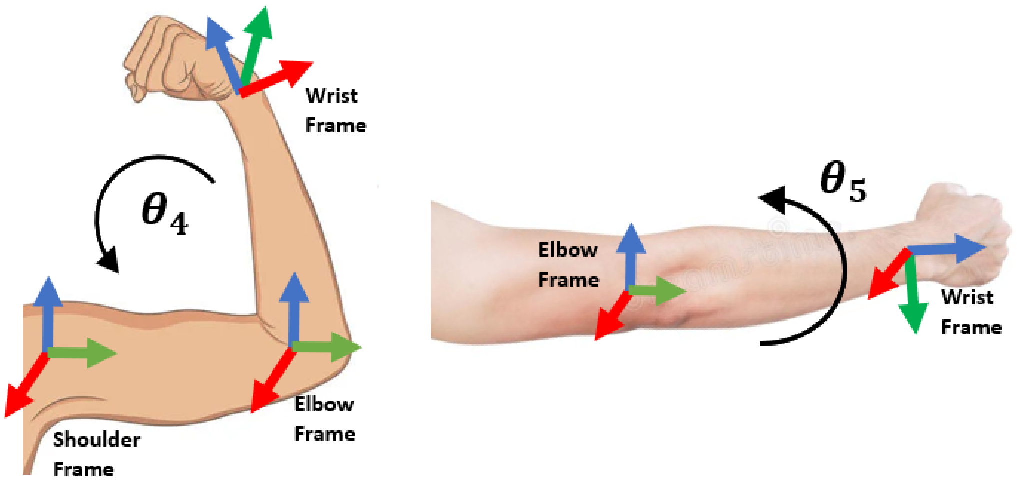


Figure 4.8: θ_4 Elbow Flexione - Extension (Left) and θ_5 Wrist Pronation - Supination rotation (Right).

After the initial calibration explained in the previous chapter, is it possible to find $A_{E,\,MISS}^E$, which represents the possible initial misalignment between the frames E and W due to human error in fixing the Vicon objects. As a consequence, another reference frame was added, called E_{MISS}.

Therefore, the angles θ_4 and θ_5, respectively Elbow Flexione-Extension and Wrist Pronation-Supination, are extracted from the Homogeneous matrix $A_W^{E,MISS}$, computed as follows:

$$A_W^{E,MISS} = (A_{E,\,MISS}^E)^{-1} A_W^E$$

To isolate the angles, the procedure is exactly the same as for the shoulder angles in the previous subsection. Then, from $A_W^{E,MISS}$ it were computed the Euler Angles XZY. All the explained calculations were done for both arms using the *Scipy Spatial Transform* Python library, considering intrinsic rotations [12].

2.3 Direct Joint Mapping Result

As illustrated in figure 4.9, the consequence of the Joint mapping teleoperation is a perfect robot imitation of the human upper limb pose. The main advantage is the possibility of controlling directly the position of the elbow of the robotic arm. Then, in case of an obstacle, the human operator can avoid it easily by using visual feedback, without worrying about the kinematics of the robot.

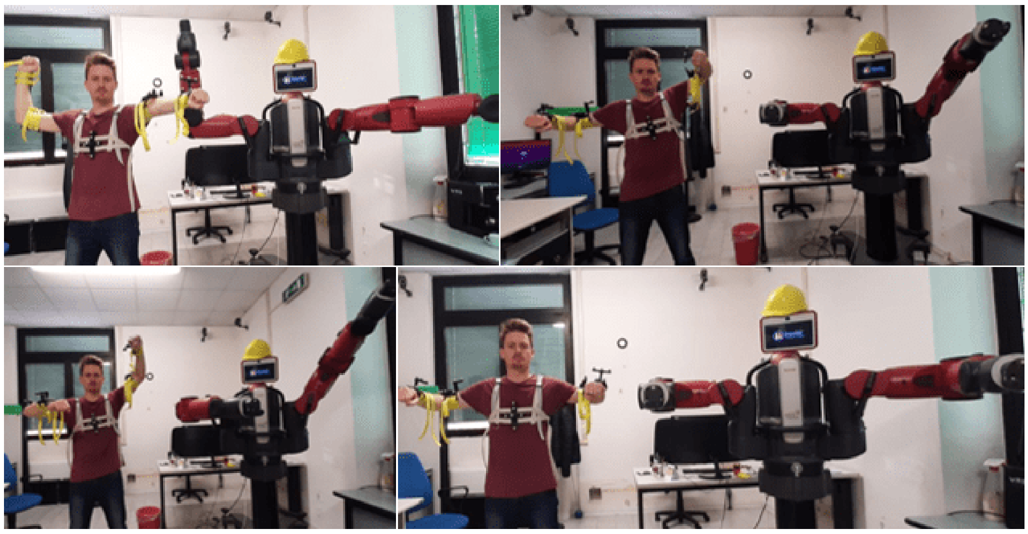


Figure 4.9: Dual arm Joint Mapping teleoperation.

2.4 Control Of θ_6: Wrist Flexion - Extension Rotation

As explained in the last section, the joint mapping was performed controlling only the first five angles of the kinematic chain, namely the shoulder and the elbow angles,, in order to control the wrist position and wrist Pronation-Supination rotation. To make the teleoperation more task suitable, the sixth joint, called $\theta_6 = W_1$, see Figure 2.5, was also controlled based on the direction of the instantaneous end effector cartesian velocity along the x and z directions. Depending on the performed task, the joint was controlled to be coherent with the velocity direction or to be inopposition, by simply using a plus or a minus sign in equation 4.7.

Defining $p_{x,\,t}$ and $p_{z,\,t}$ the end effector position at time t, it is possible to calculate $v_{x,\,t}$ and $v_{z,\,t}$ the end effector velocity at time t as follow:

$$v_{x,\,t} = \frac{p_{x,\,t} - p_{x,\,t-1}}{\Delta T} \quad and \quad v_{z,\,t} = \frac{p_{z,\,t} - p_{z,\,t-1}}{\Delta T}$$

where ΔT is the time interval between the two samples, which is calculated considering the frequency at which the robot is controlled, so is equal to 1/50 HZ. Starting with an initial condition of $\theta_{6,t=0} = 0$, the joint angle θ_6 is computed as follow:

$$\theta_{6,\,t} = \theta_{6,\,t-1} \pm K_f \; arctan\left(\frac{v_{z,\,t-1}}{|v_{x,\,t-1}|}\right) \tag{4.7}$$

where K_f is a factor that scales the importance of the velocity and produces a smoother movement. The velocity vector is calculated considering only the Z and X components as follows:

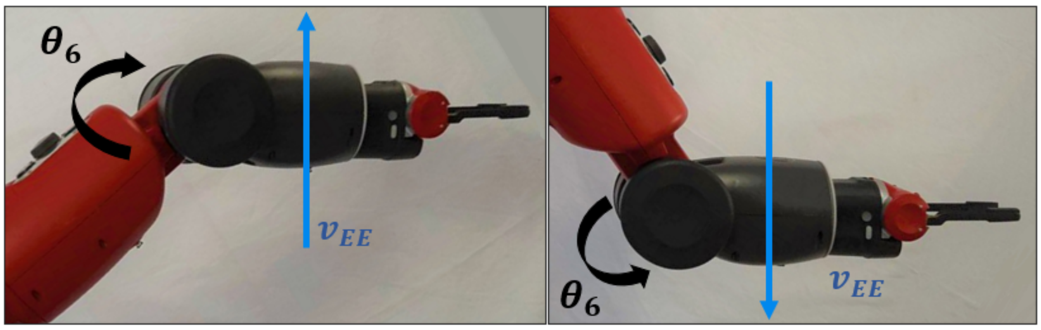
$$v_{xz}(t) = arctan\left(\frac{v_{z,\,t-1}}{|v_{x,\,t-1}|}\right)$$

Being coherent with the velocity can be useful to avoid some obstacle within the robot workspace.

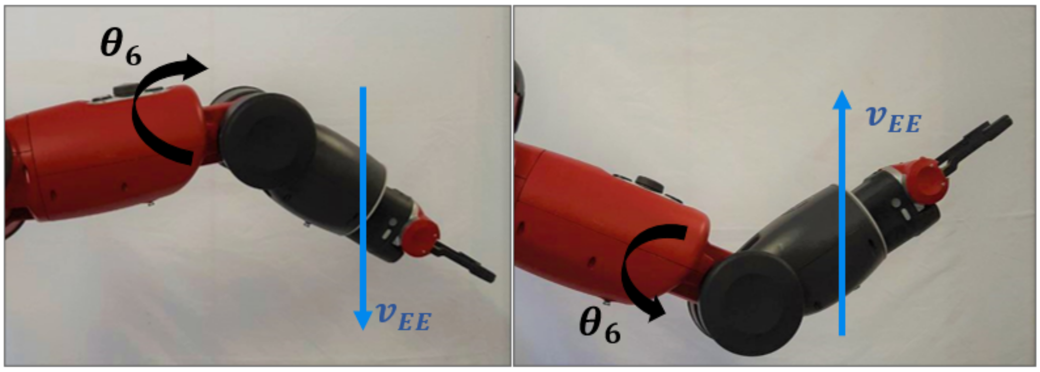
$$\theta_{6,\,t} = \theta_{6,\,t-1} + K_f \; v_{xz}(t)$$

As illustrated in Figure 4.10b, whenever the velocity vector has a negative sign and the end effector is going down, θ_6 is controlled simulating a wrist movement downwards, and vice versa if the velocity vector has a positive sign.

Being not coherent with the velocity keeps the gripper always parallel to the ground. This can be useful in grasping tasks in which the objects are



(a)



(b)

Figure 4.10: In 4.10a Sixth joint control for each arm based on the end-effector velocity. Using this control, the human operator is assisted in obstacle avoidance . In 4.10b Sixth joint control for each arm based on the end-effector velocity. Using this control, the human operator is assisted in obstacle avoidance

placed on a low table, as performed in this thesis in the next chapters. The computation is the following:

$$\theta_{6,\,t} = \theta_{6,\,t-1} - K_f\; v_{xz}(t)$$

Chapter 5

Humanoid Robots Imitation Learning

The term *Imitation Learning* refers to the process of acquiring a control strategy to achieve a task using a learning algorithm starting from a human demonstration. Calinon et al. in [2] propose an overview of learning approaches for the acquisition of control strategies to achieve a task based on the concept of motion primitive. Based on this framework, the teaching of the desired trajectory and the following execution of the task can be obtained from different modalities and learning strategies, such as:

- **Human imitation via visual observation**: a teaching method that allows the human operator to communicate new behaviours to the robot by 'showing' instead of 'telling', usually performed utilizing a tracking system placed on the human operator limbs composed by cameras or sensors;
- **Programming by demonstration (PbD)** concentrates on the extraction of the control law from the demonstration of the task performed by a human, for example via kinesthetic teaching, which means that the operator moves the robot along the trajectory, or teleoperation, via intuitive telemanipulation or using some haptic instruments;
- **Reinforcement Learning (RL)** can be used to either learn a skill from a reward signal or like in this scenarion, is used to adapt and improve the skill from an already existing demonstration. Through different repetition of trials and errors, the robot is capable to improve his control law.

Figure 5.1 shows some examples of learning strategies used on humanoids or robotics arms. The example at the top left is taken from [13] in which the humanoid was taught to reach a chess piece and grasp it. In the example at the top left, from [14], it was taught the humanoid ASIMO the bimanual pouring of a liquid in a glass. The example at the bottom left is taken from [15] in which the robotic arm was taught to flip a pancake using an RL approach. Lastly, in the bottom left, from [16], the humanoid Fujitsu HOAP-3 learned a feeding task.

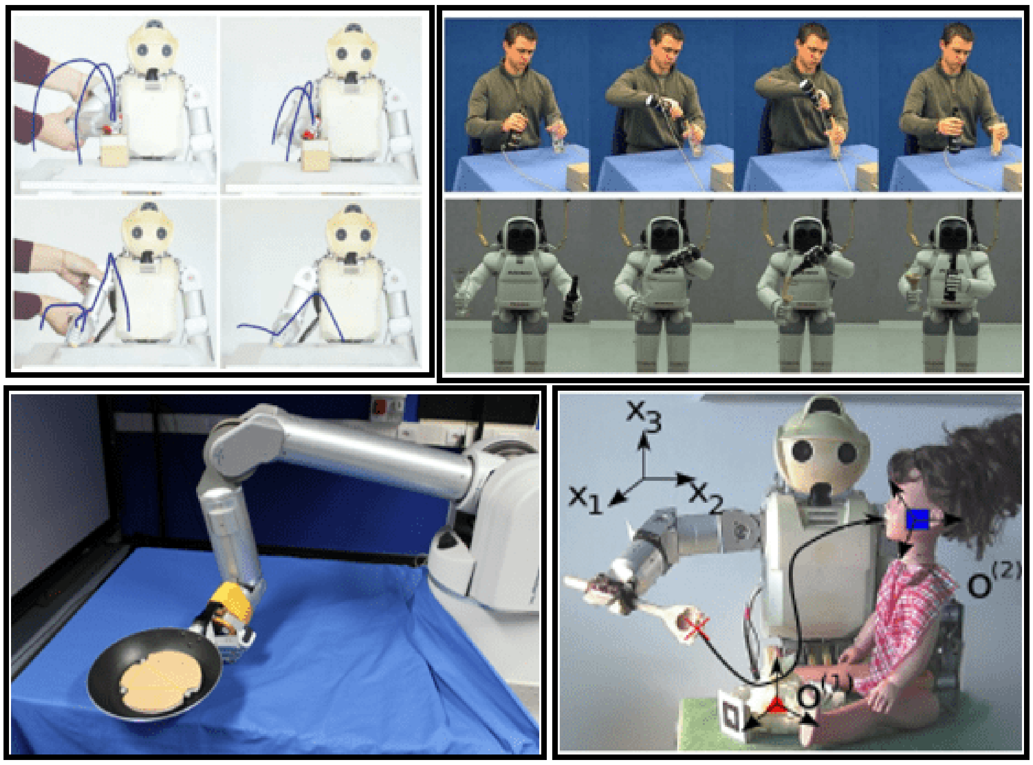


Figure 5.1: Examples of Robot Learning via Kinesthetic Demonstrations or human imitation. Images taken from [13] [14], [15], [16].

This chapter will present an overview of a learning algorithm via Programming-by-Demonstrations based on Gaussian Mixture Regression, and a first initial study case based on an existing dataset.

1 Dynamical System Modulation Via Gaussian Mixture Regression

This section will present the algorithm used to extract from several demonstrations a Dynamical Movement Primitive (DMP) that is modulated with a converging control law. This will allow a robot to learn a simple goal-directed motion and accurately replicate it, despite dissimilarities in the initial conditions or environmental perturbations. The starting point of the presented method is PbD. It is a method that tries to extract a control law from task demonstrations performed by a human. It is important to notice that while in [13] PbD was performed via kinaesthetic demonstrations, in this thesis it was achieved via Teleoperation. As illustrated in figure 5.2, the algorithm is based on the acquisition of a *Training Set* of N demonstration $(\zeta^i,\ \xi^i)_{i=1}^N$, from which a function $\mathcal{F}_\xi(\zeta)$ is estimated, where ξ^i is a noisy measurement of $\mathcal{F}_\xi(\zeta^i)$:

$$\xi^i = \mathcal{F}_\xi(\zeta^i) + \epsilon^i \tag{5.1}$$

where ϵ^i is the Gaussian noise. The statement is to model the joint distribution of the *input variable* ζ and the *output variable* ξ as a Gaussian Mixture Model. By merging these variables in a vector $v = [\zeta^T\ \xi^T]^T$, it is possible to model its probability density function as a mixture of K Gaussian functions.

$$P(v) = \sum_{k=1}^{K} \pi_K \mathcal{N}(v;\ \mu_K,\ \Sigma_K), \quad \textit{such that} \quad \sum_{k=1}^{K} \pi_K = 1.$$

Where $\pi_k\ \epsilon\ [0,\ 1]$ are the priors and $\mathcal{N}(v;\ \mu_K,\ \Sigma_K)$ are Gaussian functions with mean value μ_k and covariance matrix Σ_k, defined as follow:

$$\mathcal{N}(v;\ \mu_K,\ \Sigma_K) = ((2\pi)^d\ |\Sigma_K|)^{-\frac{1}{2}}\ exp\ (-\frac{1}{2}(v-\mu_k)^T \Sigma_k^{-1}(v-\mu_k)), \tag{5.2}$$

where d is the dimensionality of the vector v.
The mean vector μ_k and the covariance matrices Σ_k can be separated into their respective input and output components:

$$\mu_k = [\mu_{k,\,\zeta}^T\ \mu_{k,\,\xi}^T]^T \tag{5.3}$$

$$\Sigma_k = \begin{bmatrix} \Sigma_{k,\,\zeta} & \Sigma_{k,\,\zeta\xi} \\ \Sigma_{k,\,\xi\zeta} & \Sigma_{k,\,\xi} \end{bmatrix} \tag{5.4}$$

The K gaussian functions, see Equation 5.2 characterized by an own mean vector and covariance matrices are computed through the Expectation Maximimization (EM) algorithm, expalined in the next subsection. The GMM computes a joint probability density function for the input and the output so that the probability of the output conditioned on the input is a GMM. Hence, it is possible, after the training process, to recover the expected output variable $\tilde{\xi}$, given the observed input variable ζ.

$$\tilde{\xi} = \mathcal{F}_{\xi}(\zeta) = \sum_{k=1}^{K} h_K(\zeta)(\mu_{k,\,\xi} + \Sigma_{k,\,\xi\zeta}\, \Sigma_{k,\,\zeta}^{-1}(\zeta - \mu_{k,\,\zeta})), \tag{5.5}$$

where $h_K(\zeta)$ are the weights given by:

$$h_K(\zeta) = \frac{\mu_k\, \mathcal{N}(\zeta;\ \mu_{k,\,\zeta},\ \Sigma_{k,\,\zeta})}{\sum_{k=1}^{K} \mu_k\, \mathcal{N}(\zeta;\ \mu_{k,\,\zeta},\ \Sigma_{k,\,\zeta})} \tag{5.6}$$

The Tilde sign indicates that we are dealing with expectation values.

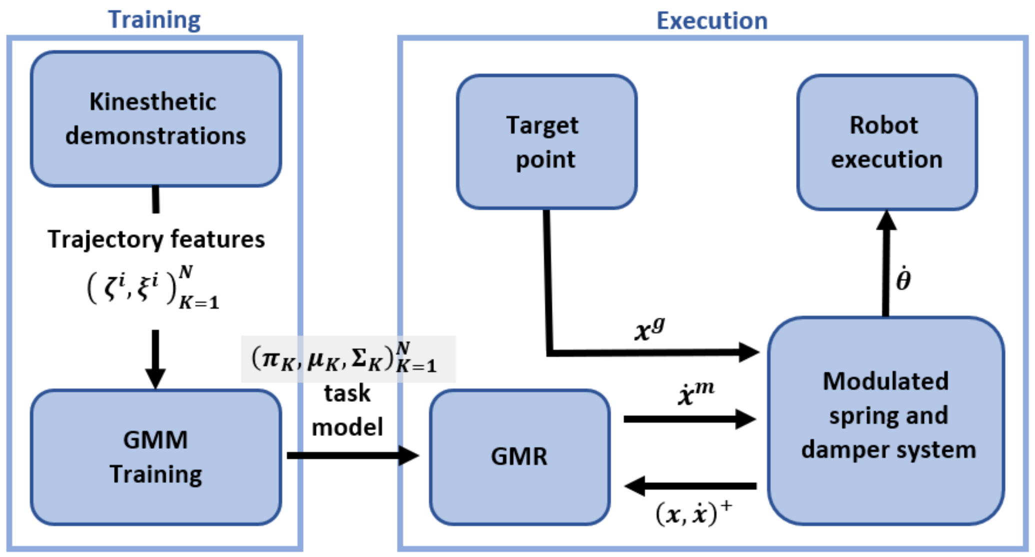


Figure 5.2: Architecture of the Learning algorithm.

Lastly, the GMM explained above was used to train a *Velocity Model*, by taking as input the time and as output the velocity of the end-effector $\dot{x}$. In such a way the movement is modelled as a velocity profile in the following form:

$$\dot{x}^m = \tilde{\mathcal{F}}_{\dot{x}}(t) \tag{5.7}$$

1.1 GMM Training Via Standard E-M Algorithm

The Expectation-Maximimization (EM) algorithm is a general method of finding the Maximum Likelihood Estimation (MLE) from incomplete data via an improved descent algorithm. [17]. Assuming that the data $\mathcal{X} = \{x_1, ..., x_N\}$ are generated in an independent way from a mixture density as

$$P(x_i) = \sum_{j=1}^{M} P(x_i|\omega_j;\ \theta_j)P(\omega_j)$$

where each component of the mixture is denoted ω_j and parametrized by θ_j. Given the independence assumption, the log-likelihood of the parameters given the dataset is the following:

$$l(\theta|\mathcal{X}) = \sum_{i=1}^{N} log \sum_{j=1}^{M} P(x_i|\omega_j;\ \theta_j)P(\omega_j)$$

According to [18], the EM algorithm for mixture models is an iterative method for solving this credit-assignment problem by finding the best model of the data that maximize the likelihood $l(\theta|\mathcal{X})$. The intuition is to look for a "hidden" random variable z indicating which data point was generated by which component, and then the maximization problem would decouple into a set of simple maximizations. By using z as the indicator unknown variable, a "complete-data" log-likelihood function can be written in the following form:

$$l_c(\theta|\mathcal{X}, \mathcal{Z}) = \sum_{i=1}^{N}\sum_{j=1}^{M} z_{ij} log P(x_i|\omega_j;\ \theta_j)P(\omega_j)$$

Therefore, the EM is an iterative algorithm divided into two steps. The *Expectation* step computes the expected complete data log-likelihood, indicated by $\mathcal{Q}(\theta|\theta_{||})$ while the *Maximization* step finds the parameters that maximize the likelihood. The two steps can be described as follows:

$$E\ step:\ \mathcal{Q}(\theta|\theta_k) = E[l_c(\theta|\mathcal{X}, \mathcal{Z})|\mathcal{X}, \theta_k] \tag{5.8}$$

$$M\ step:\ \theta_{k+1} = argmax_\theta \mathcal{Q}(\theta|\theta_k) \tag{5.9}$$

In real-valued data 5.8 and 5.9 can be used to model a mixture of Gaussian. The **E** step is used to compute the probability that gaussian **j** generates data point **i** as:

$$h_{i,j} = \frac{|\hat{\Sigma}_j^k|^{-1/2}\mathcal{N}(x_i;\ \hat{\mu}_j^k,\ \hat{\Sigma}_j^k)}{\sum_{l=1}^{M}|\hat{\Sigma}_j^k|^{-1/2}\mathcal{N}(x_i;\ \hat{\mu}_j^k,\ \hat{\Sigma}_j^k)} \tag{5.10}$$

According to the notation of the normal distribution in 5.2. Afterwards, the **M** step re-estimates the means and covariances of the gaussian using the weights obtained in 5.10 as:

$$\hat{\mu}_j^{k+1} = \frac{\sum_{i=1}^{N} h_{i,j} x_i}{\sum_{i=1}^{N} h_{i,j}} \tag{5.11}$$

$$\hat{\Sigma}_j^{k+1} = \frac{\sum_{i=1}^{N} h_{i,j}(x_i - \hat{\mu}_j^{k+1})(x_i - \hat{\mu}_j^{k+1})^T}{\sum_{i=1}^{N} h_{i,j}} \tag{5.12}$$

To perform the EM algorithm and compute the Gaussian Mixture Model of the dataset, the Python library *Scikit-learn* was used [19].

1.2 Modulated Spring And Damper System

The task model described before is used in addition to a modulated spring-and-damper dynamical system in order to enable a robotic arm with n joints to reproduce the task with sufficient flexibility. Calinon et. al, in [16] presented an algorithm directly in joint angle variables, even if the modulation $\dot{x}^m$ is in end-effector space. This can be advantageous for avoiding singularity problems related to inverse kinematics. The spring damper system will be:

$$\ddot{\theta}^s = \alpha(-\dot{\theta} + \beta(\theta^g - \theta))$$

where $\theta \in \mathbb{R}^n$ is the vector of the joint angles. This dynamical system produces straight paths (in joint space) to the target θ^g, which acts as an attractor of the system. This ensures that the robot reaches the target smoothly, despite any possible perturbations. The dynamical system is modulated by the variable $\dot{x}^m$ given by the task model of equation 5.7. To weigh the modulation is introduced the modulation factor $\gamma \in \mathbb{R}_{[0,1]}$, which weighs the importance of the task model with respect to the spring-and-damper system.

- If $\gamma = 0$ only the Spring Damper system is considered.
- If $\gamma = 1$ only the task model is considered.

To guarantee the convergence of the system to the target θ^g, γ has to tend to zero at the end of the movement, using the following:

$$\ddot{\gamma} = \alpha_\gamma(-\dot{\gamma} - \frac{1}{4}\alpha_\gamma \gamma) \quad with \quad \gamma_0 = 1$$

where γ_0 is the initial value and $\alpha_\gamma \in [\mathbb{0}, \mathbb{1}]$ is a scalar.

Since $\dot{x}^m$ lies in the end-effector space, the modulation is performed by solving the following constrained optimization problem:

$$\begin{aligned}
&\dot{\theta} = \underset{\dot{\theta}}{\operatorname{argmin}}(1-\gamma)(\dot{\theta}-\dot{\theta}^s)^T \bar{W}_\theta(\dot{\theta}-\dot{\theta}^s) + \gamma(\dot{x}-\dot{x}^m)^T \bar{W}_x(\dot{x}-\dot{x}^m) \\
&Subject\ to: \quad \dot{x} = J\dot{\theta}
\end{aligned} \tag{5.13}$$

Where **J** is the Jacobian of the robot arm m kinematic function **K**. $\bar{W}_\theta \in \mathbb{R}^{nxn}$ and $\bar{W}_x \in \mathbb{R}^{mxm}$ are diagonal matrices necessary to compensate for the different scales of the x and θ variables. The solution to the minimization problem (5.13) is given by [20]

The solution of the optimization problem according to [16] is the following:

$$\dot{\theta} = (W_\theta + J^T W_x J)^{-1}(W_\theta \dot{\theta}^s + J^T W_x \dot{x}^m)$$

Where $W_\theta = (1-\gamma)\bar{W}_\theta$ and $W_x = \gamma \bar{W}_x$

To summarize, the task is performed by integrating the following dynamical system:

$$\begin{aligned}
&\ddot{\theta}^s = \alpha(-\dot{\theta} + \beta(\theta^g - \theta)) \\
&\dot{\theta} = (W_\theta + J^T W_x J)^{-1}(W_\theta \dot{\theta}^s + J^T W_x \dot{x}^m)
\end{aligned}$$

To obtain a straightforward algorithm, in this thesis the modulated spring damper system was computed directly in cartesian space, to keep direct control of the position and orientation of the end-effector in case of some obstacle within the workspace. All the performed calculations are explained in the next section.

2 First Test Using An Existing Trajectories Dataset

As a first study step, the algorithm was tested using an existing dataset of trajectories performed by a robot to draw an alphabetic letter. The trajectory was performed twelve times with a different initial point. The result, illustrated in figure 5.3, are twelve similar trajectories with the same shape.

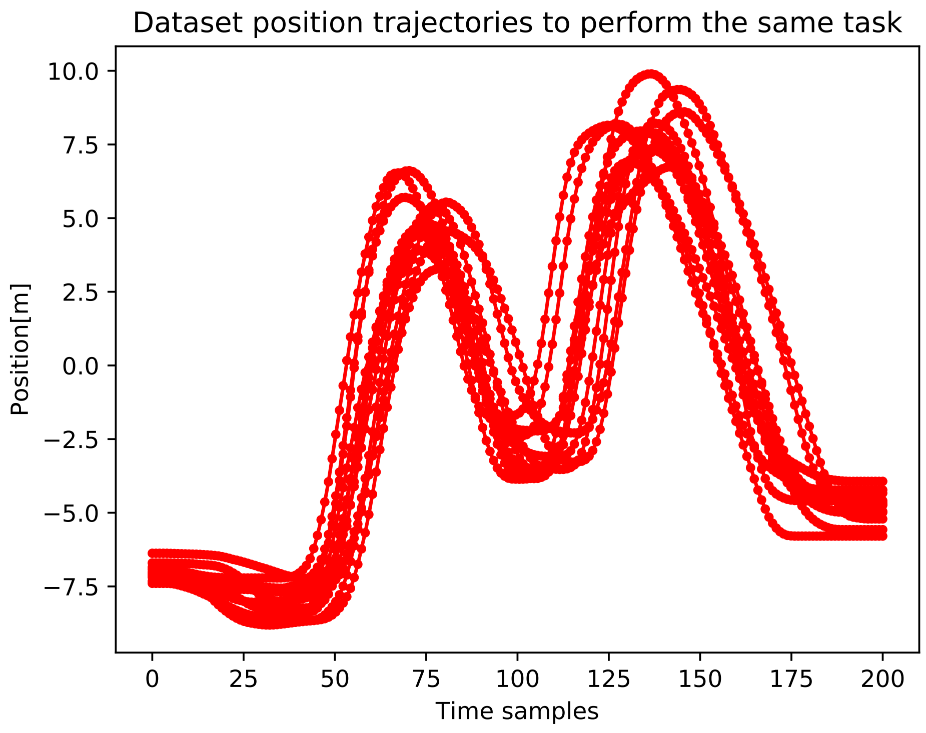


Figure 5.3: Plot of N trajecories to perform the same task

To perform the GMR computation explained in the previous section, from each position trajectory was computed the velocity trajectory, starting with initial velocity equal to zero. The computation was performed as follow:

$$v_k = \frac{p_k - p_{k-1}}{\Delta T} \tag{5.14}$$

where p_k and p_{k-1} are the position respectively at instant time k and k-1. ΔT is the constant time interval set to $1/50Hz$, that is 0.02 seconds.
After the computation of twelve velocity trajectories, the Gaussian mixture

regression was performed, obtaining a Motion Primitive as a velocity trajectory. In Figure 5.4 are illustrated all the components of the GMR algorithm.

The dots scattered in the plot represent the training data. The ellipses are the Gaussian components described in Equation 5.2, each one characterized by a mean vector, Equation 5.3 and covariance matrix, Equation 5.4. Finally, the red thick line defines the DMP as a velocity trajectory obtained by GMR, described in equation 5.5.

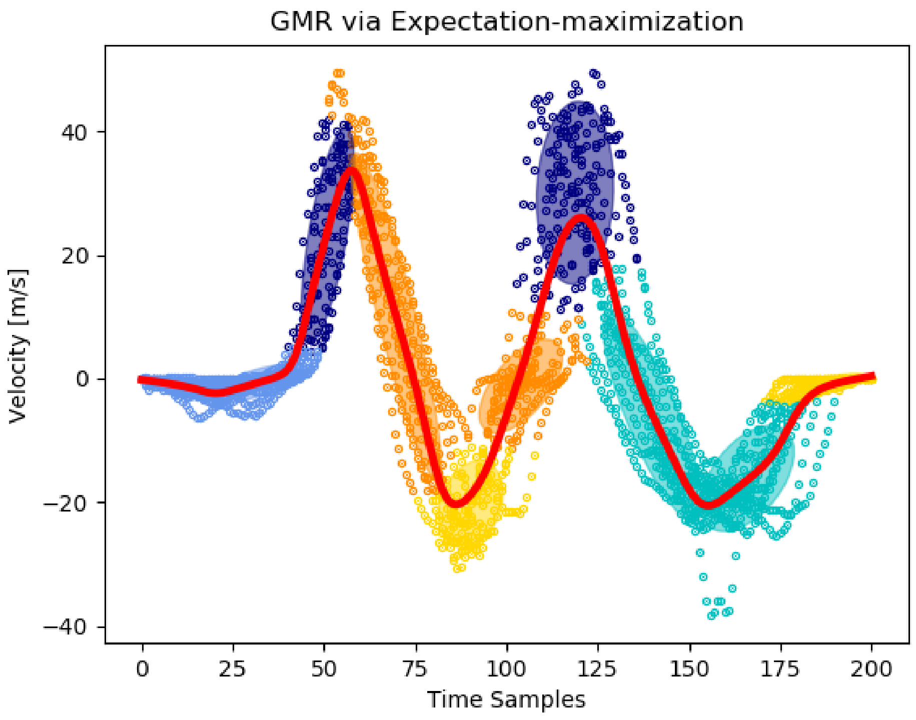


Figure 5.4: GMR of the velocity dataset computed via equation 5.14 form the position trajectories in Figure 5.3. The dots symbolise the training data, the ellipses are the Gaussian components, and the red thick line defines the trajectory obtained by GMR.

Considering a point X^g as the target the spring damper system can be written as follow:

$$\ddot{x}^S = \alpha\,(-\dot{x} + \beta(x^g - x))$$

Since the algorithm was performed in discrete time, for an easier understanding, the control law can be written as a PD controller in the following way:

$$a(t)^S = -K_d v(t) + K_p(x^g - x(t)) \tag{5.15}$$

where α and β are gains of the control law. Therefore, following the approach explained in the previous section, a discrete-time control law was computed as follows:

$$v(t+1) = \gamma\tilde{\mathcal{F}}_v(t+1) + (1-\gamma)v^S(t) \tag{5.16}$$

where γ is the modulation factor.
The velocity of the spring-damper system $v^S(t)$ was calculated as a first-order newton approximation from the acceleration in 5.15, as follows:

$$v^s(t) = v(t) + \tau a^S(t) \tag{5.17}$$

where $v(t)$ is the real velocity of the system at time t. Then, by replacing 5.15 in 5.17, it is possible to obtain:

$$v^s(t) = v(t) + \tau(-K_d\dot{x} + K_p(x^g - x))$$

The calculations were performed as finite-difference equation, considerng $x(0) = X_0$ and $V(0) = 0$.

In Figure 5.5 is represented the algorithm block diagram. $R(t)$ is the control regulator, which coincides with equation 5.16.

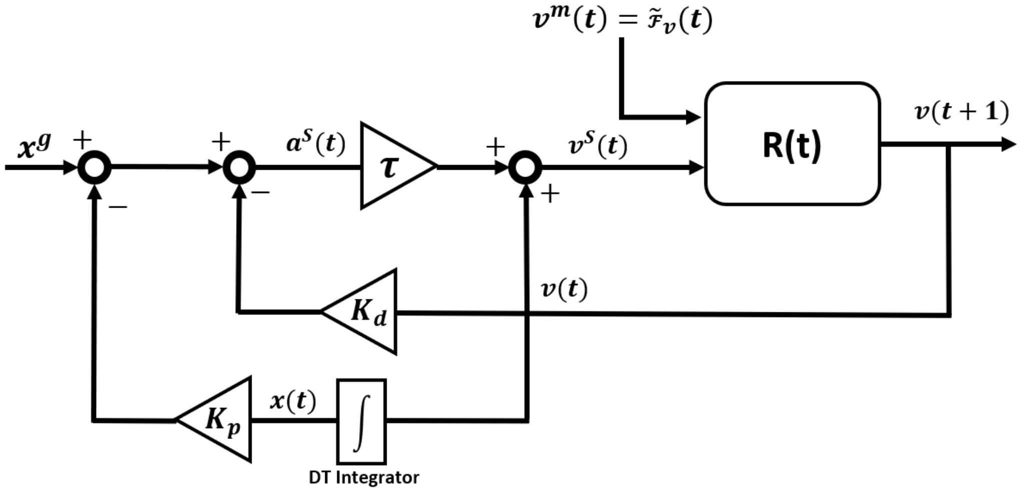


Figure 5.5: Control Law block diagram.

The convergence to the target depends on the modulation factor γ. Figure 5.6 represents different types of modulation factors used in this thesis, with are linear, parabolic and exponential behaviour, representing a different degree of convergency.

$$
\begin{aligned}
&\textbf{Linear} : \gamma = \frac{-t}{Time\ Samples} + 1 \\
&\textbf{Exponential} : \gamma = \exp\left(-5.5 \cdot \left(\frac{t}{Time\ Samples}\right)\right) \\
&\textbf{Parabolic} : \gamma = \left(\frac{-t}{Time\ Samples} + 1\right)^2
\end{aligned}
\tag{5.18}
$$

In this applied case: $Time\ Samples = 200$ By keeping the modulation factor constantly to zero or to one, the robot will follow respectively the PD controller trajectory or the GMR position trajectory.

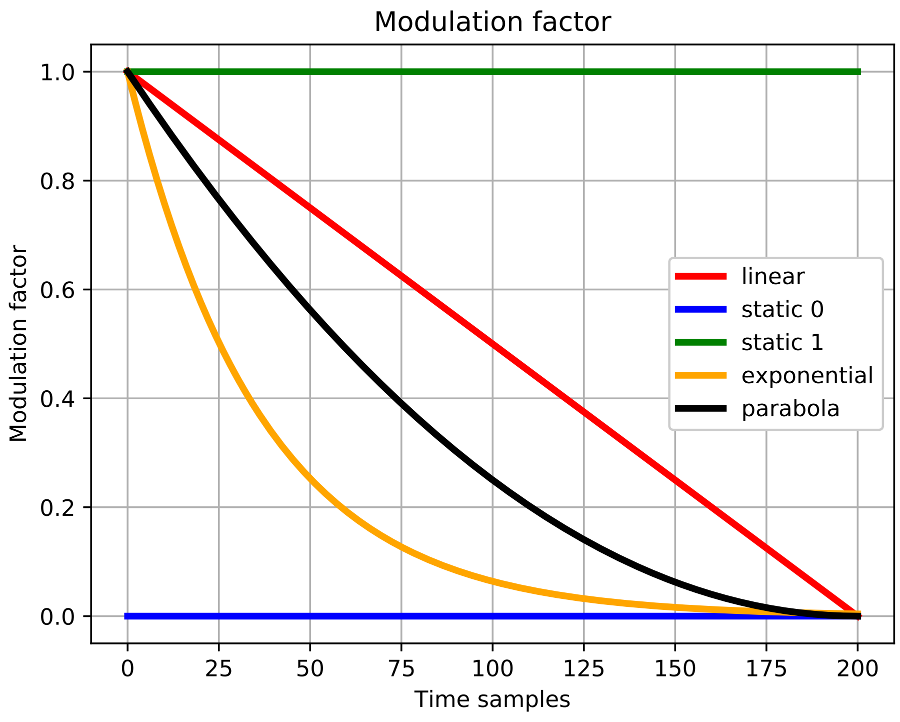


Figure 5.6: Different types of modulation factor γ

3 Resutling Modulated Trajectory

This section will present the resulting trajectory varying several parameters in the control law. In any case, the resulting trajectory, Figure 5.7, will be a modulation between a position trajectory obtained from the GMR Dynamical Movement Primitive velocity trajectory and a position trajectory obtained from the PD controller, to ensure the fast convergence to the target point x^g.

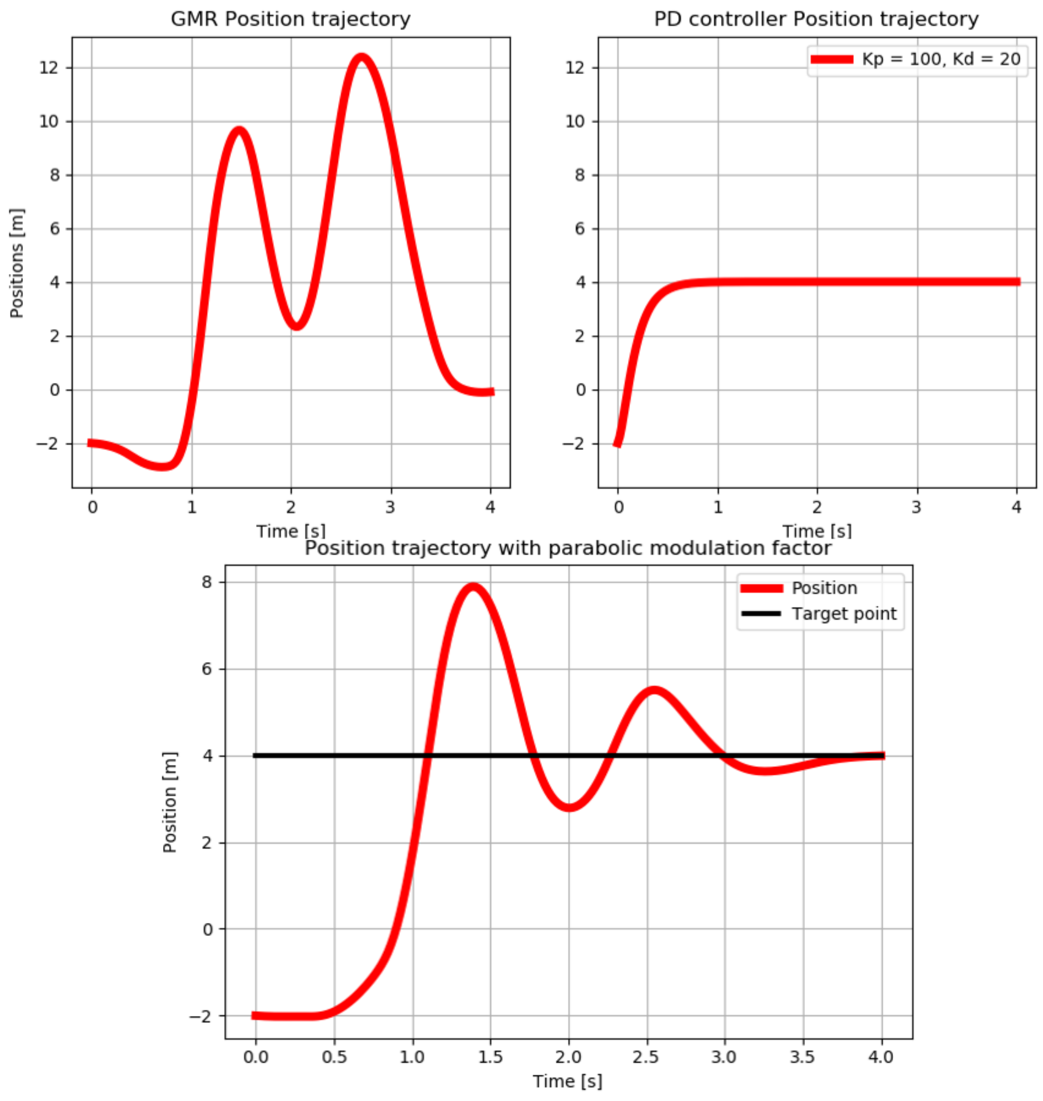


Figure 5.7: Position trajectory obtained from the GMR Dynamical Movement Primitive velocity trajectory (UP left) and position trajectory obtained from the PD controller (Up Right). The output is a modulation between these two trajectories (Down).

3.1 Algorithm Adaptation

One of the essential characteristics of robot Programming by human demonstration is the flexibility to variations of initial or final conditions in order to teach the robot a feasible trajectory for any similar tasks. Figures 5.8 and 5.9 (Right) show two different trajectories with the modulated control law respectively starting from two different initial points and with two different target points. In both cases, the target was reached in a robust way.

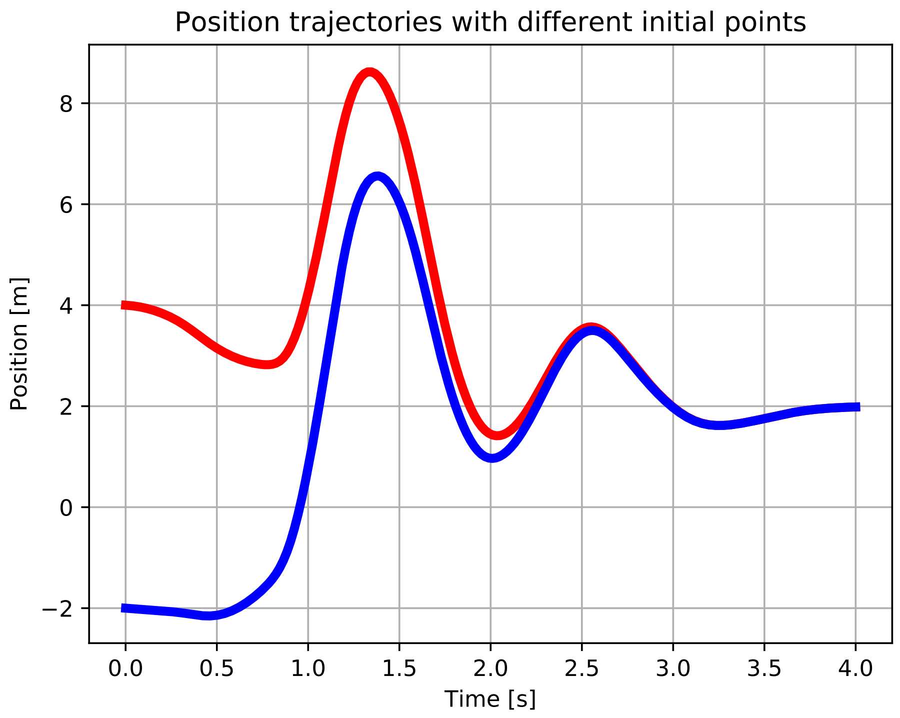


Figure 5.8: Trajectories with two different initial points, respectively 4.0 (Red) and −2.0 (Blue).

Figure 5.9 (Left) shows also the robustness of the algorithm in the case the target point is in the opposite direction with respect to the DMP direction of the motion.

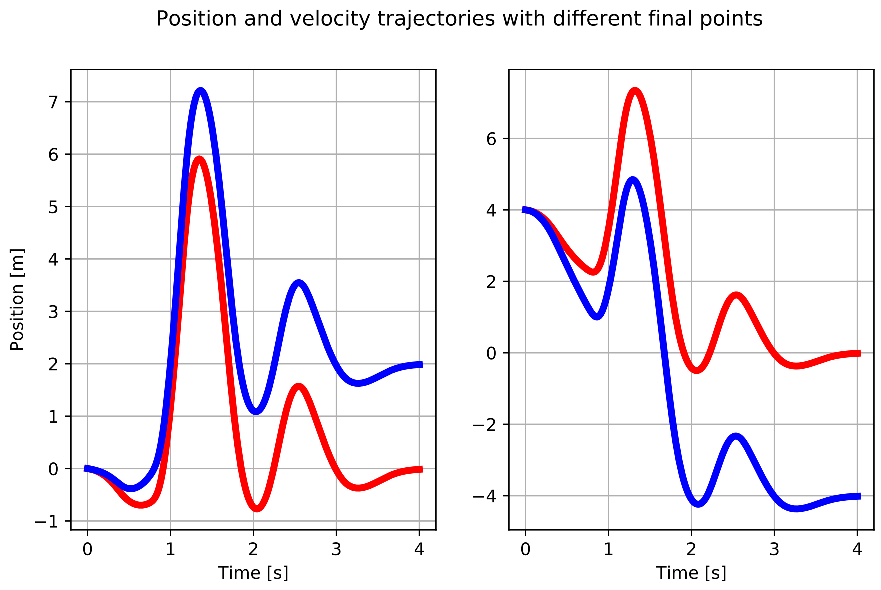


Figure 5.9: Trajectories with two different target points, respectively 3.0 and 0.0 (right) and in the case the target point is in the opposite direction with respect to the DMP direction of the motion, respectively 0.0 and −4.0.

3.2 Tuning Of The Spring-Damper System

To ensure the convergency to the target point a good tuning of the PD controller is crucial. Figure 5.10 shows the plot of the output trajectory and the PD controller with the values of K_p and K_d respectively of 100 and 20, which guarantee a fast asymptotic convergency without any kind of overshooting.

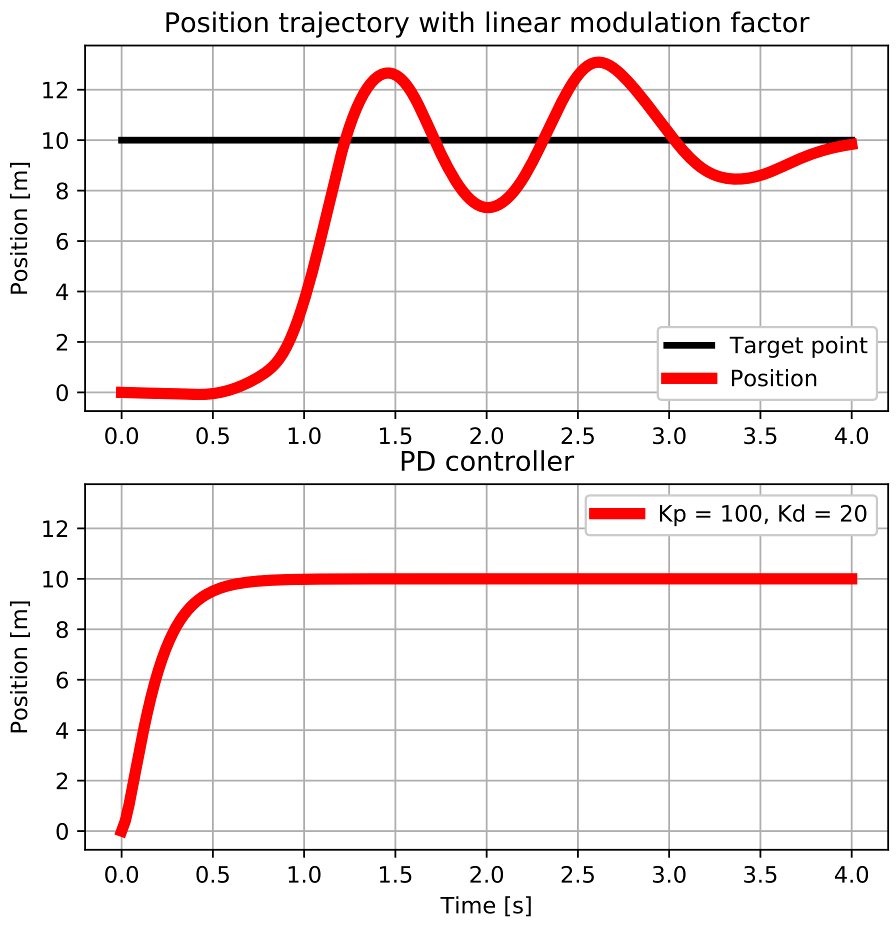


Figure 5.10: Modulated output trajectory converging to the target point (Up) and tuned PD control law ensuring the asymptotic convergency (Down)

Those values were chosen because a less aggressive control law could fail with the convergence of the modulated output trajectory. As illustrated in figure 5.11 with the values $K_p = 20$ and $K_d = 10$, the target point on the output trajectory was not reached on time.

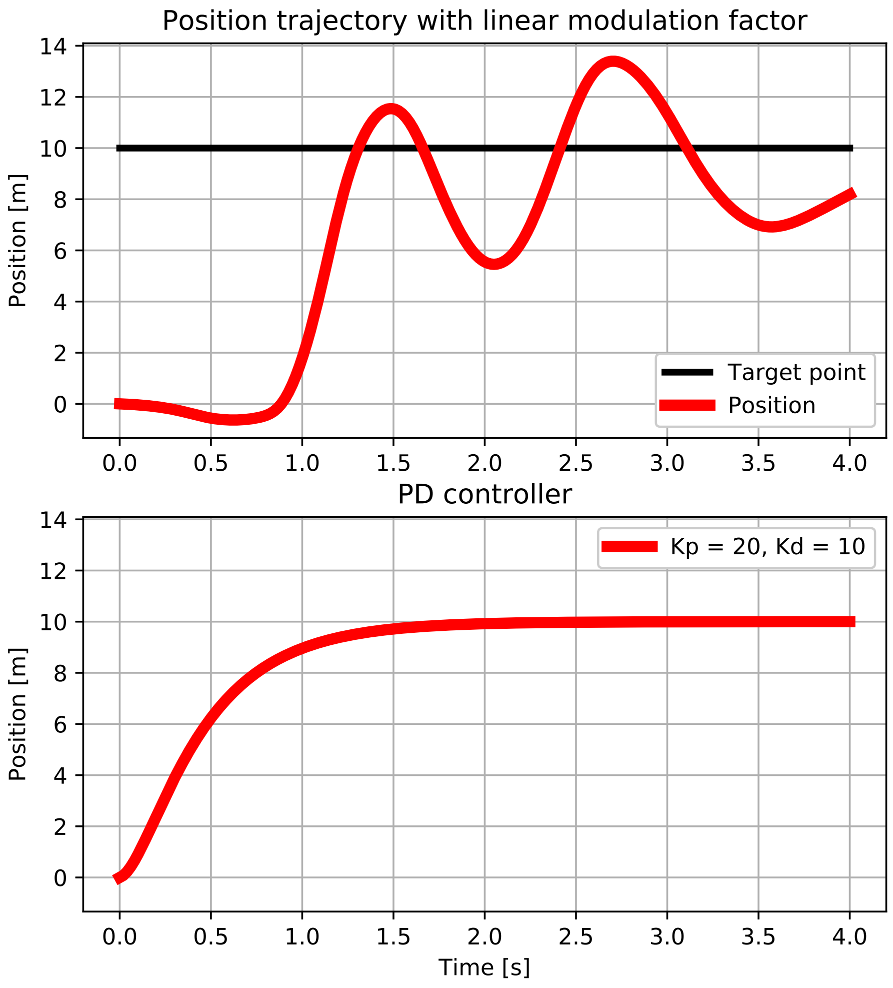


Figure 5.11: With a less aggressive control, the modulated output trajectory (UP) does not converge to the target even if the PD control law (Down) does converge.

3.3 Differences On The Modulation Factor γ

This subsection compares the three types of modulation factors γ explained above. In particular, sliding from the GMR trajectory to the PD control law trajectory in different ways could bring completely distinct results.

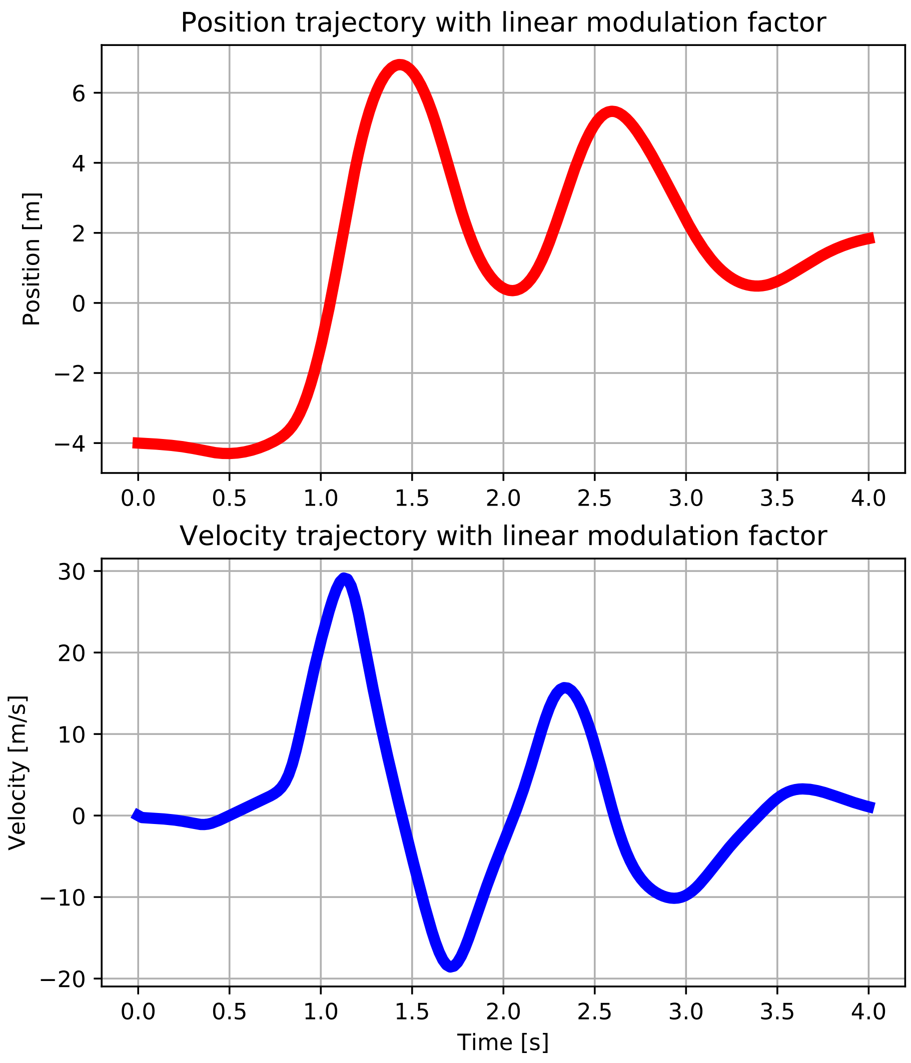


Figure 5.12: Output position (Up) and velocity (Down) trajectories using a linear modulation factor .

Figures 5.12, 5.14 and 5.13 illustrate the position and velocity trajectories with the three different modulation factors, respectively linear, parabolic and exponential explained in section 2.

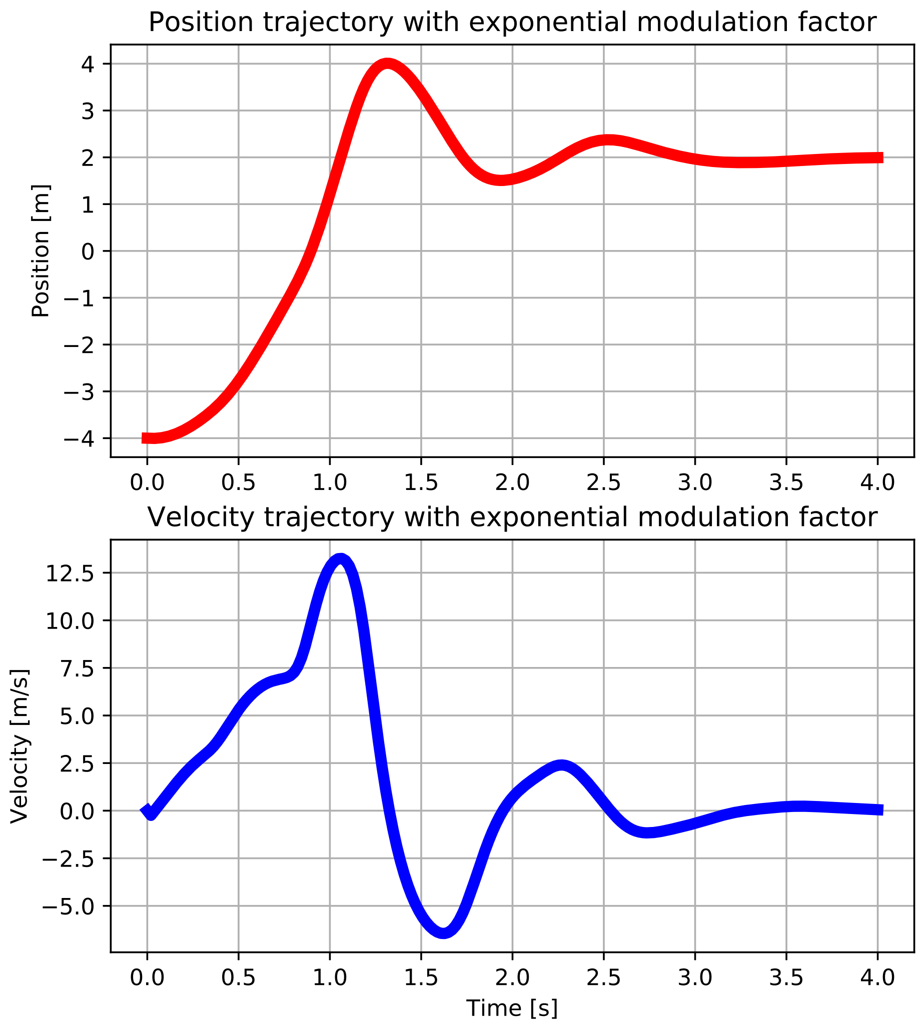


Figure 5.13: Output position (Up) and velocity (Down) trajectories using an exponential modulation factor.

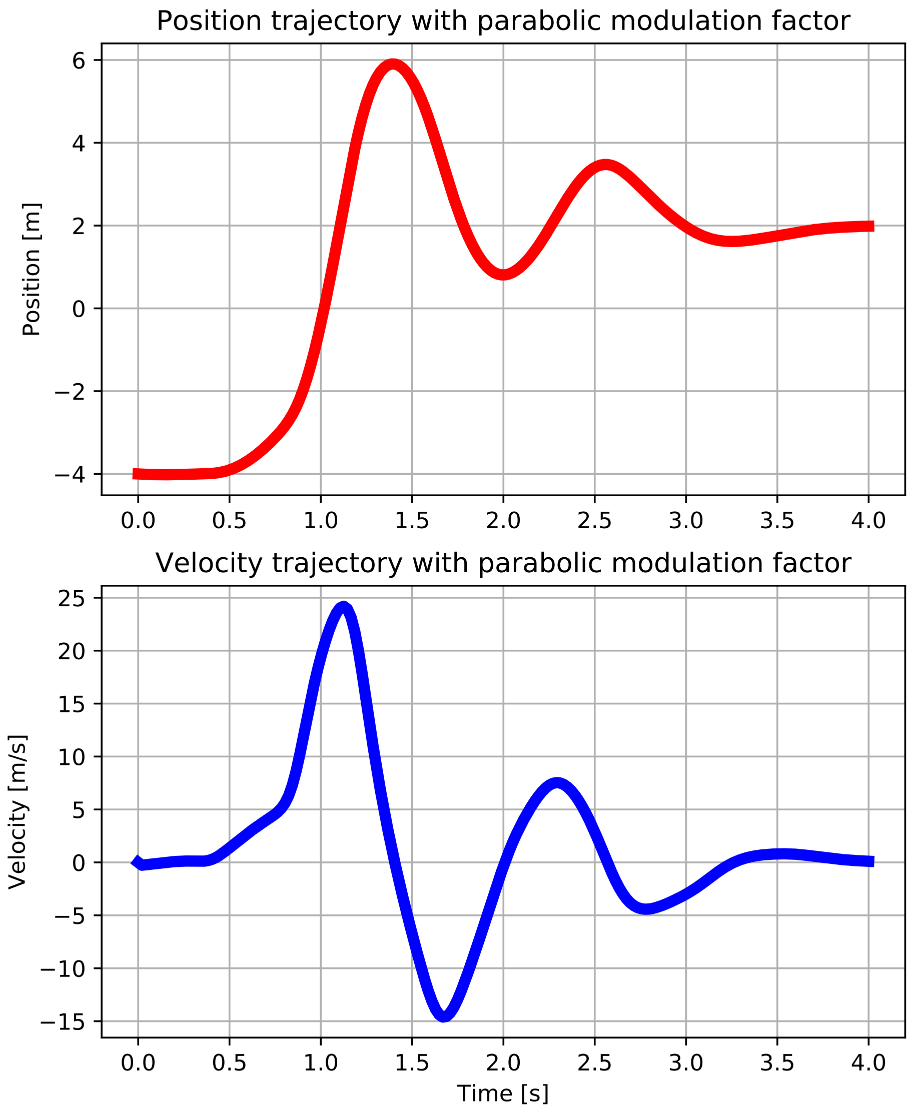


Figure 5.14: Output position (Up) and velocity (Down) trajectories using an exponential modulation factor.

Figures show that each result can have some advantages but also some drawbacks.

- The linear modulation factor maintains the GMR profile for a long time but if the target is too far from the initial point, the convergence is not guaranteed even if the PD controller is well-tuned with an aggressive control.
- The exponential modulation factor is too aggressive from the convergence side. The DMP from the GMR is followed only at the very beginning of the trajectory.
- The parabolic modulation factor is a good compromise between the first two. It guarantees convergence to the target while keeping the DMP from the GMR.

The drawback of the linear modulation factor is shown in figures 5.15 in which the modulated output trajectory is not able to reach the target.

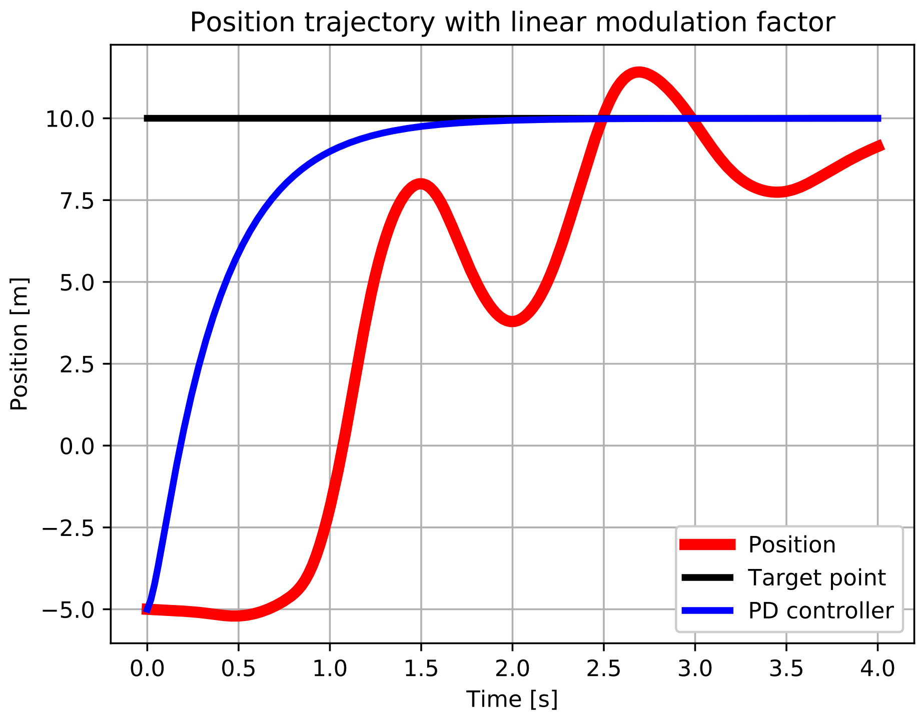


Figure 5.15: Possible drawback of the linear modulation factor. The modulated output trajectory (red) is not able to reach the target (black) even if the PD control (Blue) is well-tuned.

3.4 Robustness To Obstacle

To check the robustness of the presented control law, some straightforward obstacle avoidance applications were attempted by changing online the target point simulating an obstacle in the middle of the workspace. Figures 5.16 and 5.17 show the output trajectories changing the target respectively from 8.0 to 12.0 and from 8.0 to 4.0. In both cases, to move as more as possible towards the control law trajectory the exponential modulation factor was utilised.

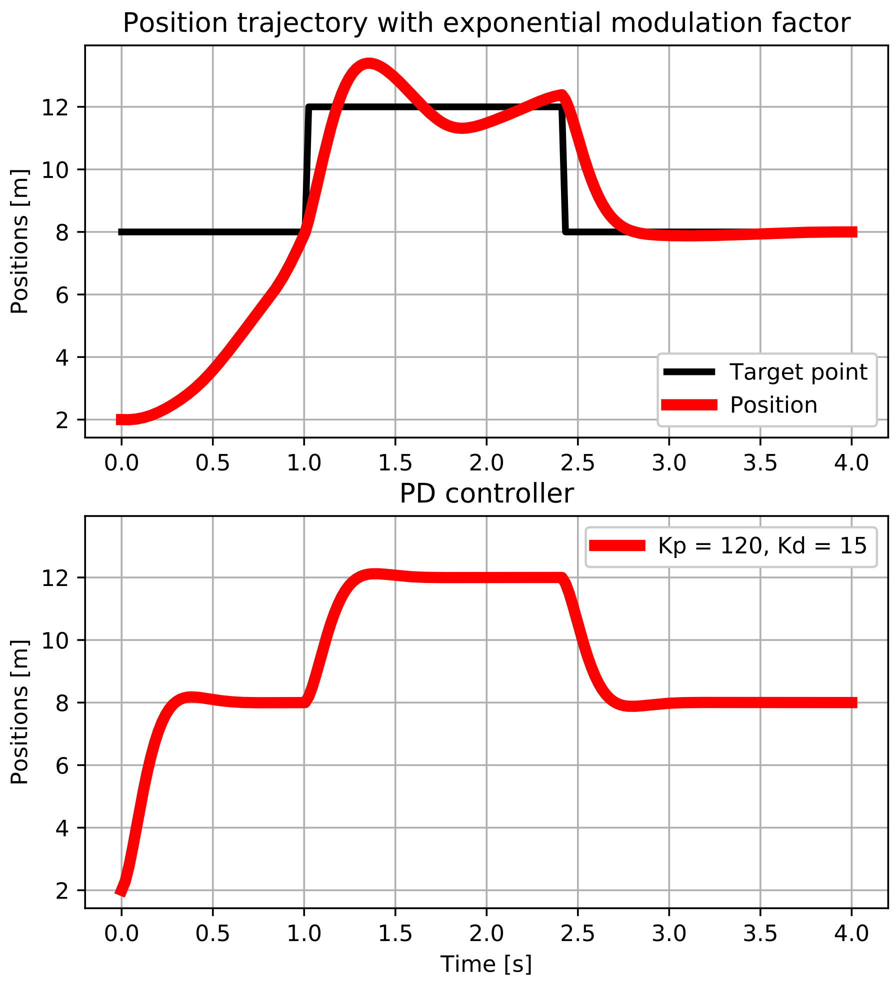


Figure 5.16: Output modulated trajectory in case on a obstacle.

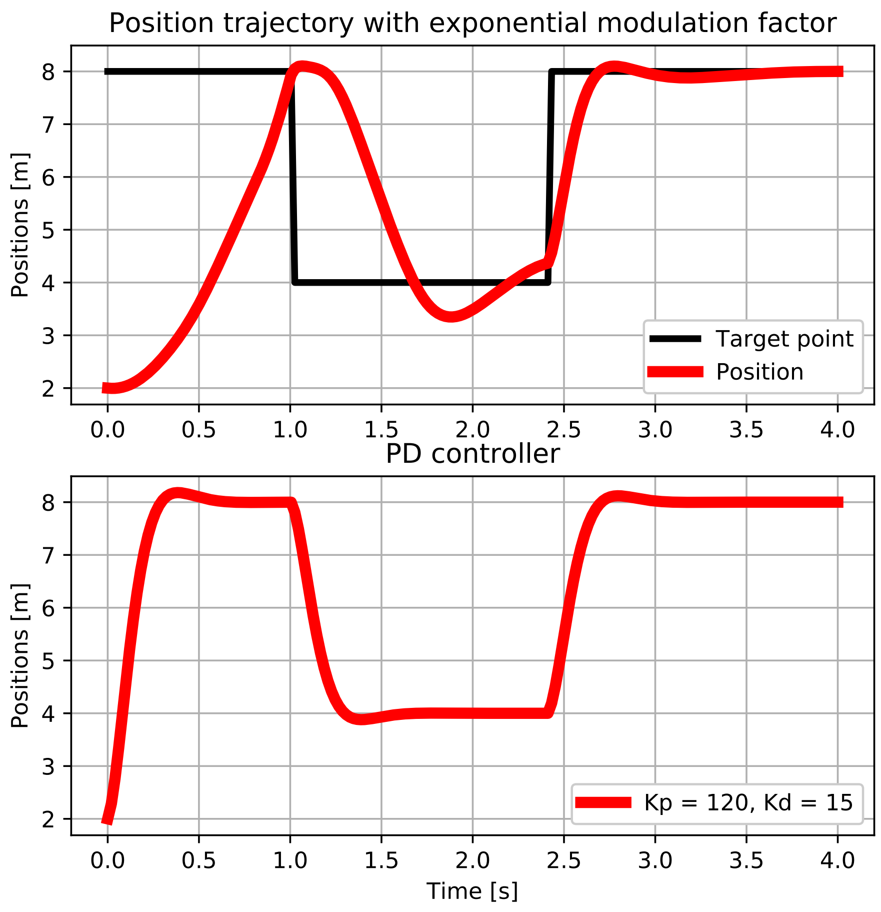


Figure 5.17: Output modulated trajectory in case on a obstacle.

Chapter 6

Real Case Scenario: Teleoperation And Learning Tests

This Chapter will present a practical experiment with the aim of teaching the dual-arm Robot Baxter a new job, such as grasping a Bottle placed on a table with one hand and subsequently pouring the liquid into a glass held by the other hand. The job is divided into three tasks, namely:

- Grasp the bottle;
- Reach the glass;
- Pouring of the water.

The job was performed as a state machine, in which each task is a state characterized by its own control loop.

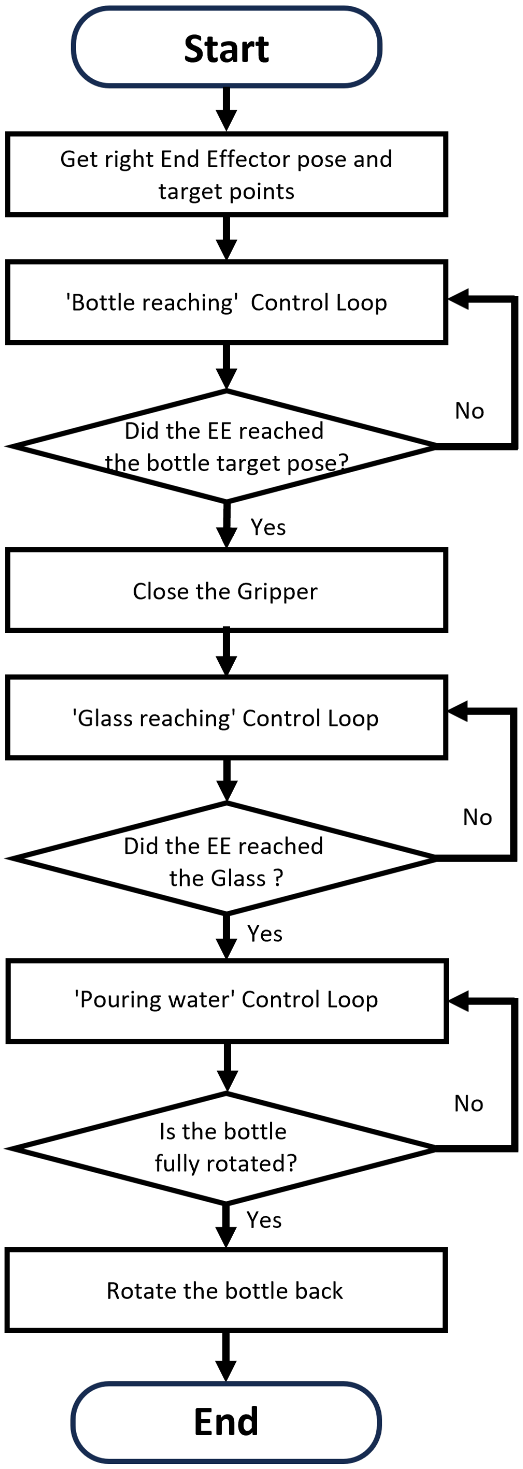

Start
Get right End Effector pose and target points
'Bottle reaching' Control Loop
Did the EE reached the bottle target pose?
No
Yes
Close the Gripper
'Glass reaching' Control Loop
Did the EE reached the Glass ?
No
Yes
'Pouring water' Control Loop
Is the bottle fully rotated?
No
Yes
Rotate the bottle back
End

Figure 6.1: Entire task Flow Chart

1 Imitation Learning Via Teleoperation

Following the passages described in chapter 5, the first objective is to demonstrate to the robot the correct execution of the task. Each task was performed by the human operator via teleoperation fourteen times, while the end-effector pose and its velocity were recorded. Each demonstration started from a different initial pose and terminated at a different point. To obtain that, the position of the bottle on the table was slightly changed every time. According to 2.1, the end effector pose was recorded as position and quaternion orientation, namely for each time instant i:

$$pose_i = [p_x,\ p_y,\ p_z,\ q_x,\ q_y,\ q_z,\ q_w]$$

On the other hand, the velocity was recorded in terms of cartesian velocity and RPY rate, namely:

$$pose_i = \left[v_x,\ v_y,\ v_z,\ \dot{R},\ \dot{P},\ \dot{Y}\right]$$

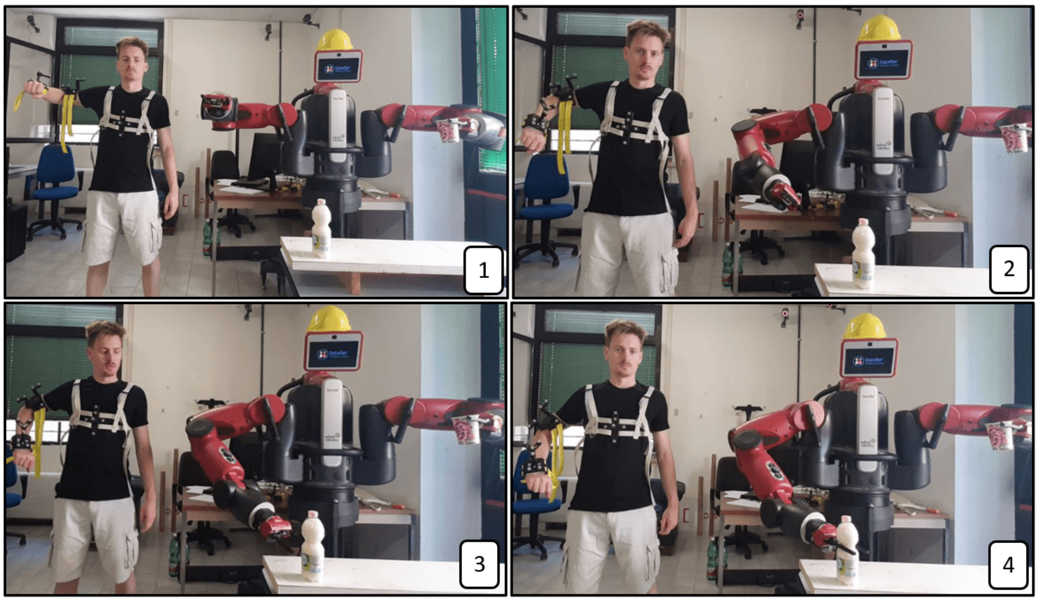


Figure 6.2: Teleoperation of the task 'Grasp the bottle'.

Figures 6.2 and 6.3 show the demonstration through intuitive robot manipulation of the entire job. Figure 6.2 exhibited the sequence of the teleoperated task 'Grasp the bottle', while in figure 6.3 the sequence of both the other tasks, such as 'Reach the glass' and 'Pouring of the water'.

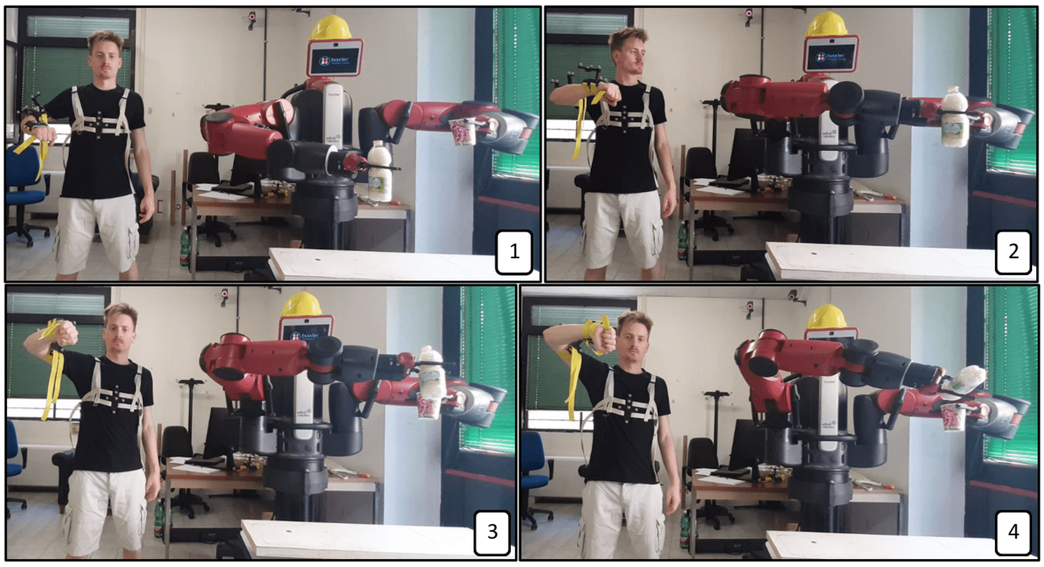


Figure 6.3: Teleoperation of the tasks 'Reach the glass' and 'Pouring of the water'.

All the recorded poses and velocities trajectories of the end effector during the teleoperation, according to the algorithm described in chapter 5, are necessary to perform GMR. Figure 6.4 shows the X coordinate position and velocity trajectory recorded.

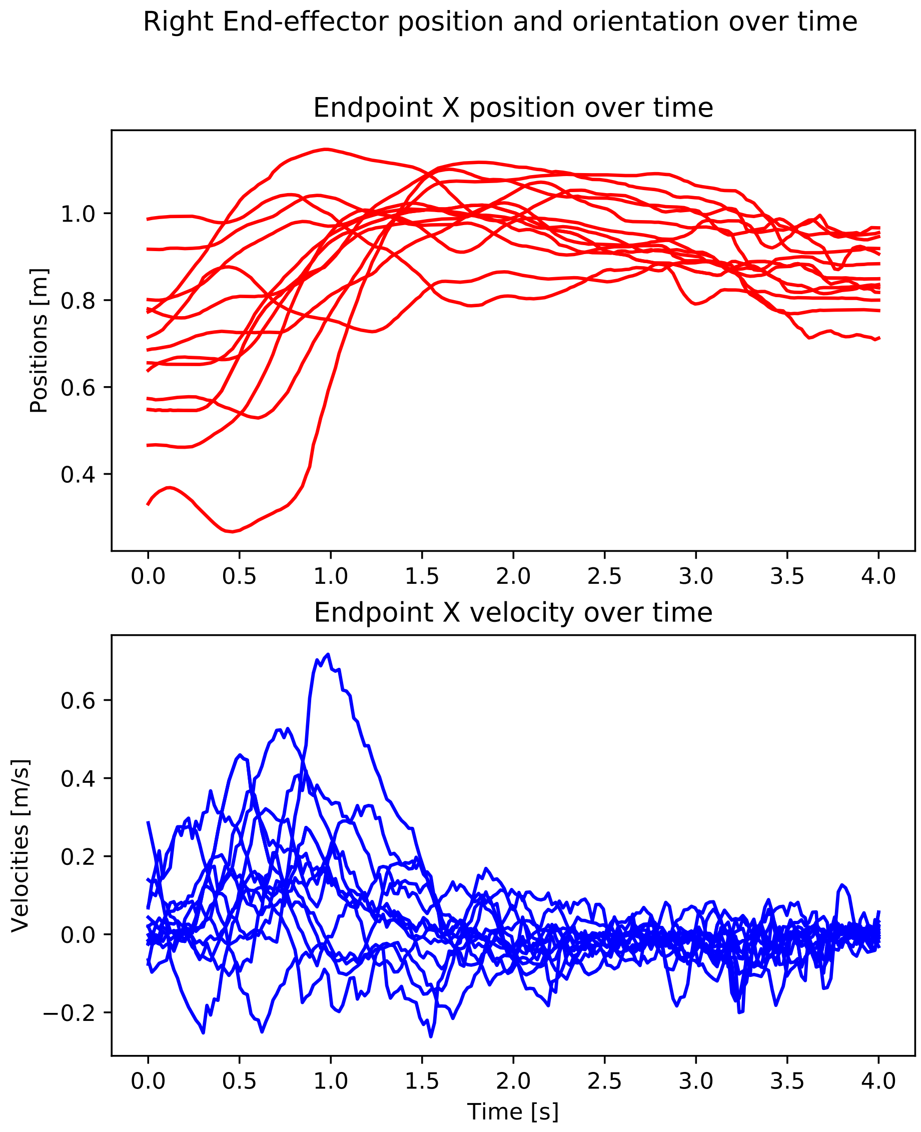


Figure 6.4: Examples of X position and velocity sets from the teleoperation demonstrations.

1.1 Gaussian Filter

As it is possible to notice in Figure 6.4, the demonstrated trajectories present a lot of noise, due to teleoperation errors. In digital signal processing, a Gaussian filter is a filter having an impulse response which is an approximation of a Gaussian function used for designing a finite impulse response digital filter

[21], in the form:

$$G(u) = e^{-u^2}$$

Mathematically, this filter modifies the input signal by convolution with a Gaussian function. The result illustrated in Figure 6.5 is a set of smoother trajectories without noise. To perform the 1D Gaussian filter the python library *scipy.ndimage.gaussianfilter1d* [22] was used.

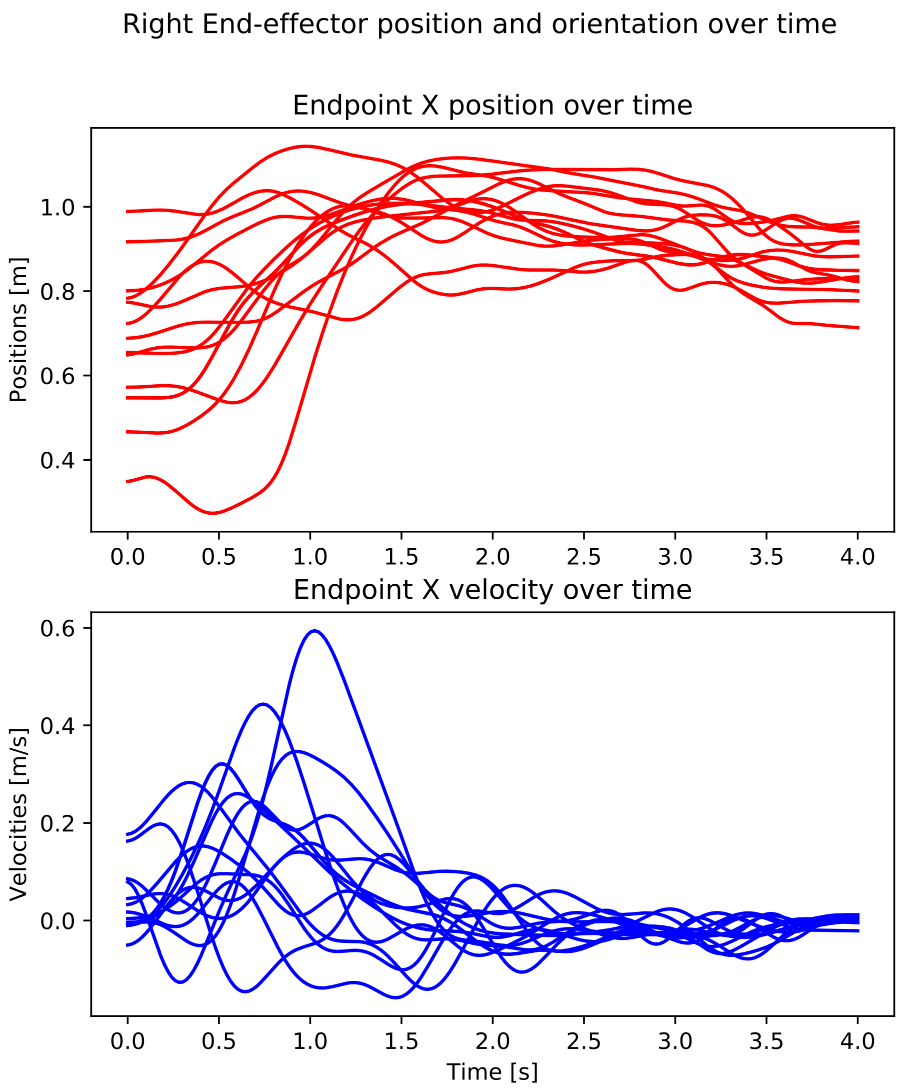


Figure 6.5: Examples of X position and velocity sets Gaussian filtered to eliminate noise.

2 GMR Computation For Each Task

In this section are described the Gaussian Mixture Regression computation for each one of the three tasks of the overall job namely 'Grasp the bottle', 'Reach the glass' and 'Pouring of the water'. The GMR was computed for each velocity component, both for position and orientation.

$$Velocity\ components = \left[v_x,\ v_y,\ v_z,\ \dot{R},\ \dot{P},\ \dot{Y}\right] \tag{6.1}$$

For each component of 6.1 the GMR was computed according to the calculations in chapter 5.

2.1 Grasp The Bottle

In Figure 6.6 are illustrated the GMR of the velocity of the cartesian components, $\dot{x}$, $\dot{y}$ and $\dot{z}$.

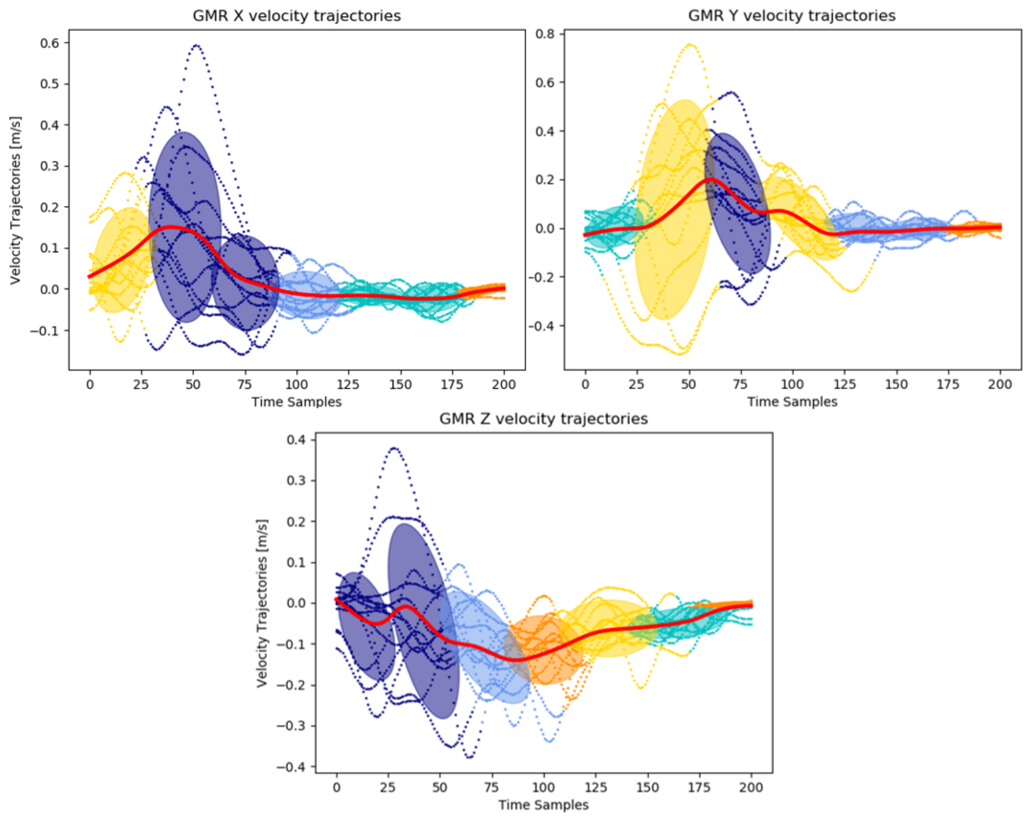


Figure 6.6: GMR of the velocity of the cartesian components in the bottle grasping task.

To compute the output modulated velocity of equation 5.16, a target cartesian point was initialized as $[X^g,\ Y^g,\ Z^g]$, coinciding with the bottle cartesian position on the table. Finally, a spring-damper system was computed for each component as in Equation 5.15 to asymptotically converge to the target. Figure 6.7 illustrates the GMR of the velocity of the three orientation components represented in Roll Pitch and Yaw velocity. $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$.

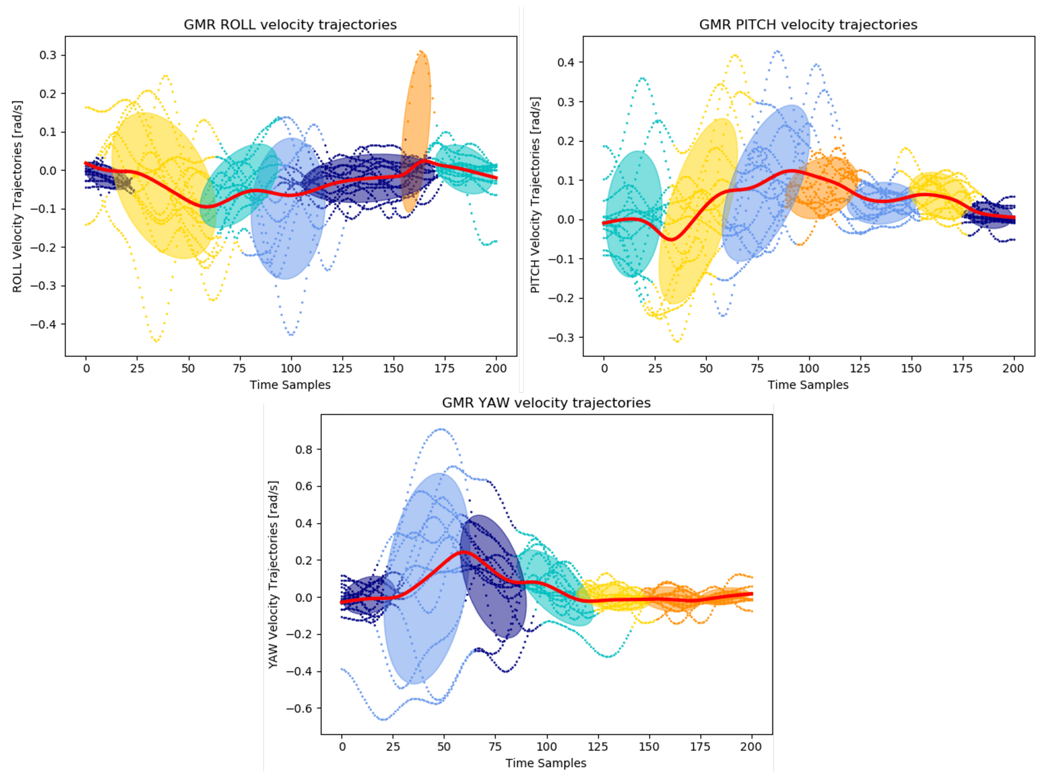


Figure 6.7: GMR of the velocity of the RPY components in the bottle grasping task.

To obtain a feasible orientation trajectory, at each time instant, the RPY velocity was transformed into quaternion velocity as $\dot{q}_x$, $\dot{q}_y$, $\dot{q}_z$ and $\dot{q}_w$, using the *Scipy* Python library [12], considering intrinsic rotations along xyz axes, in a similar method of Chapter 4. Subsequently, a modulated spring damper system was designed for each independent component of the vectorial part of the quaternion, see Equation 3.1, namely q_x, q_y and q_z. To compute the output modulated velocity of equation 5.16, a target quaternion point was initialized as $\left[\dot{q}^g_x,\ \dot{q}^g_y,\ \dot{q}^g_z,\ \dot{q}^g_w\right]$, coinciding with the desired quaternion orientation that the robot should reach to grasp the bottle on the table. The last quaternion component q_w is computed according to the unit quaternion definition of Equation 3.2, as:

$$q_w = 1 - \sqrt{q_x^2 + q_y^2 + q_z^2} \tag{6.2}$$

2.2 Reach The Glass

In Figure 6.8 are illustrated the GMR of the velocity of the three cartesian components, $\dot{x}$, $\dot{y}$ and $\dot{z}$.

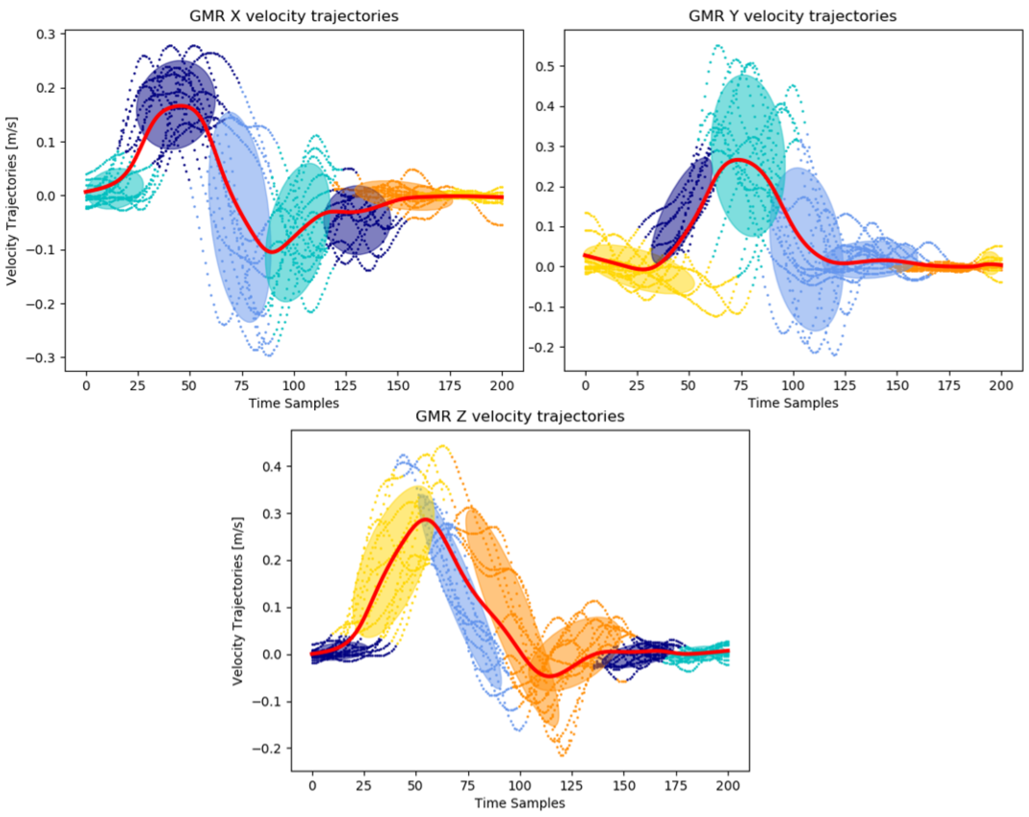


Figure 6.8: GMR of the velocity of the cartesian components in the glass reaching task:

To compute the output modulated velocity, as in the previous case, a new target cartesian point was initialized as $[X^g,\ Y^g,\ Z^g]$. To find the target position the glass cartesian position, which coincides with the left robot end effector, is translated by a vector that represents the bottle encumbrance. In this case $[X^g,\ Y^g,\ Z^g]$. The purpose of this translation is to avoid collision between the two end effector and to centre the glass with the bottleneck during the pouring task, see Figure 6.10. Finally, a spring-damper system was computed for each component as in Equation 5.15 to asymptotically converge to the target.

Figure 6.9 illustrates the GMR of the velocity of the three orientation components represented in Roll Pitch and Yaw velocity. $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$.

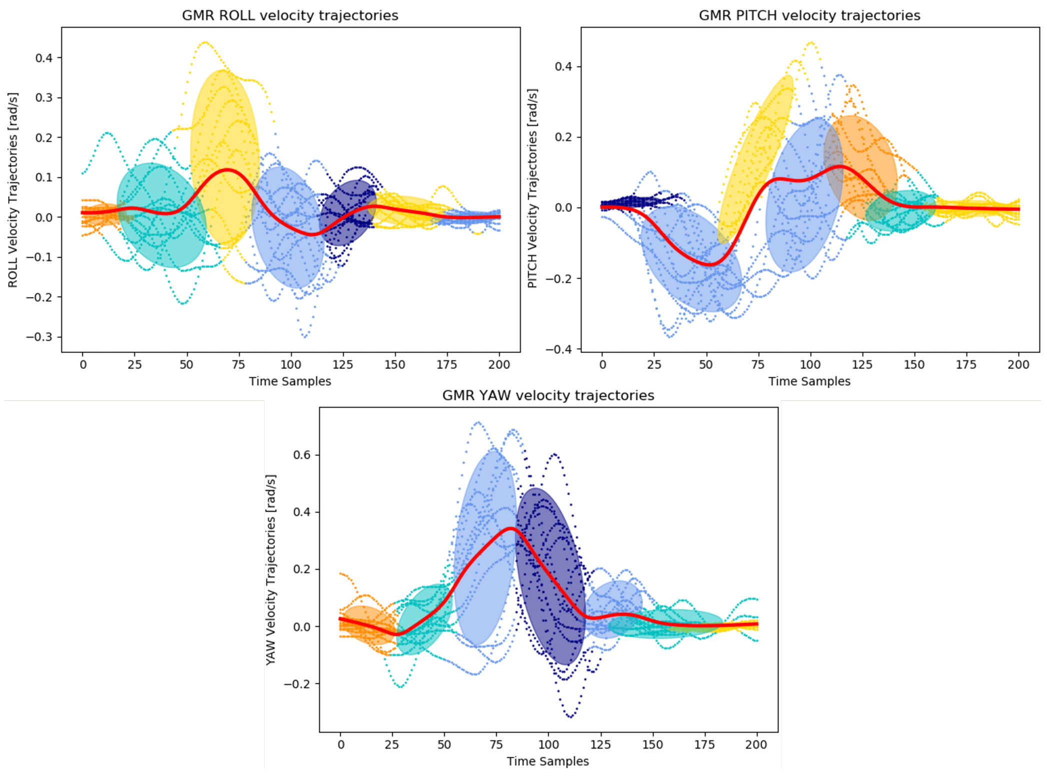


Figure 6.9: GMR of the velocity of the RPY components in the glass reaching task.

As in the previous case, the RPY velocity was transformed into quaternion velocity as $\dot{q}_x$, $\dot{q}_y$, $\dot{q}_z$ and $\dot{q}_w$. Subsequently, a modulated spring damper system was designed for each independent component of the vectorial part of the quaternion q_x, q_y and q_z. To compute the output modulated velocity of equation 5.16, a target quaternion point was initialized as $\left[\dot{q}^g_x,\ \dot{q}^g_y,\ \dot{q}^g_z,\ \dot{q}^g_w\right]$. In this task, the target orientation point was computed by implying a rotation of 90 along the X axes. According to Equation 4.4, the orientation of the robot left end-effector can be found from the ROS TF as a rotation matrix. Calling $R^B_{EE_L}(\eta, \epsilon)$ the rotation matrix from frame *Base* to the Frame *End Effector Left*:

$$\mathbf{R^B_{EE_L}}(\eta,\epsilon) = \begin{bmatrix} 2(\eta^2+\epsilon_x^2)-1 & 2(\epsilon_x\epsilon_y-\eta\epsilon_z) & 2(\epsilon_x\epsilon_z+\eta\epsilon_y) \\ 2(\epsilon_x\epsilon_y+\eta\epsilon_z) & 2(\eta^2+\epsilon_y^2)-1 & 2(\epsilon_y\epsilon_z-\eta\epsilon_x) \\ 2(\epsilon_x\epsilon_z-\eta\epsilon_y) & 2(\epsilon_y\epsilon_z+\eta\epsilon_x) & 2(\eta^2+\epsilon_z^2)-1 \end{bmatrix}$$

Therefore the target orientation can be computed as a rotation matrix from frame *Base* to the Frame *End Effector Right*, by rotating $R^B_{EE_L}(\eta,\epsilon)$ of $\alpha = 90$ along the X axis as follow:

$$\mathbf{R^B_{EE_R}}(\eta,\epsilon) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \mathbf{R^B_{EE_L}}(\eta,\epsilon) \tag{6.3}$$

Once the new rotation matrix is calculated, the target orientation point can be calculated by computing the quaternion corresponding to a given rotation matrix [9], following the rotation matrix notation of equation 4.5 as follow:

$$\eta = \frac{1}{2}\sqrt{r_{11}+r_{22}+r_{33}+1}$$

$$\epsilon) = \frac{1}{2}\begin{bmatrix} sgn(r_{32}+r_{23})\sqrt{r_{11}-r_{22}-r_{33}+1} \\ sgn(r_{13}+r_{31})\sqrt{r_{22}-r_{33}-r_{11}+1} \\ sgn(r_{21}+r_{12})\sqrt{r_{33}-r_{11}+r_{22}+1} \end{bmatrix}$$

Lastly, once the converging trajectories of q_x, q_y and q_z were computed the last quaternion component q_w can be found according to the unit quaternion definition, see Equation 6.2.

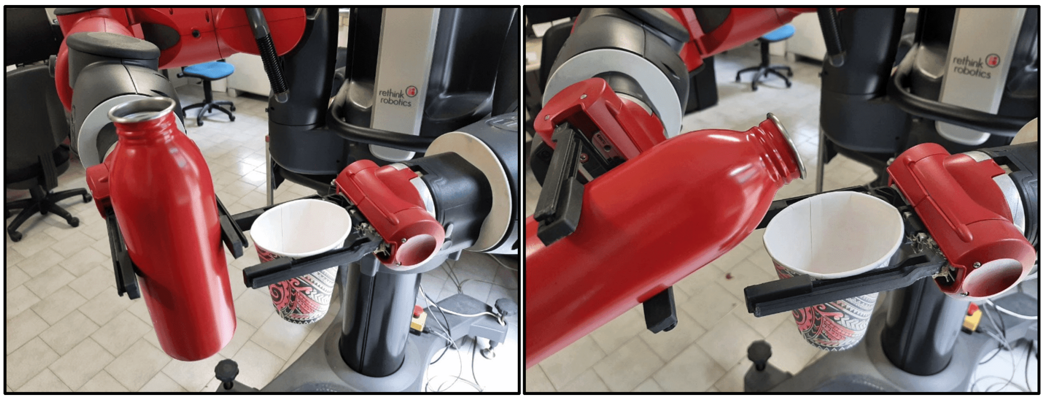


Figure 6.10: Relative position between the bottle and the glass.

2.3 Pour The Liquid

In Figure 6.11 are illustrated the GMR of the velocity of the three cartesian components, $\dot{x}$, $\dot{y}$ and $\dot{z}$.

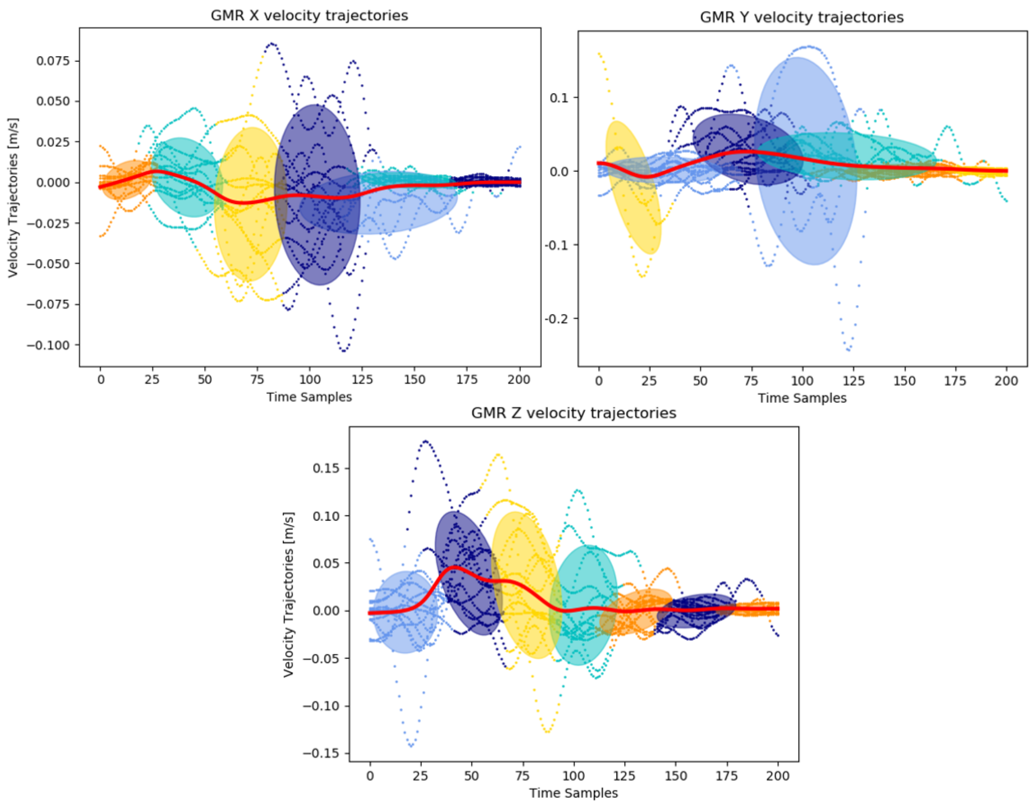


Figure 6.11: GMR of the velocity of the three cartesian components in the pouring task.

The modulated velocity was then computed as in the previous cases. As it is possible to notice from Figure 6.11, exclusively the Z component presents a relatively high peak of velocity obtained from the pouring demonstrations, due to the necessity to keep the bottle steady along the X and Y directions during the pouring task.

Figure 6.12 illustrates the GMR of the velocity of the three orientation components represented in Roll Pitch and Yaw velocity. $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$.

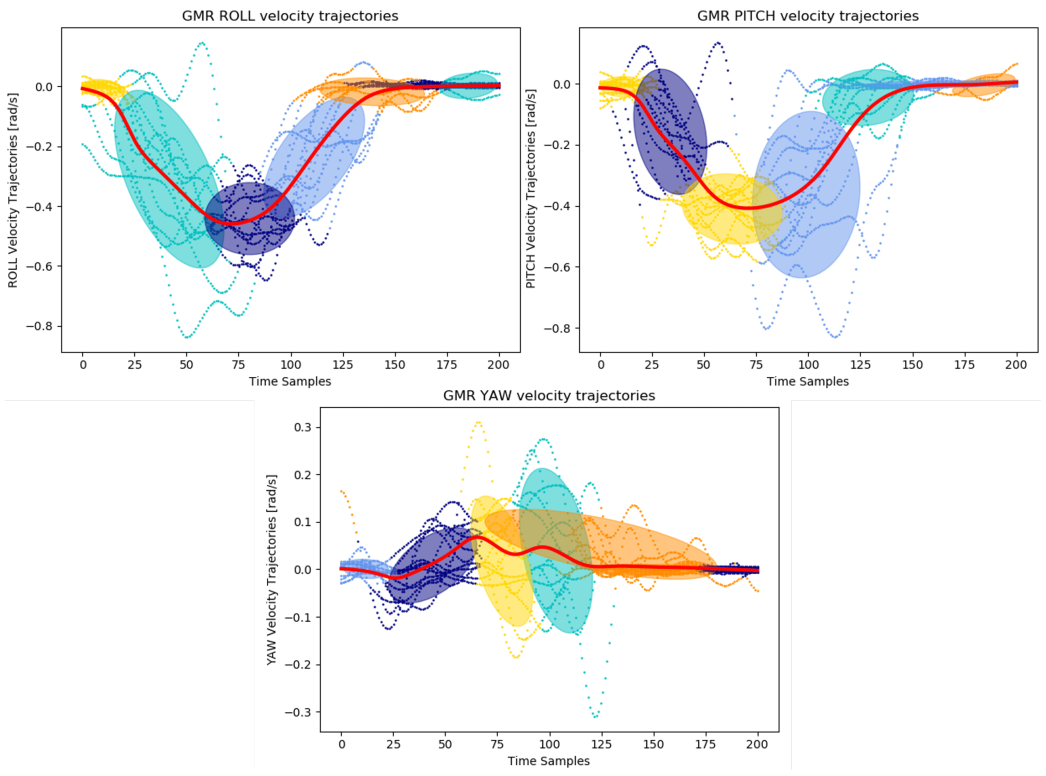


Figure 6.12: GMR of the velocity of the RPY components in the pouring task.

To perform the pouring of the water, the target quaternion point was obtained by rotating the current end-effector orientation of $\alpha = 90$ along the Z axis. The rotation was calculated as in Equation 6.3, by considering a different rotation matrix as follows:

$$\mathbf{R^{B}_{EE_R}}(\eta, \epsilon) = \begin{bmatrix} cos(\alpha) & -sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{R^{B}_{EE_L}}(\eta, \epsilon)$$

3 Overall Task Trajectory

By applying the PbD algorithm explained in the previous section, the entire job was completed in a robust manner. As notable in Figures 6.13 and 6.14, the end effector reaches the bottle position, grasps the bottle in a safe mode by reaching the target point with a translation only along the Z direction and then points towards the glass where it performs the pouring of the liquid, with a single smooth rotation.

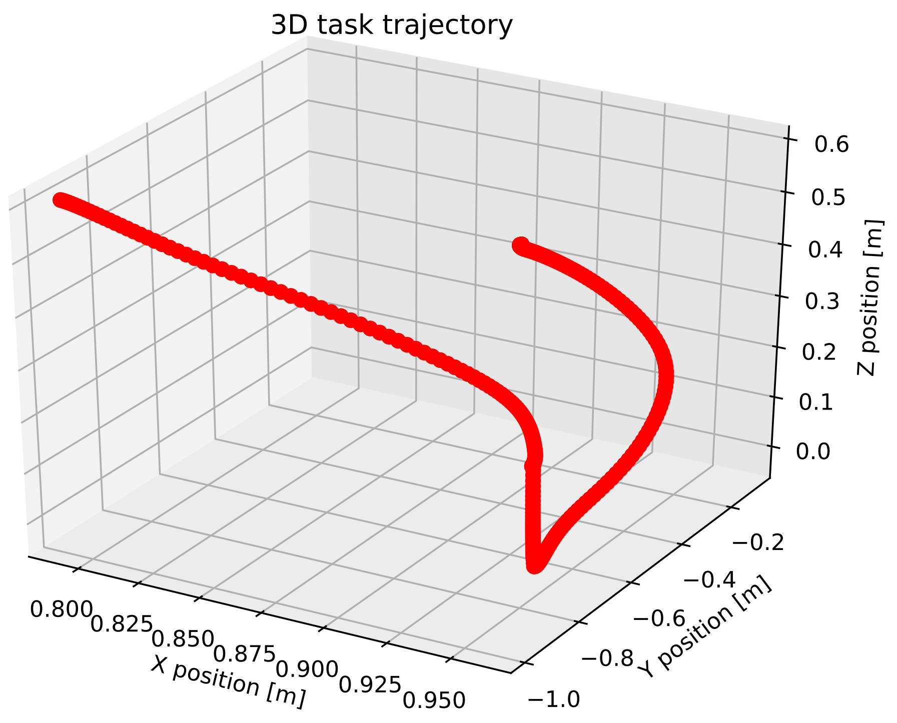


Figure 6.13: Entire task trajectory 3D plot

Figure 6.14 shows the evolution of each component trajectory over time. It is possible to notice that from the time instant 8 seconds to the end, the cartesian position is kept constant while the orientation is changing. This time interval is the ’pouring of the water task’. It is required to keep the position steady to not spill the water on the ground.

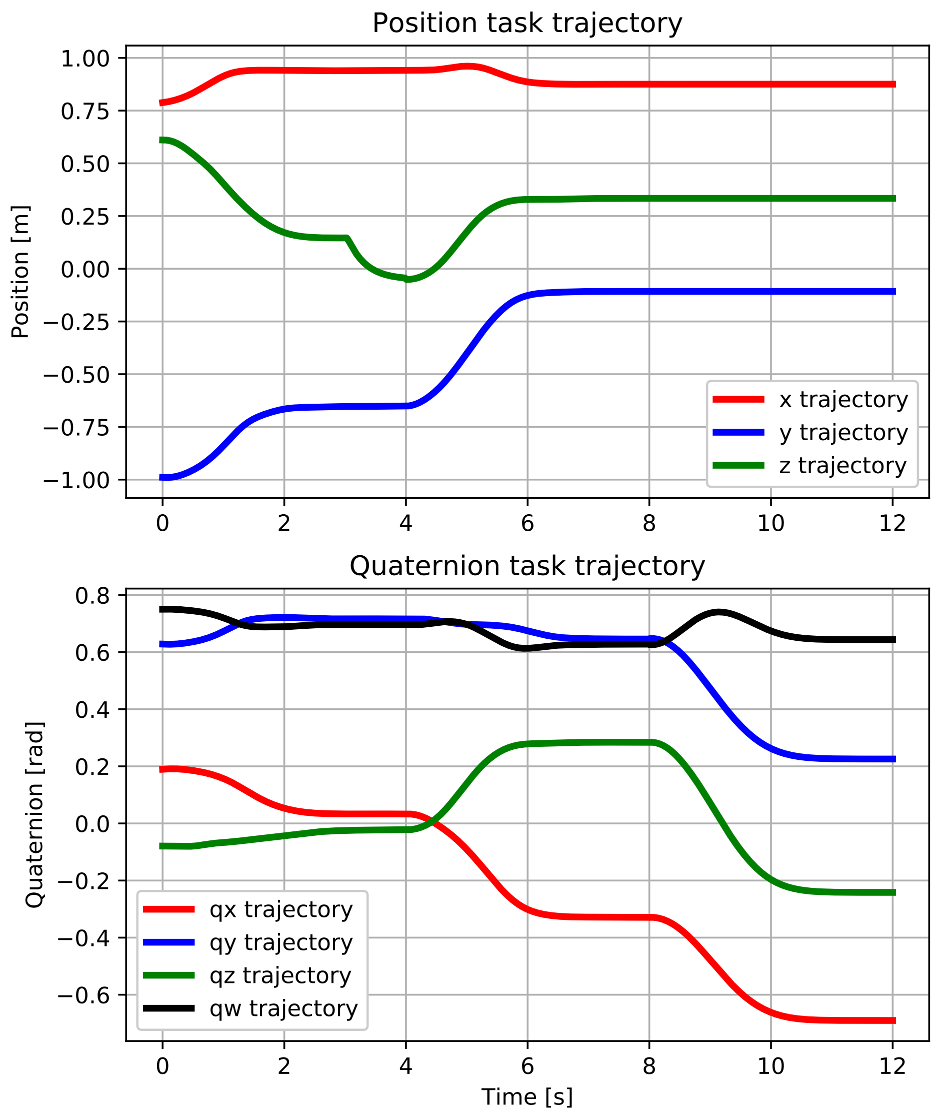


Figure 6.14: Entire task position trajectory (Up) and entire task orientation trajectory (Down)

3.1 Adaptation To Variations Of Initial Or Final Conditions

To validate the result of Chapter 5 about the Robustness to variations of initial conditions the control algorithm was tested starting from different positions within the robot workspace. Figure 6.15 illustrates four different trajectories all converging to the same final point, which in this case is the bottle position.

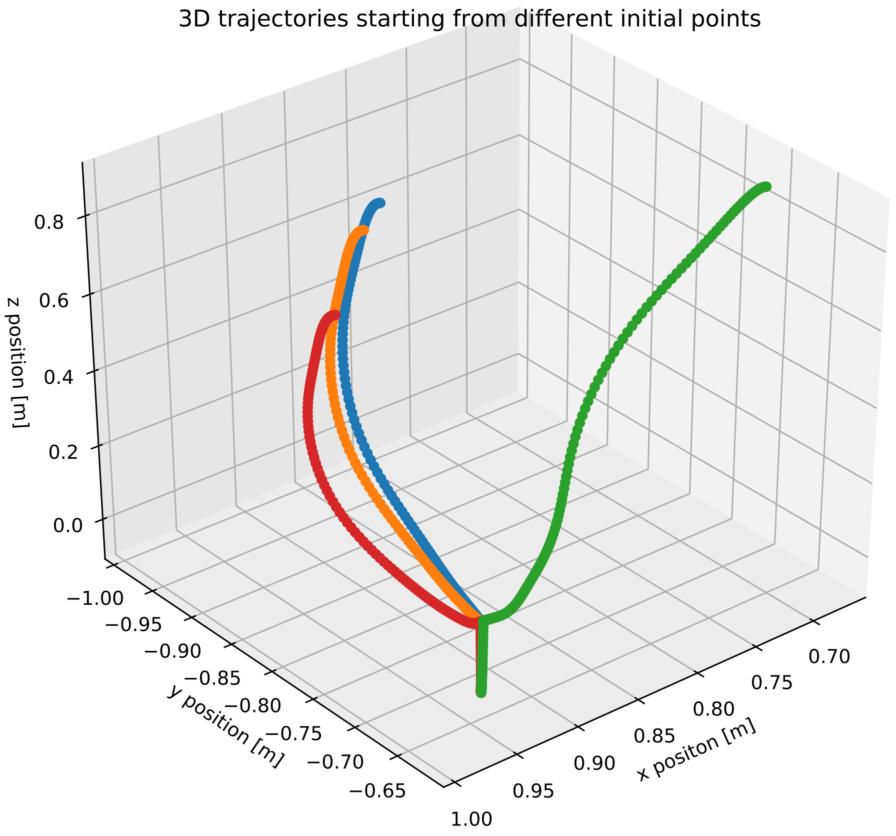


Figure 6.15: 3D trajectories starting from different initial points.

The same validation about the algorithm robustness can be observed by changing the target point. In Figure 6.16 are shown four different trajectories converging to distinct target points. This was easily obtained by changing the end-effector position of the robot's left limb.

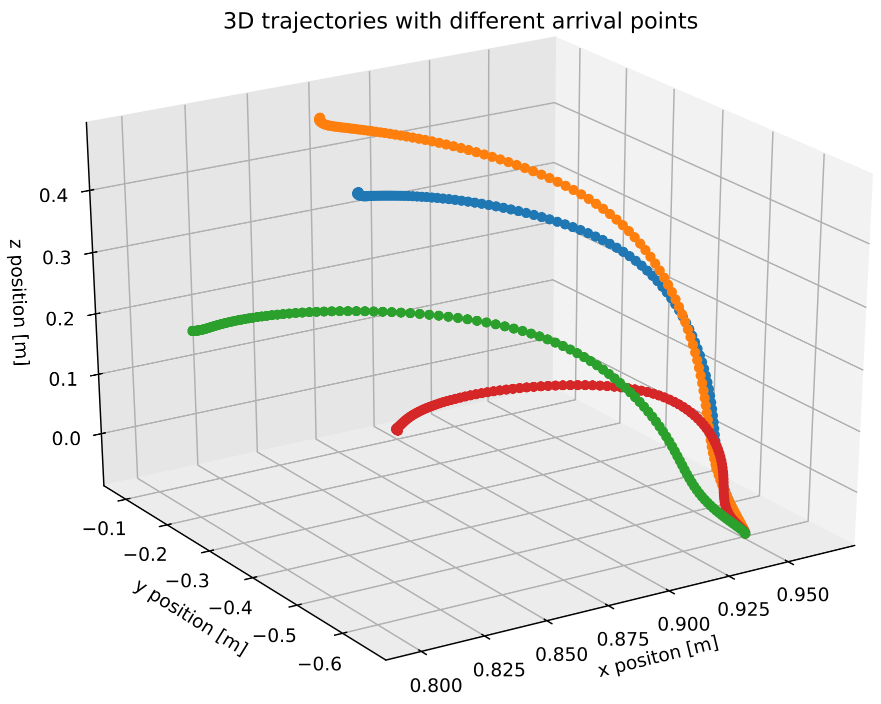


Figure 6.16: 3D trajectories with different target points.

3.2 Practical Results

As illustrated in Figure 6.17 the dual-arm Baxter robot is able to correctly replicate the job in complete autonomy. The only external information needed is the position of the bottle in the workspace. Figure 6.17 sequence $1-4$ shows the bottle grasping sub-task, while the sequence $5-8$ the glass reaching and the water pouring sub-tasks.

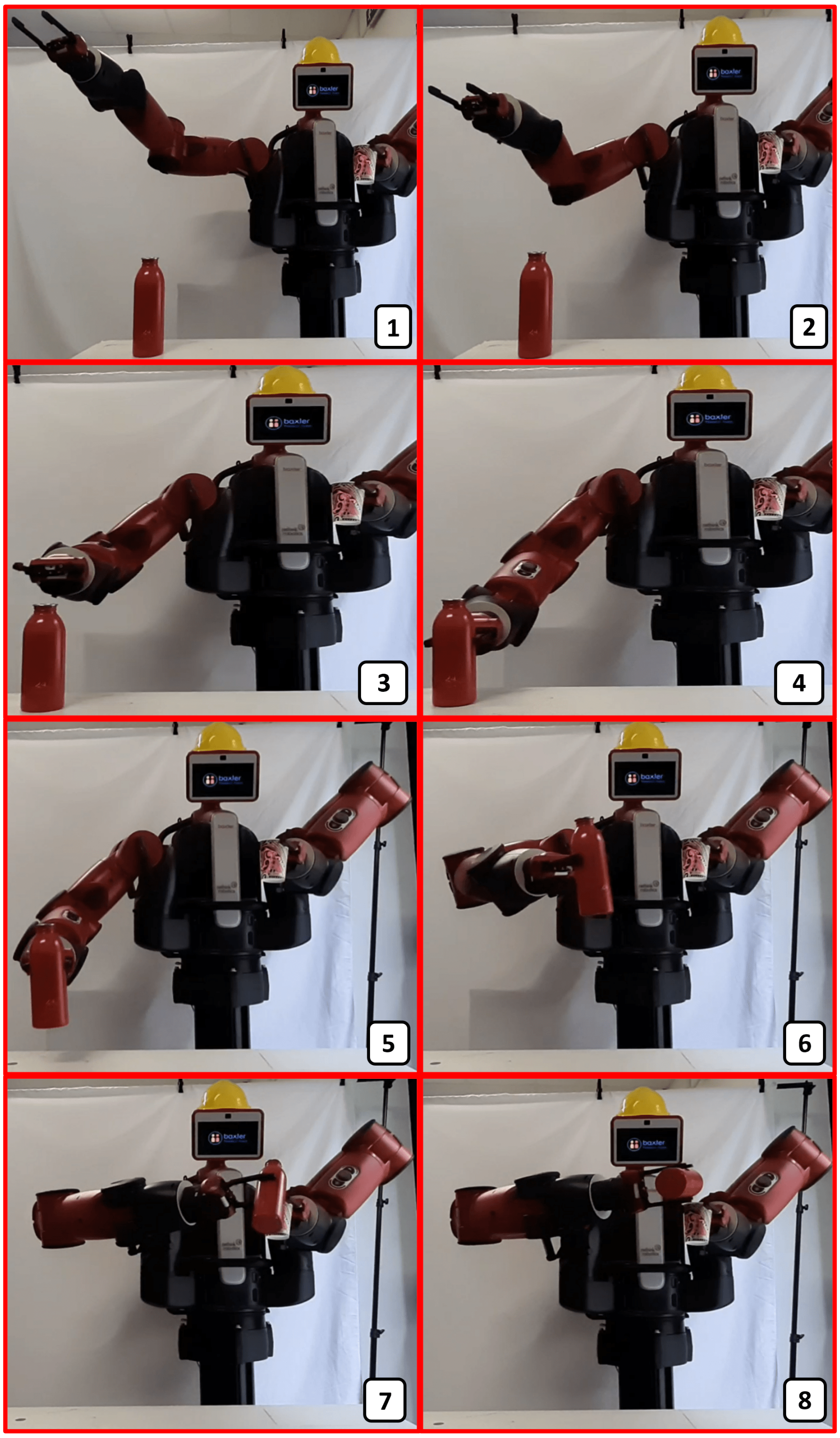


Figure 6.17: Learned job performed by the robot in complete autonomy

Chapter 7

Conclusion

The objective of this thesis was to perform Imitation Learning to teach a dual-arm robot a new task starting from a human demonstration. The objective task was the grasping of a bottle placed on a table with one hand and then pouring the liquid into a glass held by the other hand. To achieve so, in this thesis project two different setups for the interactive telemanipulation of a humanoid bimanual robot are proposed, used for the development of basic actions like reaching and grasping some objects, etc. In particular, a Baxter robot made by Rethink Robotics was controlled. The solution developed is based on ROS, an open-source operating system specialized for robotic applications, and the Vicon Tracker, a tracking system based on infrared sensors, to track the position and orientation in the space of an object. In this case, it was used to track motion in the space of the human upper limbs. Teleoperation was used as an instrument to achieve Learning from Demonstration (LfD), also called Programming by Demonstration (PbD), a machine learning-based procedure for the extraction of a control law starting from the demonstration of a task performed by a human. Following the Learning Control approach, a Dynamical Movement Primitive (DMP) was generated as a reference velocity profile of the task. Gaussian Mixture Regression (GMR) is a stochastic method used in this thesis to train a Gaussian Mixture Model (GMM) from a collected Training set and then extract the velocity model of the system. A spring-damper system was then used to guarantee the asymptotic convergence of the trajectory to the target point. A modulation factor γ was used to slide from the DMP to the convergent control law. In such a way, the human operator is able to teach the robot a new task using teleoperation showing it the right movements to perform. The robotic system was therefore able to learn how to replicate that task in an autonomous and robust manner, insensitively to disturbances of the environment and with possible different starting or arrival points.

In future works, the teleoperation platform can be integrated with alternative end-effectors such as a human-like hand. The grasping will be controlled directly from the user operator through EMG sensors placed on both human forearms. This robotic interface will, according to IntelliMan objectives, be able to learn new tasks using DMP approach exploiting more advaced algorithms, such as Hidden Markow Models (HMM), or RL algorithms. Afterwards, a shared-control telemanipulation system AI-driven will be developed for both prosthetics and robot manipulation applicatoins. The primary objective will then be the development of a configuration that accurately perceives the operator's intentions and responds accordingly, reaching the capability of autonomously executing the task or a perfect co-activity. By implementing the sliding autonomy paradigm [23], the system will exhibit resilience and adaptability in the face of changing circumstances.

Bibliography

[1] *Eu project intelliman*. [Online]. Available: `https://intelliman-project.eu/`.

[2] S. Calinon and D. Lee, "Learning control," *Humanoid robotics: A reference. Springer Netherlands*, 2017.

[3] *Stanford university*. [Online]. Available: `https://www.ros.org/`.

[4] *Rethink robotics*. [Online]. Available: `https://sdk.rethinkrobotics.com`.

[5] *Vicon motion systems ltd*. [Online]. Available: `https://www.vicon.com/software/tracker/`.

[6] T. Foote, "Tf: The transform library," ser. Open-Source Software workshop, Apr. 2013, pp. 1–6. DOI: `10.1109/TePRA.2013.6556373`.

[7] N. I. Balder and D. Tolani, "Real-time inverse kinematics of the human arm," *Teleoperators and Virtual Environments 1996*, Jan. 1996.

[8] J. Graßhoff, L. Hansen, I. Kuhlemann, and K. Ehlers, "7dof hand and arm tracking for teleoperation of anthropomorphic robots," *Proceedings of ISR 2016: 47st International Symposium on Robotics, Munich, Germany*, 2016.

[9] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics - Modelling, Planning and Control*. Springer.

[10] C. D'Ettorre, A. Mariani, A. Stilli, *et al.*, "Accelerating surgical robotics research: A review of 10 years with the da vinci research kit," *IEEE Robotics and Automation Magazine*, vol. 28, no. 4, pp. 56–78, 2021. DOI: `10.1109/MRA.2021.3101646`.

[11] J. Delmerico, S. Mintchev, A. Giusti, *et al.*, "The current state and future outlook of rescue robotics," *Journal of Field Robotics*, vol. 36, no. 7, pp. 1171–1191, 2019.

[12] *The scipy community: Rotation.* [Online]. Available: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.transform.Rotation.html`.

[13] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *IEEE Transactions on Robotics, vol. 24, no. 6, pp. 1463-1467*, 2008.

[14] M. Mühlig, M. Gienger, and J. J. Steil, "Interactive imitation learning of object movement skills," *Autonomous Robots*, 2012.

[15] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with em-based reinforcement learning," *IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 2010, pp. 3232-3237*, 2010.

[16] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gesture by imitation," *in IEEE Robotics and Automation Magazine, vol. 17, no. 2, pp. 44-54*, 2010.

[17] G. Xuan, W. Zhang, and P. Chai, "Em algorithms of gaussian mixture model and hidden markov model," vol. 1, 145–148 vol.1, 2001.

[18] Z. Ghahramani and M. Jordan, "Supervised learning from incomplete data via an em approach," *Advances in neural information processing systems 6*, 1993.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[20] A. G. Billard, S. Calinon, and F. Guenter, "Discriminative and adaptive imitation in uni-manual and bi-manual tasks," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 370–384, 2006.

[21] Y.-B. Yuan, X.-F. Qiang, J.-F. Song, and T. V. Vorburger, "A fast algorithm for determining the gaussian filtered mean line in surface metrology," *Precision Engineering*, vol. 24, no. 1, pp. 62–69, 2000.

[22] *The scipy community: Gaussian filter1d.* [Online]. Available: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.gaussian_filter1d.html#scipy.ndimage.gaussian_filter1d`.

[23] D. P. Losey, C. G. McDonald, E. Battaglia, and M. K. O'Malley, "A review of intent detection, arbitration, and communication aspects of shared control for physical human–robot interaction," *Applied Mechanics Reviews*, vol. 70, no. 1, 2018.