POLITECNICO DI TORINO

Master Degree in Biomedical Engineering

Master's Thesis

Development of a teleoperated hand-arm robotic platform for the evaluation of shared autonomy algorithms

Supervisors:

Prof. Carlo Ferraresi
Prof. Marco Controzzi
Ing. Angela Mazzeo

Student:

Manuela Uliano

Academic year 2020/2021

Abstract

Humans are unable to operate in hazardous or inaccessible environments. However, the need to perform specific tasks in those places and the curiosity to discover unexplored environments have increasingly turned the spotlight on teleoperation. A teleoperation system requires the presence of a remote robot controlled by a human operator. Teleoperation plays a significant role in different areas, like the nuclear industry, mining, underwater exploration, etc. For difficult tasks, it might be easier for the operators to delegate some actions to the robot, thus sharing the autonomy of the task execution with the robot.
The present master's thesis aims at the development of a teleoperation system for the evaluation of shared autonomy algorithms.
The teleoperated system comprises a 6-axis industrial robot (Universal Robot, model UR5) and an anthropomorphic artificial hand (Prensilia SRL, model MIA). The teleoperation system is composed of a wearable inertial motion capture system (Perception Neuron, model PN 32 V2) and a dataglove (CyberGlove, model CyberGlove II) which allow measuring the motion of the operator's arm and hand, respectively. The software has been implemented in ROS (Robot Operating System), which is a flexible framework for writing robot software, using the C++ programming language.
For this application, different teleoperation strategies have been implemented for the hand and the arm.
The dataglove is used to classify the hand closing and opening actions, which are then automatically executed by the artificial hand. The developed decoding algorithm has been tested with different subjects, which were asked to open and close their hand while moving the arm after a short calibration procedure, without errors. The delay introduced by this algorithm is negligible.
The pose of the palm, retrieved by the measurements of the inertial motion capture system, is used to proportionally control the Tool Center Point (TCP) of the robotic arm. An experiment has been carried out to evaluate the performances of the system. Participants were asked to wear the inertial motion capture system and place their right hand on a holder attached to a collaborative robot arm (Universal Robot, model UR5). This robot was programmed to execute precise motions that were then compared to the positions and orientations computed by the algorithm and executed by the teleoperated robot arm. The performances have been evaluated in terms of precision, accuracy, delay and drift.
Finally, a qualitative evaluation of the whole system has been carried out on an industrial use case, showing promising capabilities also for teaching the robot to execute human-inspired tasks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robots emerged in the industrial scene during the '60s: they entered the factories to substitute workers in repetitive or hard to perform tasks. Thanks to technology advancement, robotics had a rapid evolution. An important change has taken place in the early 2000s when start working directly alongside employees with no safety caging. Since then, robots began to be more and more part of our life. Robots have taken place in many everyday environments such as hospital or supermarket (service robots), as well as in industrial settings (industrial robots).
The annual International Federation of Robotics (IFR) press release said: "Robots are clearly on the rise, in manufacturing and increasingly in everyday environments [...] Robotics in professional applications has already had a significant impact in areas such as agriculture, surgery, logistics or public relations and it's growing in economic importance. [...] Future product visions point to domestic robots of higher sophistication, capability and value, such as assistive robots for supporting the elderly, for helping with household chores and for entertainment." ((online retrieved March 16, 2021) [11]
The significant presence of robots in our everyday life leads the research to investigate the human-robot interaction (HRI) in order to understand, design, and evaluate robotic systems for use by or with humans. Human-robot interaction (HRI) is currently a very extensive and diverse research and design activity. Studies on HRI involve different technical disciplines: mechanical and electrical engineering, computer and control science, and artificial intelligence.
Four areas of application of HRI can be identified [12]:

1. Human supervisory control of robots. Robots, called telerobots, are capable of performing a limited series of programmed actions automatically. They can perceive and transmit to the human operator the external environment and their joints configuration.

2. Remote control of vehicles for non-routine tasks in hazardous or inaccessible environments. Robots perform tasks in remote environments thanks to the control of the remote human. Such machines are called teleoperators.

3. Automated vehicles that require the presence of the human being as a passenger. Among these vehicles are included automated highway and rail vehicles and commercial aircraft.

4. Social robots to provide entertainment, teaching, comfort, and assistance for children, elderly, autistic, and disabled people.

1.1 Teleoperation

The primary aim of a teleoperation system is to provide means to correctly execute the desired task in a remote environment. Depending on the application, the involved distance could vary from kilometres to centimetres.
All existing teleoperated systems are composed of two main parts: the operator environment and the remote one. The communication channel between both environments permits the conveyance of commands from the operator to the remote devices and the transmission of the remote task informations back to the operator. Typical communication channels are Ethernet, Internet, wireless, radiofrequency, and even satellite communication.

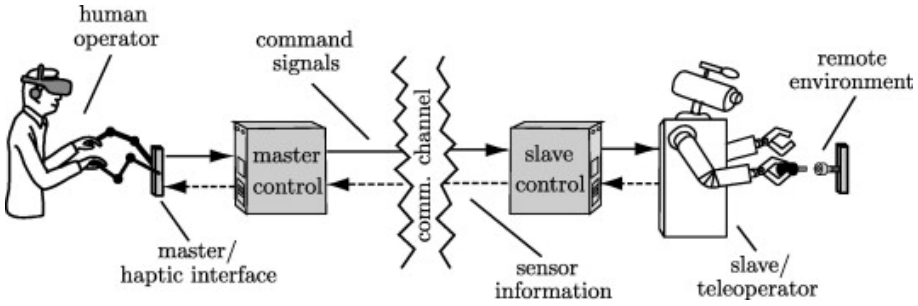


Figure 1.1: Haptic teleoperation system [1].

The operator monitors and provides high-level commands. The *operator environment* is composed of a multi-modal human system interface used to excite the operator senses to show the status of the performed task in the remote environment, and to process the operator commands. The typical human exited sense is the vision, as well as the sense of hearing and touch. The operator has the possibility to chose different types of interfaces to express the desired robot motion or action; a multi-modal interface could be chosen according to the specific application.
The *remote environment* includes teleoperated devices, sensors and objects that contribute to the teleoperation task. It is possible to know the state of the remote system, such as robot angles, distances, and positions, thanks to the sensors. The physical actions are performed in the remote zone by the actuators. They consist in electrical, hydraulic, pneumatic and magnetic motors.
The coupling between the human operator and the remote robot can be weak or strong. An example of weak coupling is a human giving commands to the robot by pushing buttons and watching the resulting action; the strong coupling occurs in a bilateral teleoperation scenario instead. Therefore, the level of coupling is related to the control distribution between operator and remote robot controller [13, 14].
It is possible to distinguish three different teleoperation strategies [2]:

1. Closed-loop control (direct teleoperation): the operator controls the actuators of the teleoperator through direct signals and obtains real-time feedback. The operator

and the robot work concurrently. The direct teleoperation is successfully performed only with minimal control loop delay.

2. Collaborative control: also in this case the operator controls the actuators, but now there is some internal control loop in the teleoperator. User and robot share a task and work like a team.

3. Supervisory control: the principal control part is on the teleoperator side. The teleoperator is autonomous and the operator mainly monitors and gives high-level commands.

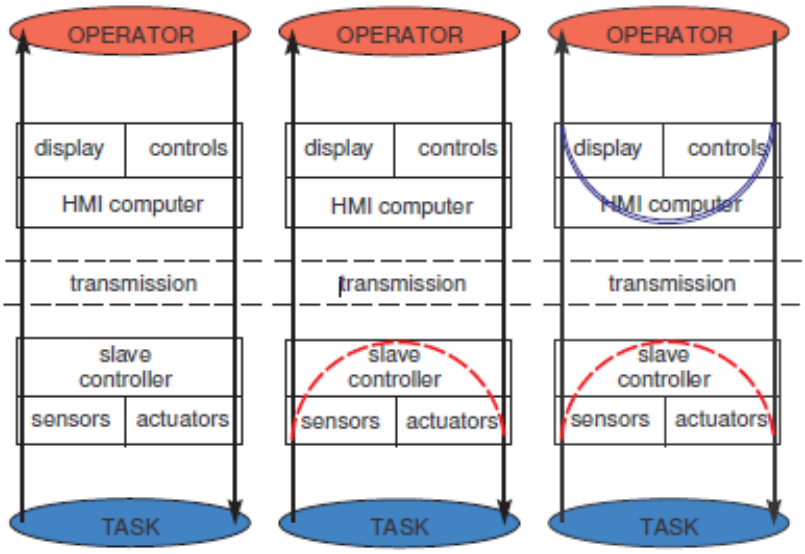


Figure 1.2: The three teleoperation strategies [2].

In the design of a teleoperation system, the main objectives are robustness, the feeling of presence, task performance, and transparency.
In order to design a suitable teleoperation system, the controller is required to be *robustly* stable with respect to uncertainties resulting from the operator, the remote environment, the communication channel, and the sensors.
The teleoperation system has to allow to overcome limitations related to the remote environment and the operator. Remote environment limitations are distance, scale, time delay or hazardousness. An operator limitation is, for example, the hand tremor in a microsurgical application. If all limitations were overcome, the teleoperation system could have high *task performance*. For the evaluation of task performance, the most common quantities are task completion time, error measures, applied forces, induced or dissipated energies.
The *feeling of presence* refers to the operator's feeling of being in the remote environment. In an ideal situation, the operator cannot distinguish between the real world and the remote environment. However, technical limitations make it difficult to reach this state. A telepresence system should enable the operator to feel immersed and involved in the

remote environment. High immersion means that the user have a congruent picture of the remote environment. High involvement means that the user is provided by all stimuli essential to interact with the remote environment. Both conditions means that the operator has a high extent of sensory information to naturally explore and manipulate the remote environment [14].
Depending on the typology of the task, task performance increases while increasing the feeling of presence. Everyday tasks are done almost automatically because the operator is confident with the performed task. For this reason, they don't require a strong feeling of being present in the remote environment. However, for unknown tasks or in unstructured, unknown and changing environments, a strong feeling of presence will help the operator to better perform the task (feeling of presence has a positive effect on task performance). In order to evaluate the feeling of being present, it is possible to consider objective and subjective measures. Objective performance measures are task completion time or reaction time to a remote stimulus. Subjective performance are evaluated with questionnaires asking the human operator about her/his individual feelings.
The feeling of presence is a subjective aim, compared to the *transparency* that is a quantitative aim. With transparency we mean that the technical means between the operator and the environment is not felt [1].

1.1.1 Historical overview

The history of teleoperation can be traced back to the inception of radio communication and Nikola Tesla's findings. At the end of the 19th century, Tesla developed the earliest principles and systems that led to the birth of teleoperation, as reported in the US patent 613809 *Method of and Apparatus for Controlling Mechanism of Moving Vessels or Vehicles*. In the 1945s, Goertz and his colleagues of the Argonne National Laboratory near Chicago developed the mechanical pantograph mechanisms to manipulate radioactive materials. This episode marks the birth of the first master-slave teleoperator.
Ten years later Goertz improved the previous system, introducing Electrical servomechanisms and closed-circuit television. Through the closed-circuit television, the operator could operate at any distance, presenting the telepresence validity.
From the 1960s, telemanipulators and video cameras found space in undersea applications, especially by the U.S., U.S.S.R., and France. Examples of underwater applications were mineral extraction and cable-laying. To perform those tasks the operator could watch the environment through video cameras and interact with it.
During the 60's, researchers were focused on solutions for Moon exploration. The distance from the Earth to the Moon revealed the main problem in teleoperation: the transmission time delay. Since then, people started to study new control loops to avoid instability due to time delay [15].
Technology has evolved, leading to robotic systems and means of communication improvement, from optical wires to the Internet which removes distance limitation.

1.1.2 Application areas of teleoperation

Teleoperation can be applied to different sectors. The most relevant application areas of teleoperation are reported below.

Space

Space can be seen as one of the most important application areas of teleoperation for different reasons: the physical presence of a human in space involves high costs and risks because of the hostile environment. In some cases, like Sun exploration, the presence of a human is impossible at all.

Figure 1.3: Canadarm 2 [3].

Three different space application subgroups are distinguishable: space exploration robots, satellites, and outer-space robot arms [2]. Over the years, there was an evolution in space application: from space shuttle activities, where the operator had the full control of the manipulator, to planetary missions, where the operator is like a supervisor of autonomous telerobots.
The installation of robot arms on Space Station to do tasks outside is notable. From 1981 to 2011, five robot arms have been built for space. One of the most famous is the Canadian Remote Manipulator System (RMS), installed on the US space shuttles.
Although the research in this field is going on, problems related to reliability requirements, weight constraints, hostile environments and communication time delay are still unsolved [16].

Underwater

The first reported application of underwater telerobotics was in 1966 when the U.S. Navy's CURV recovered a nuclear bomb from the ocean.

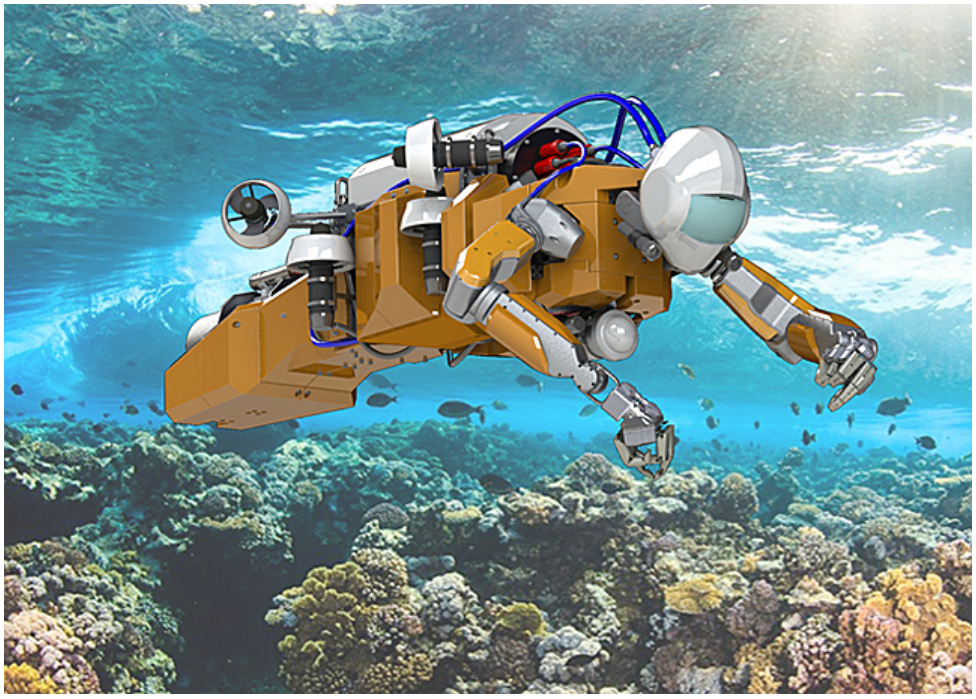


Figure 1.4: Touch-sensitive underwater robotic platform for deep sea coral reefs.

From that time, the use of ROVs (remotely operated vehicles) began to grow for different reasons: the high costs of human pilots, the time required for ascending and descending, the extensive support structures needed, and, sometimes, the risk for human life.
The history of ROVs applications has started in the 1980s for offshore oil and gas industry. Underwater teleoperation started with manned submersibles and evolved to the remotely control of submersibles by human operators; for this application were used long fibre-optic cables for data communication [16].
The ROVs applications expanded to business, military missions, communication (telephone) industry, and scientific investigations carried out by marine geologists, biologists, and archaeologists. Unlike most of the industrial applications, the marine sampling requires delicate manipulation capability to avoid the damages to organisms or fragile samples.
Common problems in this field are related to the water environment: the high pressures, the poor visibility (still unsolved), and the communication difficulties [3].

Surgery

One type of teleoperator is the endoscope employed for investigation, biopsy, and simple surgery into the gastrointestinal tract. Similar devices were inserted into skeletal joints to extract or rebuild tissues or bones.
The teleoperation finds fertile grounds in the surgical field to improve some medical limitations.
First of all, teleoperation have made operations much less invasive. Technological advances in this field lead to the development of minimally invasive surgery techniques. In addition, teleoperated devices permit the surgeon to perform more accurate and repetitive movements, measure forces and they reduce oscillations. One of the most famous robotic

surgeon system is the well-known Da Vinci Robot.
Besides, teleoperation allows expert surgeons around the world to operate without requiring them to travel. In the future, this latter approach will produce saves money, time and effort. In 2001, a team of surgeons from Johns Hopkins University in Baltimore (U.S.) performed a surgical operation on some patients at Rome's Policlinico Casilino University (Italy).
However, the experience proved the necessity of a surgeon presence on-site to solve unexpected complications [2].

Figure 1.5: Robot Da Vinci.

Assistance

Assistive robots especially involve disabled and elderly people. The principal purpose is the improvement of the quality of life of these subjects by supporting independent living and mobility. In this field, teleoperated devices can vary from fixed systems to devices based on wheelchairs or mobile robots.
One of the latest teleoperation application in this field regards the assistance of elderly affected by dementia disorder. Indeed, the healthcare is moving from hospitals to homes thanks to assistive technologies, for two reasons in particular: first, the insufficient number of caregivers, and second to prevent the loss of autonomy for patients.

The teleoperated system proposed in [4] comprises of two main parts: a wearable inertial motion capture device and a dual-arm robot, YuMi. In this way, it is possible for caregivers to remotely take care of older people.
All assistive systems share a common problem: the choice of the Human-Machine Interface. This must be appropriate and intuitive for people with diminished capacity. Typically, HMI involves the use of buttons, voice and gesture recognition, joystick/haptic interface, and so on. Moreover, another relevant problem is related to the safety of the teleoperator, because he shares the workspace with the teleoperated system.

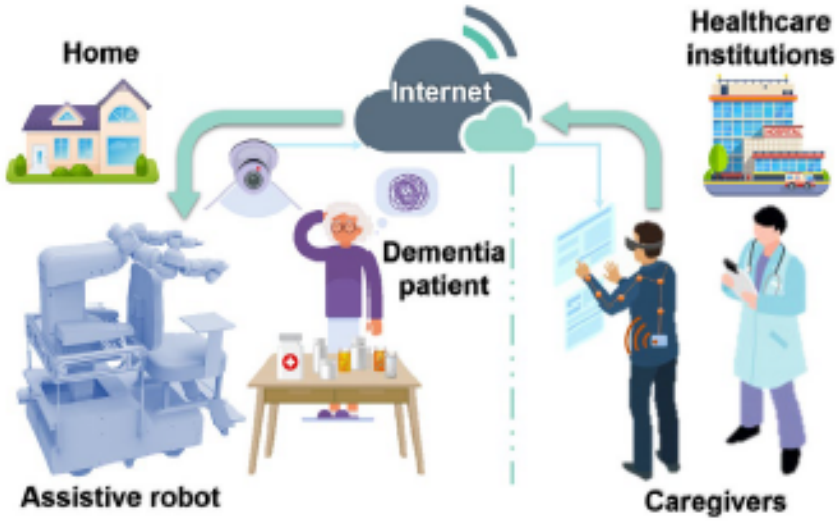


Figure 1.6: The overall architecture of the telerobotic system for remote dementia care presented in [4].

Education

One of the newest application area of teleoperation is education. Sometimes visiting a factory is more expensive and time-consuming for a classroom, but, thanks to teleoperation, the professor can illustrate the theoretical concepts during the lecture using the operation of a remote real plant. Besides, the students can experiment and practice in remote experimental plants from everywhere and at different times. This make possible to develop a lot of remote laboratory projects to teach arguments of different engineering fields [3].

Other applications

Teleoperation finds application also in other areas:

- Military
- Industry and construction
- Mining
- Humanitarian Demining

Military applications include reckoning missions, useful for gathering information about an enemy. In order to avoid the loss of humans, air and ground teleoperated reckon vehicles have been developed [2].
In *industrial* applications, teleoperation is useful for inspection, repair, and maintenance operations in places like power plants. Another type of application is the maintenance of electrical power lines, an hostile location for humans.
An additional application of teleoperation systems regards the nuclear industry. In this

case, teleoperation permits to limit the exposure of human workers to the radioactive environment for a long time. In nuclear plants, teleoperation systems replace human during dangerous activities, such as the maintenance of nuclear reactors, decommissioning and dismantling of nuclear facilities, and emergency interventions.
Another interesting application of teleoperation is *construction*. Typical tasks are earth-moving, compaction, road construction and maintenance, and trenchless technologies.
A different and fascinating field of application for teleoperation is *mining*. Humans are exposed to risk during underground drilling operation, and sometimes mines are almost inaccessible. The most common devices used for teleoperation in mining are load-haul-dump (LHD) machines and thin-seam continuous mining (TSCM) machines, which can work in a semiautonomous and teleoperated way. In this environment, interference can be a problem depending on the mine material and it can lead to video feedback with poor quality. An additional problem is the position measurement, needed for control, when the vehicle is beneath the surface. In order to overcome these problems the use of gyroscopes, magnetic electronic compasses, and radar to locate the position of the vehicles are introduced .
Finally, it is interesting presenting another application related to *humanitarian demining*. Land mines are very easy to place, but very hard to be removed. Teleoperated robots have been developed to reduce the high risk that exists when this task is performed by humans. An important aspect of teleoperated devices for demining is the robustness of the remote station to resist a mine explosion. The remote station has to be also cheap enough to minimize the loss when the manipulation fails and the mine explodes. Because of the complexity of the task, the teleoperation systems include not only teleoperated robotic arms but also teleoperated robotic hands [3].

Figure 1.7: Robot ROBOTET for maintenance of electrical power lines [3].

1.1.3 Teleoperation in the battle against COVID-19

The World Health Organization (WHO) on January 30, 2020, declared the SARS-CoV pandemic as a "global emergency". The first cases have been identified in Wuhan City,

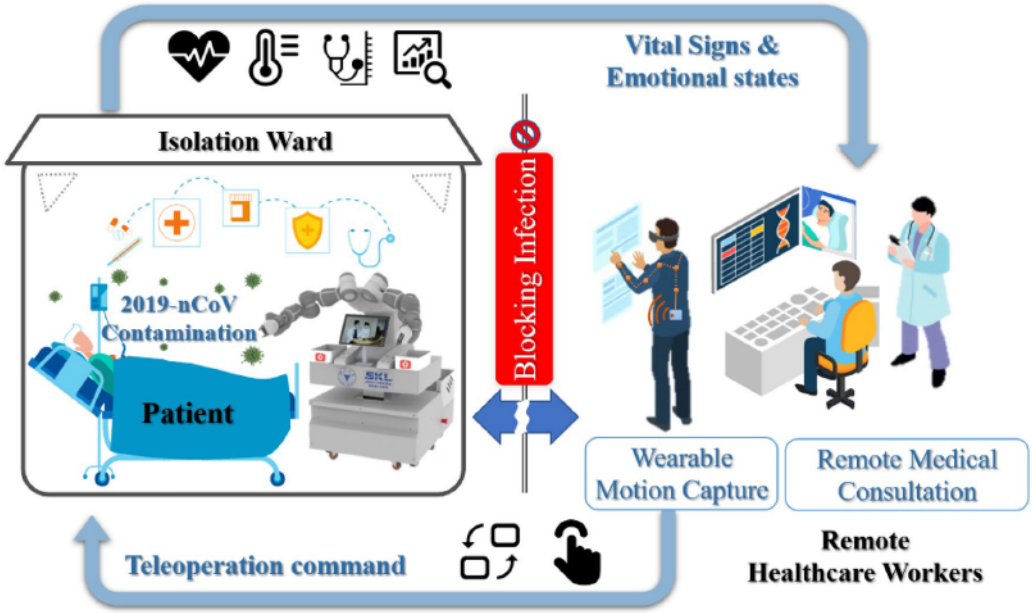


Figure 1.8: Telerobotic system for assisting patients affected by COVID-19 [5].

China, in December 2019. Rapidly, COVID-19 pandemic had spread within China first and all over the world involving different countries: China, Italy, France, U.K., Germany, United States, Brazil, and so on.
The reproduction number of infection R_0, defined as the average number of cases that an infected person will cause, is used to evaluate the spread of COVID-19. An estimated R_0 between 1.5 and 3.5 for the novel coronavirus shows an highly transmissibility resulting a global health crisis.
The death rate is high among older people and patients with diabetes, cancer, cardiovascular and chronic respiratory disease.
Social *distancing* has been identified as the best solution to avoid viral infection. For that reason, the scientific and technological world is fielding medical robots and telemedicine systems in order to control the spread of the infection to a large population. In healthcare, intelligent robotics and autonomous systems can play a key role. It is possible to identify different areas of robots application in healthcare: reception, nursing, ambulance, telemedicine, hospital service, cleaning, spraying/sanitization, surgery, radiology, rehabilitation, outdoor delivery and general assistance [17].
Robotic systems with a user interface for their remote control by a human operator, named Telerobots, offer the most obvious benefit in terms of assisting the healthcare system during the COVID-19 pandemic. In particular they can assist the medical staff for disinfection purposes, drug and food delivery to the warded patients, swab test and health screening, and also as communication medium between physicians and patients.
Some interesting teleoperation systems have been developed during this pandemic period. A typical system for this application (as the one presented in [5]) is composed of two main subsystems: the teleoperation system and the telepresence system. The first subsystem consists of a wearable motion capture device, a dual-arm collaborative robot and a pair of data gloves to capture the finger motions. The second one consists in a tablet computer. In this way, the teleoperated robot becomes the healthcare worker's eyes, ears, and body

in the isolation ward. This newly designed telerobotic system combines the strengths of healthcare workers (expert knowledge for patient care) with the strengths of robotics (social distancing and capabilities to work in hazardous environments).

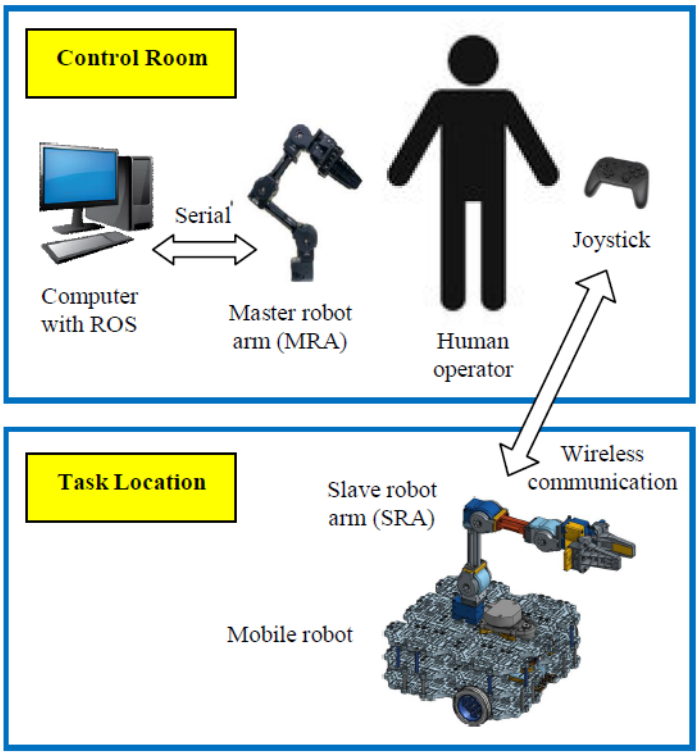


Figure 1.9: Dual-arm robot system with a mobile robot platform [6].

Another interesting teleoperation system has been developed in order to prevent infection of medical doctors and nurses during health screening and swab testing during COVID-19 outbreak [6]. The system consists of two sets of robot arms, namely the Master Robot Arm (MRA) and the Slave Robot Arm (SRA), and a mobile robot platform. The robot arm on the mobile robot platform mimics any movement of the MRA that the human operator moves in the control room.
These examples highlight the potential of teleoperation systems during this difficult period. A chance to prevent close contact between people during COVID-19 pandemic is indeed to use the teleoperation systems.

1.2 Shared Autonomy in teleoperation

Different factors make robotic teleoperation a complex task: the high degree-of-freedom manipulators, operator inexperience, and limited operator situational awareness. In order to reduce complexity of teleoperation, researchers introduced the shared autonomy control paradigm.
There are two different views of shared autonomy [7]:

- Single-Dimensional view

- Multi-Dimensional view

According to the *single-dimensional* view (also known as the classical view), there are different degrees of autonomy between direct control and fully autonomous systems.
Shared autonomy is intended as a general term that is used to mean the whole spectrum of degrees of autonomy. But, as shown in figure 1.10, there are multiple expressions that names the semi autonomous control, as supervisory control, shared control or collaborative control.

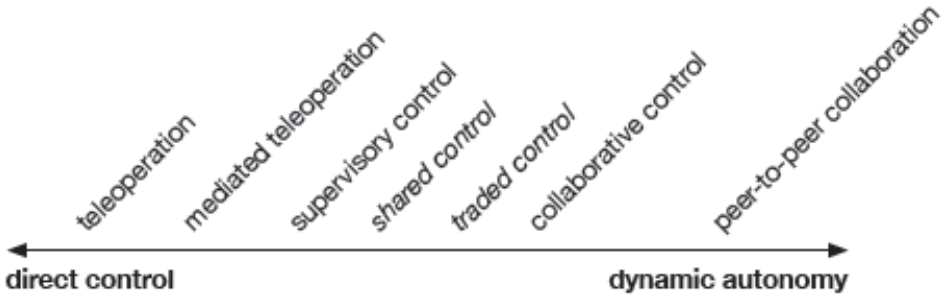


Figure 1.10: Different degree of autonomy as presented in [7]

The autonomous system is seen as an intelligent tool that, thanks to his autonomy, can release the human operator from many details of the action. In shared autonomy there is a semi autonomous system and such a system is teleoperated, but it is carrying out some actions autonomously. The artificial system deals with the low-level details of the task execution, while the user deals with the high-level one: there is a human-centred perspective.
When autonomous agents engage in interaction, many different phenomena appear. The *multi-dimensional* view of shared autonomy arise from the evaluation of these phenomena. According to this vision, interaction between autonomous systems takes into account three different levels of autonomy: freedom of intentions, freedom of decision, and freedom of acting. These levels stratify the shared autonomy space. For this reason, the shared space by participating agents must consider these autonomy levels.
Two extreme examples are now presented to clarify these concepts (cfr figure 1.11): the case of a fully teleoperated system and the case of collaborative control.
In the first example, there is no autonomy on the system side and even the actions are selected by the human. The system is under direct control. In the collaborative control, human and system work in the same space and at the same time. The system is guided by the user intention, but some autonomy is implemented on the lower level in order to free enough the human operator. Additionally, communication between the robot and the human is required to coordinate the behavior. This perspective is devoid of any hierarchy between the agents (e.g. robot vs. human).
In the shared autonomy context, an autonomous system is limited by a set of constraints given by the current situational context and by the other agents sharing the situation. The possibilities for each system depend on how the agents shape their behavior. As a consequence, in shared autonomy the autonomy must be seen in the context of the whole situation and of the other interacting systems.

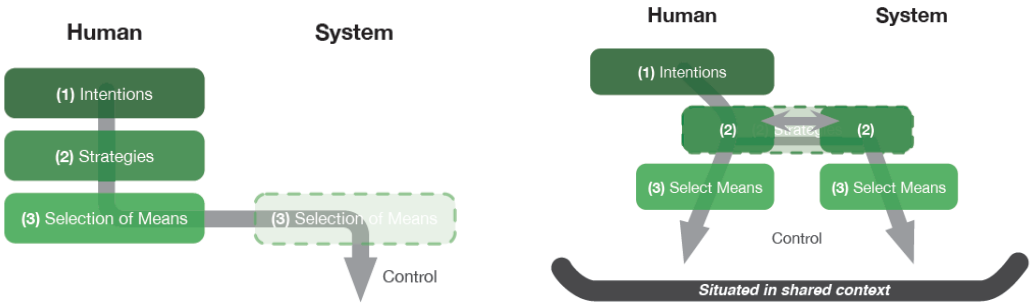


Figure 1.11: On the left side the fully teleoperated system condition; on the right side the collaborative control situation.

There are two main reasons for shared autonomy in teleoperation: perceive collaboration and get good performance in the task [18]. Human psychological studies have shown that the relationship between desired autonomy and task performance is mediated by motivation, perceived utility, individual self-efficacy, information asymmetry, cognitive distraction, tasks interdependence, and task formalization and variability.

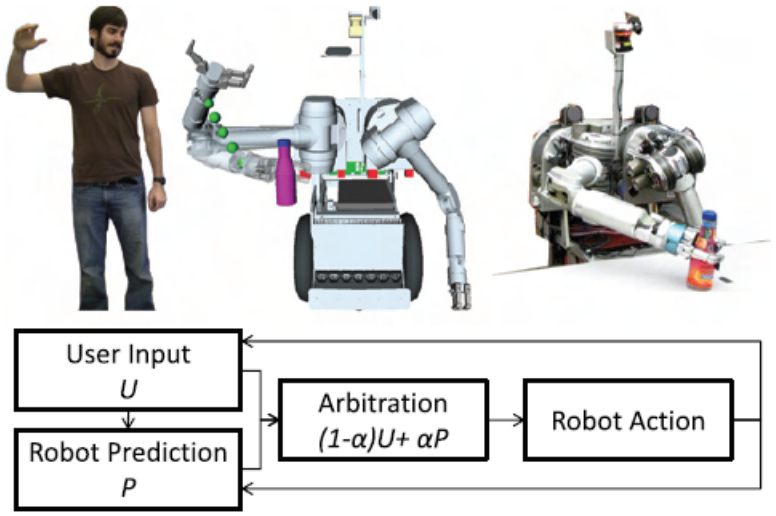


Figure 1.12: Assistiv teleoperation as presented by [8]. Firstly, the user provides an input U. The robot predicts their intent, and assists them in achieving the task. Then, policy blending arbitrates user input and robot prediction of user intent.

In assistive teleoperation, the robot helps the user in accomplishing the desired task, making teleoperation seamless [8]. In this context, the robot carrying out two fundamental tasks to help the user:

1. prediction of teleoperator intention
2. arbitration with teleoperator's input

As shown in figure 1.12, at any instant, given the input, U, and the prediction, P, the robot combines them using a state-dependent arbitration function $\alpha \in [0, 1]$. Policy blending

with accurate prediction has a strong corrective effect on the user input. Obviously, the robot has to predict accurately and arbitrate appropriately to have good results.
In literature there are few examples of teleoperated hand-arm robotic platforms that have been developed in order to evaluate shared autonomy algorithms. It was demonstrated that the autonomy onset strategy that leads to perceive collaboration interaction changes with the characteristics of the task.

1.3 Open questions and motivation behind the development of a teleoperated platform

Does the autonomy onset strategy that leads to a collaborative and/or efficient interaction depends on the automatability level of the task?
The research team of the Human Robot Interaction laboratory of the BioRobotics Institute will be conducted to find an answer to this question and the aim of the present thesis is to develop the teleoperated system that will be used for this aim (cfr section 1.4).

1.3.1 Experiment

The experiment that will be carried on consists of a boxing-up task performed in teleoperation mode by naïve users. The robot will operate on sensorized platform. Two cubes, one box and one cover will be used during the experiment. The cubes, the box and the cover could have a plain colour or a texture. Each cube is positioned on a sensor that permits the identification of the cubes presence. The work area is the area in which the boxing up task is performed.

Boxing-up task

The actions to follow to box-up the objects are here described (the order of the actions is fixed).

Action 1. Move the box to the work area.

Action 2. Take the cube A and box it up.

Action 3. Take the cube B and box it up.

Action 4. Move the cover to the work area and close the box.

Experimental conditions

The experimental conditions are five in total, obtained from the combination of two strategies for the autonomy onset with two level of awareness, plus the control condition of full teleoperation. The experimental conditions are here described in detail.

Strategies for autonomy onset. The experiment compares among the following autonomy onset strategies. *Autonomy (AU)* consists in the robot automatically performing all the operations it is able to do, and then wait for the operator command.

Human-commanded (HC) requires the operator to command the robot the operations he wants the robot to perform (operator is aware of what robot can/cannot do). *Full teleoperation (FT)* is the control condition and consists in performing the telemanipulation task without any assistance.

Awareness of the boxing up process. The awareness of the boxing up process has two levels, in which the robot camera can recognize only plain colour objects.

Level I: The cube B has a plain colour. The robot can perform only the third action.

Level II: The box and the cube B have a plain colour. The robot can perform the first and the third actions.

Outcome measures

As objective measurement of the performance of the task execution, time for task completion and the error number (involuntary object drops, or in a wrong place).
As measures of the experience of the user, we administered a modified version of the NASA TLX test: the test is conceived for measuring the workload of the persons, extended with some questions for testing the experience

- Mental Demand
- Physical Demand
- Temporal Demand
- Performance
- Effort
- Frustration

Extended:

- Comfort
- Confusion
- Naturalness
- Fluency
- Easiness

Figure 1.13: The teleopered system developed during my thesis in action during a pick a place task.

1.4 Overview of the teleoperation system

The thesis project is focused on the development of a teleoperated hand-arm robotic platform as shown in figure 1.14. The teleoperated system has been developed for the evaluation of shared autonomy algorithms. In order to accomplish this assessment, the system must make possible to carry out a pick and place task. In the execution of the task, the teleoperated system to be implemented must be first of all intuitive to operate and then have good performance in terms of pose accuracy and time delay.
An overall description of the system architecture is presented.
Different devices were considered during the hardware selection phase. After some preliminary devices evaluations, a 6-axis robotic arm (UR5), an anthropomorphic robotic hand (Mia), an IMU motion capture System (Perception Neuron 32 V2), and a motion capture data glove (CyberGlove) have been chosen.
Therefore, the proposed telerobotic system can be divided into a human-motion-capture subsystem and a robot-control subsystem.
The collaborative robot is the receiver of human instruction and the output of the action. The space in which the manipulator task is specified is the operational space. Due to the type of task that has to be performed, it is preferred to work in the operational space to give priority to the control of obstacles to the end effector rather than to the joints. Therefore, the phase of reaching is given maximum importance.
The human-motion-capture subsystem is composed of the wearable IMU motion capture System and the motion capture data glove. The operator's upper limb motion data acquired from the IMUs permit the replication of the human movement by the robot.
Besides, a motion capture data glove is used for the acquisition of the flexion/extension of the fingers, which can reflect the operator's grab or release intentions. The signals are used to control the opening or closing state of the robot's hand.
The suit is equipped with a motion capture glove. Because of his poor accuracy, a fundamental parameter to consider to accomplish the pick-place task in the best way, the CyberGlove motion capture system has been preferred (more details are reported within

the paragraph 3.1.2). The developed teleoperated hand-arm robotic system is a multi-node distributed control system based on ROS. The study of the kinematics of the human and robotic arm and hand was necessary in order to implement the control of the system. Knowledge of the ROS framework and the C++ programming language was learned for writing the codes that allowed the communication between the various hardware components. Furthermore, the assembly of the teleoperated system required good knowledge of communication protocols (specific for the different devices used).

1.5 Thesis management

The master's thesis work began in May, period during which the COVID-19 pandemic spread all over the world.
The strictly practical nature of the construction of a teleoperation setup has led to the need to postpone what was the initial program over time, due to the closure of the labs from March to May, the completion of safety procedures in June and the limited access to the lab from July to October due to the possible capacity of the rooms.
Nonetheless, I started remote activities from the very beginning. I had the chance to take the hardware home so I could conduct the first experiments. The work was planned as reported in the following Gantt.

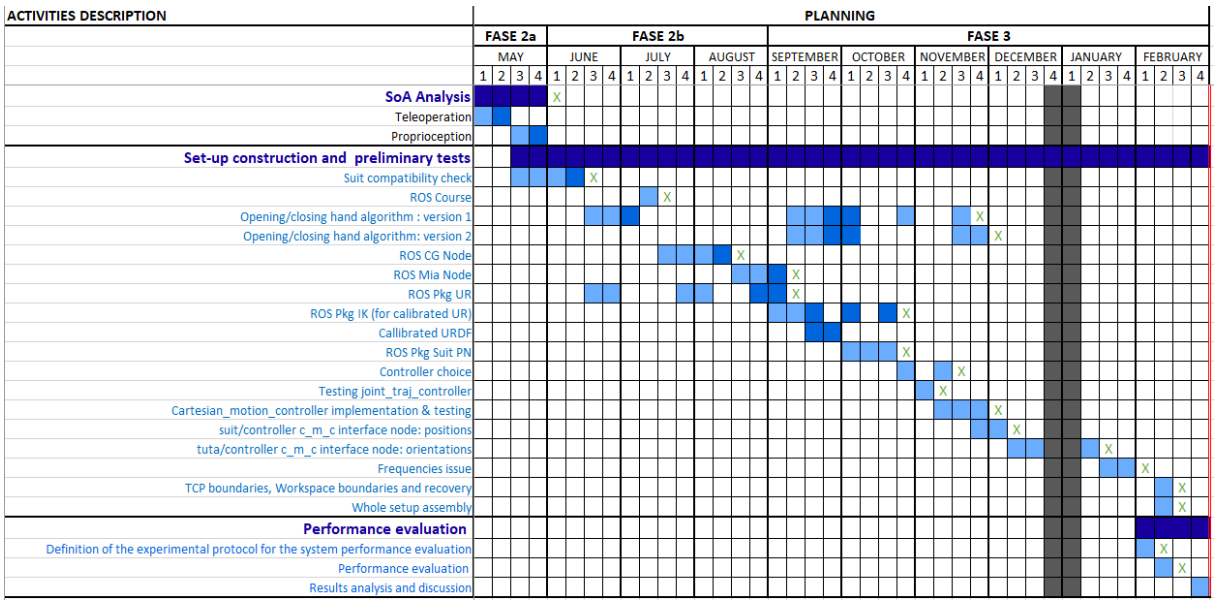


Figure 1.14: The activities planning.

The activities started with a preliminary State of Art analysis. In particular, I investigated on the different teleoperation systems found in literature in order to highlight pro and cons of each system. In particular, I examined the relationship between accuracy and precision of the task to be performed and the type of teleoperation setup. In conjunction with the State of Art analysis, the performances of the Perception Neuron motion capture system were evaluated in order to check the compatibility of the inertial system for our specific application.
The software architecture was implemented in the ROS environment.For this reason, part

of the time was dedicated to learning ROS. From the middle of June to the beginning of September, I have dedicated myself to the development of the decoding algorithm at the base of teleoperation control of the robotic hand. Then, I started to work to the teleoperation control of the robotic arm. This macro task was divided in different sub task, as shown in the Gantt.
At the end of the of the set-up definition, the whole teleoperation setup has been assembled. The last part of my work was focused on the performance evaluation of the developed teleoperation hand-arm platform.

Chapter 2

Kinematic analysis

Manipulative actions required to accomplish a task by a robot are difficult to be predicted; moreover, the environment is often unstructured. Thus, a human operator is needed to control the robot during the execution of tasks. To avoid or at least reduce the physiological and psychological fatigue of the operator, the best condition consists of the same kinematic structure between master and slave that permits to obtain an intuitive and convenient slave manipulator control. However, the master-slave telemanipulation robot systems are usually heterogeneous and application spacific.
The condition that comes closest to the ideal one considers the dominant human hand/arm as a master arm because the most intuitive and habitual way to operate for a person who is using his hand/arm directly. For this reason, the control of the slave arm is obtained via a wearable motion capture system, which can get the pose data of the links of the master arm simultaneously without reducing the range of motion. Moreover, the control of the slave hand is achieved through a wearable motion capture data glove that captures the degrees of freedom of the human hand.
The master and slave systems have different kinematic structures. In the following sections, a brief description of the biomechanical and robotic models involved in our system is presented and the Denavit-Hartenberg convention is used for the kinematic analysis.
In this chapter, I reported the kinematic analysis of the human body parts and of the devices used, while the technical description and the software developed are reported in chapters 3 and 4.

2.1 Human arm kinematic model

The human arm is composed of three rigid links: the upper arm, the forearm and the hand. These three rigid segments are linked thanks to joints with a total of 7 degrees of freedom (DoFs) [19].
The shoulder joint has three DoFs that can be assigned to flexion-extension, abduction-adduction and external-internal rotation of the humerus relative to the scapula. The elbow joint has two DoFs, corresponding to flexion-extension and pronation-supination. Finally, the two DoFs in the wrist coincide with flexion-extension and abduction-adduction.
The shoulder (glenohumeral) joint is usually considered as a ball-and-socket joint and the elbow and wrist joints were modelled as skew-oblique joints.
All these joints are bound to cover a limited range of motion for each degree of freedom

and this range is dictated by the human anatomy.
The Denavit-Hartenberg convention is used to construct the mathematical model.

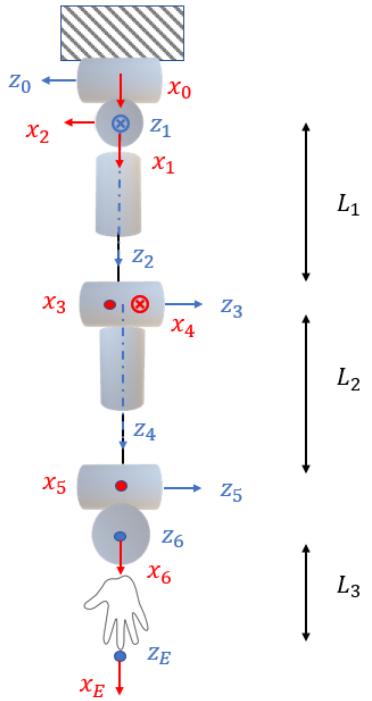


Figure 2.1: Kinematic configuration of the human arm.
L_1, L_2, L_3 are the links lengths.

	θ	d	a	α
Joint 1	θ_1	0	0	$\pi/2$
Joint 2	θ_2	0	0	$\pi/2$
Joint 3	θ_3	L_1	0	$\pi/2$
Joint 4	θ_4	0	0	$\pi/2$
Joint 5	θ_5	L_2	0	$\pi/2$
Joint 6	θ_6	0	0	$\pi/2$
Joint 7	θ_7	0	L_3	0

Table 2.1: DH parameters of the human arm.
θ_i values are the variables.

2.2 Human hand kinematic model

The human hand [20] kinematic model is composed of 19 links that imitate the corresponding human bones. The total number of degree of freedom is equal to 24 and represents the human hand joints.

The thumb and the rest of the fingers are considered with two different kinematic configurations. The index, middle, ring and little fingers are defined by 4 links and 5 DoFs. In particular, the metacarpophalangeal joint (MCP) is modelled by a 2 DoFs joint whereas carpometacarpal (CMC), proximal interphalangeal (PIP) and distal interphalangeal (DIP) have 1 DoF. The 2 DoFs of the MCP joint can be assigned to flexion-extension and abduction-adduction; the only one DoF of the CMC and PIP joints is referred to flexion-extension. The thumb finger is modelled by 3 links and 4 DoFs, instead. The metacarpophalangeal (MCP) and interphalangeal (IP) joints are modelled by 1 DoF (flexion-extension), whereas the trapeziometacarpal (TMC) thumb joint is defined by a 2 DoFs joint (flexion-extension and abduction-adduction).
Also in this case the joints are bound to cover a limited range of motion and, as in the previous paragraph, the Denavit-Hartenberg notation is used to construct the mathematical model.

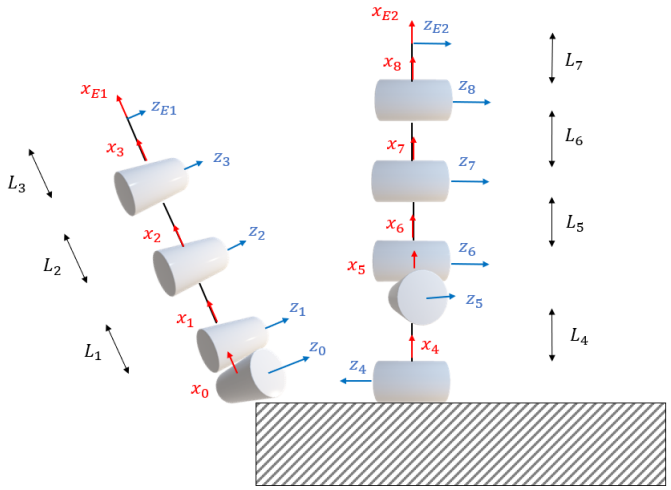


Figure 2.2: Kinematic configuration of the human hand. It is reported the thumb and index kinematic configurations. The middle, ring and little fingers replicate the index configuration. L_i values are the links lengths.

	θ	d	a	α
Joint 1	θ_1	0	0	$\pi/2$
Joint 2	θ_2	0	L_1	0
Joint 3	θ_3	0	L_2	0
Joint 4	θ_4	0	L_3	0

	θ	d	a	α
Joint 5	θ_5	0	L_4	$\pi/2$
Joint 6	θ_6	0	0	$\pi/2$
Joint 7	θ_7	0	L_5	0
Joint 8	θ_8	0	L_6	0
Joint 9	θ_9	0	L_7	0

Table 2.2: The first table reports the DH parameters of the thumb; the second table reports the DH parameters of the index, middle, ring and little fingers. θ_i values are the variables.

2.3 Robotic arm kinematic model

The teleoperated robotic manipulator is the UR5, a 6-degrees-of-freedom (DoFs) robotic arm. Here, the kinematic model of the robot is described; technical specifications of the device can be found in chapter 3 instead.The kinematic model of the UR5 is composed of 5 links and 6 joints [21].

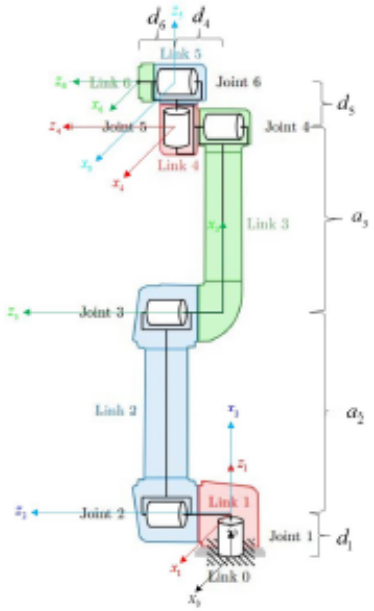


Figure 2.3: The UR5 Denavit–Hartenberg configuration [9].

	θ	d	a	α
Joint 1	θ_1	0.089159	0	$\pi/2$
Joint 2	θ_2	0	-0.425	0
Joint 3	θ_3	0	-0.39225	0
Joint 4	θ_4	0.10915	0	$\pi/2$
Joint 5	θ_5	0.09465	0	$-\pi/2$
Joint 6	θ_6	0.0823	0	0

Table 2.3: DH parameters of the UR5 robot. θ_i values are the variables.

2.4 Robotic hand kinematic model

The anthropomorphic robotic hand is Mia hand, manufactured by Prensilia SRL. Here, the kinematic model of the robot is described; technical specifications of the device can be found in chapter 3 instead.
Similar to the kinematic human hand model, the thumb and the rest of Mia fingers are considered with two different kinematic configurations. Each finger is composed of only one link. Index, middle, ring and little finger have 1 DoF each for flexion-extension, but middle, ring and little movements are coupled, so that 3 DoFs reduces to a single one The thumb has 2 DoFs: the flexion-extension and the abduction-adduction.

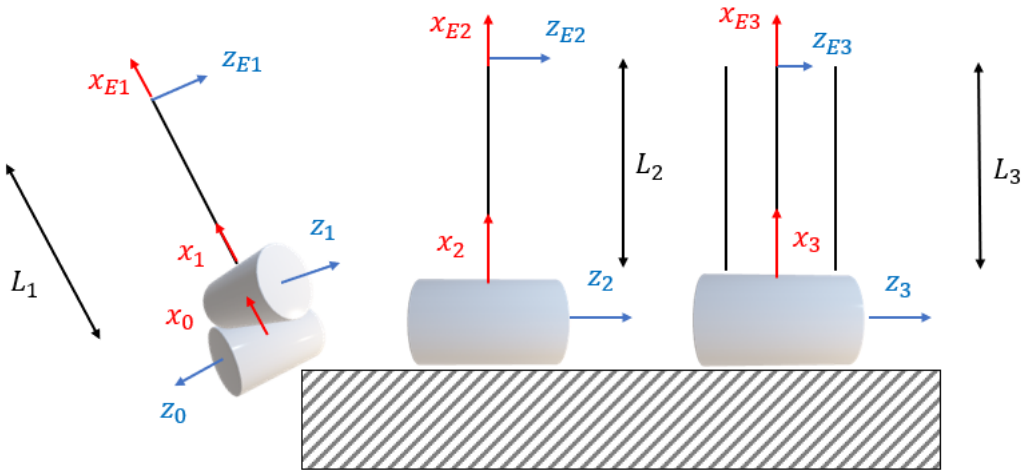


Figure 2.4: Kinematic configuration of the Mia robotic hand. L_1, L_2, L_3 are thumb, index and middle/ring/little fingers lengths.

	θ	d	a	α
Joint 1	θ_1	0	0	$\pi/2$
Joint 2	θ_2	0	L_1	$\pi/2$
Joint 3	θ_3	0	L_2	0
Joint 4	θ_4	0	L_3	0

Table 2.4: DH parameters of Mia hand.
θ_i values are the variables.

Chapter 3

Hardware architecture

3.1 Teleoperation system

The teleoperation system is composed of a wearable inertial motion capture system (Perception Neuron, model PN 32 V2) and a dataglove (CyberGlove, model CyberGlove II) which allow for measuring the motion of the operator's arm and hand, respectively.

3.1.1 Perception Neuron Mo-Cap system

The human motion capture (Mo-Cap) analysis procedure requires qualitative or quantitative measurements. The quantitative analysis can involve different motion capture systems in order to measure the biomechanical variables. [22]
Motion capture systems are divided into optical and non-optical systems. Optical systems are divided into marker-based and non-marker based; non-optical systems are divided into magnetic, electromechanical and inertial.

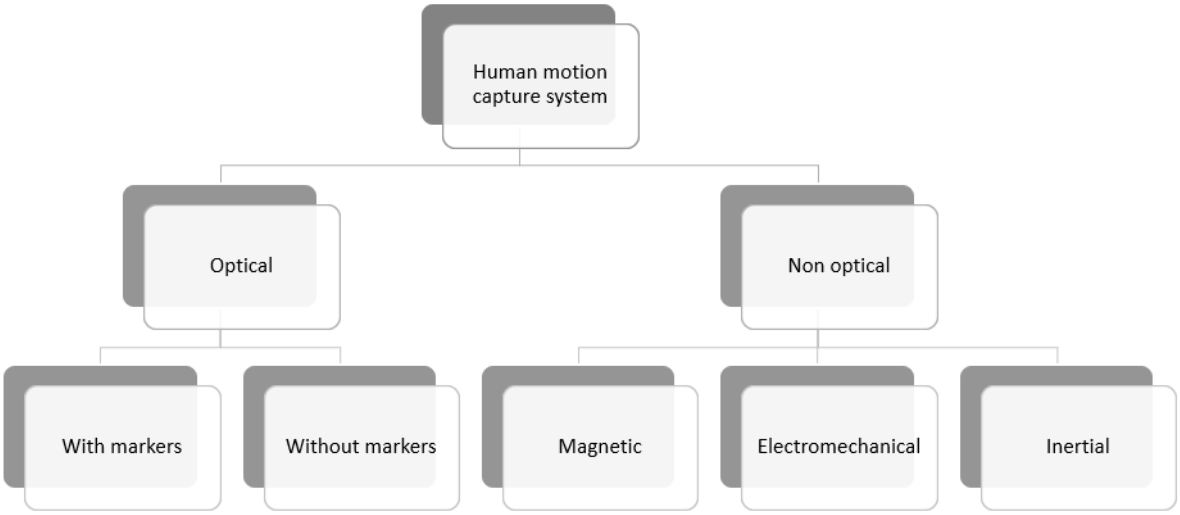


Figure 3.1: Motion capture systems overview

Optoelectronic motion capture systems represent the best solution in the measurement and quantification of human kinematics, because of their high accuracy, high precision, and high resolution. Retroreflective markers attached to the body and the cameras allow for the acquisition of position data, which are useful for the biomechanical analysis, in both static and dynamic conditions. [22] An example of an optoelectronic motion capture system is the Vicon system. However, the optoelectronic system has also disadvantages: an high cost and lengthy set-up times. The system requires a laboratory equipped with high-resolution infrared cameras, as well as a highly trained operator. Besides, optoelectronic systems are confined to the volume of space where the equipment is installed.
The improvement in the usability of IMUs (Inertial Measurement Units) introduces such devices as an alternative to the optical measurement system for motion analysis: they preserve or improve the feature of wearability, still keeping a good compromise in terms of accuracy.
The motion capture system used in our teleoperation setup is the *Perception Neuron (model PN 32 V2)*. The Perception Neuron system is a modular motion capture system. The modular system is based on the IMUs (Inertial Measurement Units), named *neuron*, which consists of a 3-axis accelerometer, a 3-axis magnetometer, and a 3-axis gyroscope. The IMU sensors could be either one (1-axis), two (2-axis) or three-axis (3-axis) sensors. However, the three-dimensional (3D) motion analysis needs 3-axis sensors in order to accurately detect movement in each direction. [22] The neurons have a size of a one square centimetre that allows to easily place them via Velcro strapping on the anatomical landmarks.

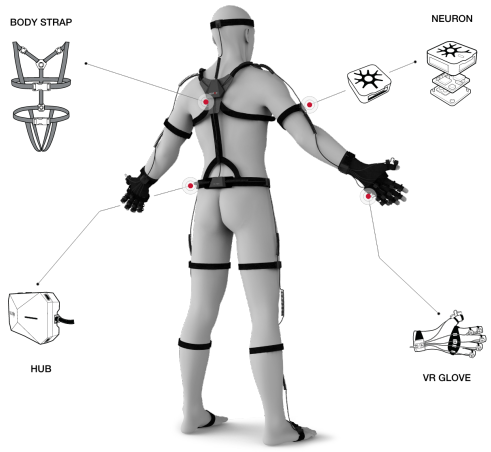


Figure 3.2: Perception Neuron motion capture system and its components.

The Perception Neuron motion capture system includes also a motion capture data glove. The glove permits the recording of the bending signals of the fingers, which can reflect the operator's grab or release intentions.
The number of sensors determines the working frequency of the system: it works at 120 Hz with 17 Neurons or less, and at 60 Hz with 18-32 Neurons.
The suit hub allows the connection and powering of all neurons and it is worn on the human hip. The hub combines individual sensor data and transmits them via USB or Wi-Fi to the Axis Neuron software, an application of the Perception Neuron running in Windows operating system (OS). The device offers also the possibility of onboard recording of motion data using the built-in micro-SD slot.
The use of the Perception Neuron motion capture system shows pros and cons, that are

listed below.

Pros:

- The system is small, composed of inertial trackers with a size of 12.5mm x 13.1mm x 4.3mm each, connected to a neuron hub (59mm x 41mm x 23mm). The whole system weighs less than 300 grams (excluding battery).
- The Perception Neuron suit works in three different configurations: full body (takes data from all the sensors available), upper body (takes data from the sensors of the upper body) and single arm (takes data from the sensors of one arm). This gives the system the characteristic of adaptability to the specific application.
- Perception Neuron was developed as a professional tool and it is intuitive and simple enough to be operated by novice users. It is a versatile system that is used for many applications, including analyzing movements for video game developers, filmmakers, visual effects professionals, biomechanics researchers, sports and medical analysts.
- The system is affordable.

Cons:

- Not precise absolute positioning.
- IMUs are extremely magnetically sensitive. Magnetic interference can be caused by fridges, large motors, computers and other large electric tools. If sensors get magnetized, each one of them needs a process of re-initialization.
- Re-calibration is needed after a motion of a certain duration, to zero-out rotational errors on sensors.

For our application, the disadvantages are negligible compared to the advantages.
The Perception Neuron motion capture system is transparent to the operator during the teleoperation thanks to the small load and encumbrance.
The Mo-Cap system is used in the upper body configuration and the total number of sensors we chose to use is equal to nine. Thanks to this choice, the system works with a frequency of 120 Hz.

3.1.2 Cyberglove motion capture data glove

The human hand has roughly 24 degrees of freedom (DoFs). Different methods can be used to measure hand movements, but most of them fail while performing functional ADL (Activities of Daily Living) because of the high number of DoFs.
Electromagnetic systems are affected by magnetic and electrical interference from metallic objects in the environment. High accuracy is ensured by marker-based optical systems. However, some problems also occur with the use of this devices: they can be used only within the area covered by the cameras; a substantial amount of time is necessary to set up the markers; markers often become occluded during the recording of tasks. Another

possibility is the use of markerless optical motion capture and inertial systems, but with accuracy problems. [10]
It is possible to deduce the pose of the fingers by exploiting the outputs returned by inertial sensors. However, since we are interested in making particularly accurate movements, which accuracy can be easily compromised by drift, we prefer the use of CyberGlove. Bending sensors technology allow increased accuracy and precision for movements of limited amount like finger's ones, that could be even more subjected to the effect of IMU sensors drift.

One of the most famous data glove systems is the *CyberGlove*.
The CyberGlove data glove uses resistive bend-sensing technology positioned on the dorsal side of the hand. The internal flexible structure measures the change in resistance to an electric current as the sensor is bent. The CyberGlove Interface Unit (CGIU) translates the voltage output of each sensor into an integer number in the range 0-255 using an 8-bit analog-to-digital converter (ADC). In this way, the resistive bend-sensing sensors transform hand and finger motions into real-time digital joint-angle data. The data in output from the CyberGlove are transmitted through the serial port to the PC to permit the successive elaboration. The maximum working frequency is 100 Hz.

Figure 3.3: CyberGlove II motion capture data glove, instrumented with 18 high-accuracy joint-angle measurements [10].

The CyberGlove II is equipped with a total of 18 bend sensors: two on each finger, four abduction sensors, plus sensors measuring thumb crossover, palm arch, wrist flexion, and wrist abduction.
As with any device, it is possible to identify the pros and cons of its use.

Pros:

- The system has a good fit. It is constructed with stretch fabric for comfort and a mesh palm for ventilation. It has open fingertips, which allow the user to easily type, write, and grasp objects.
- The CyberGlove permit the data recording from all finger joints continuously, without occluding problems, and with no special environmental constraints. [10]
- The CyberGlove permits the acquisition of repeatable data with a high resolution.
- The CyberGlove motion capture system has been used in a wide variety of real-world applications, including digital prototype evaluation, virtual reality biomechanics, and animation.

Cons:

- Each sensor does not have an exact one-to-one relationship with the anatomical angle to be measured, because of two reasons: readings from sensors related to a single joint but bending in different directions could be affected by movements towards directions other than the intended one; their placement is not accurate. This requires a subject-specific calibration procedure, often complex.
- The measurement can be influenced by the amount of space available between hand and glove.

The accurate identification of at least the closing and opening conditions of the hand is very important in several teleoperation tasks.
Therefore, we decided to use the CyberGlove, instead of the glove provided by the Perception Neuron system, in order to avoid the problems related the IMU sensors and take advantage of all the benefits provided by this device, in particular the possibility to acquire repeatable data with a high resolution.

3.2 Teleoperated system

The teleoperated system comprises a 6-axis industrial robot (Universal Robot, model UR5) and an anthropomorphic artificial hand (Prensilia SRL, model MIA). Here, technical specifications of the devices are reported; more details about the UR5 and Mia kinematics characteristics were presented in the Chapter 2.

3.2.1 The UR5 robot arm

Serial robots are widely used in manufacturing, handling material, and teleoperation.
The robotic manipulator selected for our teleoperation system is the UR5 collaborative robot produced by the Universal robots. The universal robots have developed a series of robotic manipulators that are now widely used by many universities and industries in repetitive and dexterous tasks, such as packing, assembly or testing. The collaborative robots can be introduced into existing production lines and thus easily replace humans in repetitive or non-ergonomic tasks. Collaborative robots are particularly suitable where cooperation between robot and humans is required and cooperation between robot and humans.
The UR5 is fast, easy to program, flexible, safe and offers low-level programming access to robot controller with a high cycle time [23]. The UR5 weight is equal to 20.6 kg and the diameter of the footprint is 149 mm. It covers a working range of 850 mm and it can support payloads of up to 5 kg. UR5 is conceived as a collaborative robot, thus its safety features can be easily set according to the foreseen application or human presence, or readings form external sensors.

3.2.2 The Mia hand

The end-effector of our teleoperation system is an electric anthropomorphic hand, Mia, that was conceived as a prosthesis but serves perfectly as collaborative gripper.

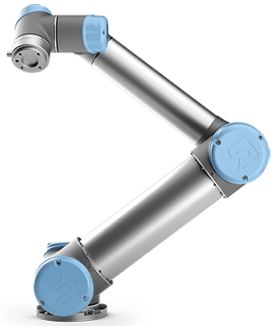


Figure 3.4: The 6-axis industrial robot (Universal Robot, model UR5)

The Mia hand can be involved in multiple scenarios, such as research, human-machine interfaces, collaborative robotics, and so on. The Mia hand is composed of three embedded motors that allow to interact with the environment and to grasp objects and toolswith three types of grasp: cylindrical, pinch and lateral, that are the types of grasp most frequently used by humans in everyday manipulative actions. The robotic hand Mia is made entirely of steel and has a Geneva mechanism that regulates the ab/adduction of the thumb by means of the same motor that regulates the flexion/extension of the index finger. The choice of having only three motors is due to the fact that it is required the realization of a gripper that is as similar as possible in size to the human hand.
It is a smart robotic system thanks to the embedded current, position and force sensors that allow for a variety of closed-loop control behaviour.

Thanks to a specific communication protocol, it is possible to send serial commands to Mia in order to control in position or in speed the three motors individually or together. In addition, specific serial commands allow to carry out different types of grasps aforementioned.
Mia hand has a weight of 500 g and it is robust and powerful, with up to 70 N maximum grip force available in all grasping patterns.

Figure 3.5: The anthropomorphic artificial hand (Prensilia SRL, model Mia)

Chapter 4

Software architecture

In this chapter, the software architecture that controls the devices within the teleoperation system is described. In the first section, a brief overview of ROS framework is presented. In the second section, the algorithm that reads data from the Cyberglove and controls the hand accordingly is presented. In the third section, the code that manages the control of the robotic arm following the motions captured with the Perception Neuron suit is described.

4.1 A brief introduction to ROS

The Robot Operating System (ROS) is a flexible framework for writing robot software. A framework in computer programming is an abstraction in which softwares providing generic functionalities can be selectively changed by additional user-written code, thus providing application-specific software. Although ROS is not an Operating System, it provides all the services that any other OS does, like hardware abstraction, low-level device control, implementation of commonly-used functionalities, message-passing between processes, and package management. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot software. The primary goal of ROS is to support code reuse in robotics research and development.
ROS has various distributions, named alphabetically. The ROS distributions are supported only on Linux-based operative systems with Ubuntu or Debian platform. Each ROS distribution is targeted to a specific Ubuntu release: the last two distributions are Melodic Morenia and Noetic Ninjemys, respectively targeted to Ubuntu 18.04 Bionics and Ubuntu 20.04 Focal. When this thesis work began, Noetic distro had just been released. Consequently, lot of packages had not yet been released for ROS Noetic. Therefore, we contributed to test melodic releases of ROS driver packages of our devices on the newly-released Noetic distro and we had the chance to suggest to the developers of the packages eventual fixes needed for package upgrade. The Robot Operating System mainly uses two languages: C++ and Python. we decided to use C++ language used in our application.

4.1.1 Basic concepts of ROS

In ROS, a process is called *node*: every node should be responsible for one task. These processes can be grouped into *packages*, which can be easily shared and distributed. Nodes communicate with each other using *messages*, a data structure, passing via logical channels called *topics*. The topic is a name that identifies the content of the message. Each node can send or get data from other nodes using the publish/subscribe model. In this way, there may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics.
Request/reply is done via *services*, which are defined by a pair of message structures: one for the request and one for the reply. A service represents an action that a node can take which will have a single result. As such, services are often used for actions which have a defined beginning and end.
It is also important to mention the functionality of the *parameter server* that allows data to be stored by key in a central location. It is part of the ROS Master, which provides naming and registration services to the rest of the nodes in the ROS system. Without the Master, nodes would not be able to find each other nor exchange messages.

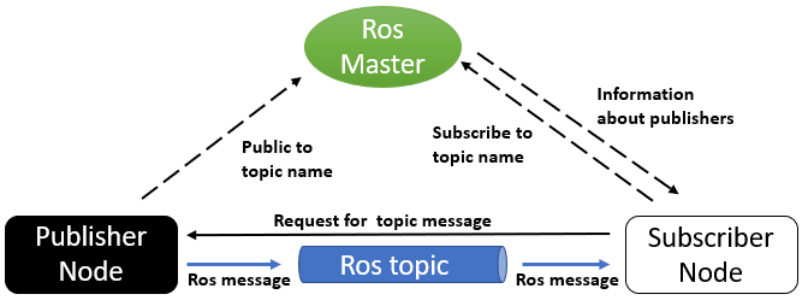


Figure 4.1: ROS Network Setup

4.2 Human hand - robotic hand teleoperation

In this section, the teleoperation strategy implemented to control the anthropomorphic robotic hand after the readings of the outputs of the sensors of the CyberGlove will be discussed.
The CyberGlove motion capture dataglove allows for the measurement of the motion of the operator's hand. We chose to use the output of the sensors of the dataglove to detect the closing and opening conditions of the human hand, in order to control the opening and closing state of the robotic hand. To reach this aim, a detection algorithm for the opening and closing condition was developed in MATLAB and C++ programming language, and its functioning was qualitatively evaluated across different subjects. Then, once a closing/opening action was detected by the algorithm, a command to close/open the robotic hand is sent.

4.2.1 Subject specific calibration of the detection algorithm

A first factor influencing the goodness of the measure performed by the CyberGlove is the human hand dimension which determines the pre-stress level of the sensors. For this reason, at the beginning of the calibration process, the hand size is measured thanks to a specific scale (Figure 4.2). If it did not overcome a threshold of 7.5 (both for ladies and men), the subject has to wear an additional glove under the Cyberglove, in order to minimize the backlash between his hand and the Cyberglove.

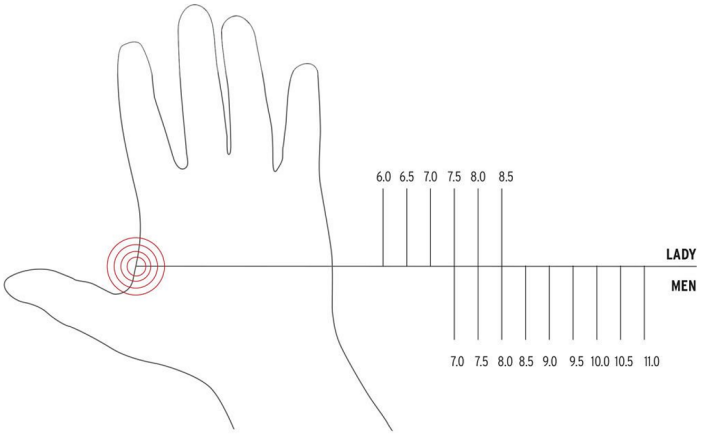


Figure 4.2: Scale used to measure the human hand size.

Moreover, an important requirement for the decoding algorithm is the independence of the detection of the opening and closing position from the wrist configuration, that could be flexed/extended and abducted/adducted. Therefore, we considered all the sensors, except the output of sensor 17 which measures the flexion/extension wrist movement and the output of sensor 18 which is responsible for measuring the abduction/adduction of the wrist (Figure 4.3).

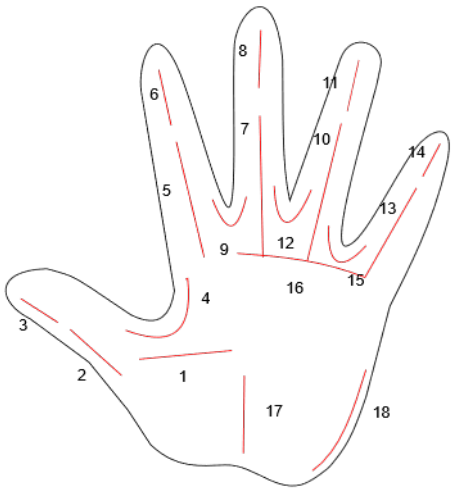


Figure 4.3: The figure shows the numbering and positioning of the sensors' dataglove.

The algorithm requires a preliminary and subject specific calibration procedure: the algorithm for the calibration was implemented in MATLAB, and it allows for the extraction of subject specific thresholds that are calculated in a standard way.
The calibration is executed in MATLAB and involves two different acquisitions: a first acquisition of outputs of the considered sensors in the opening hand condition; a second acquisition of outputs of the same sensors in the closing hand condition.

Figure 4.4: During the first acquisition is asked subjects to put their forearm on the table, with the hand's palm facing the table. During the second acquisition is asked the subjects to put their elbow on the table, take their forearm perpendicular to the table and close their hand like a plier.

Thanks to the calibration process, it is possible to calculate the mean over the time of the sensors' output during the closing condition. In the calibration process is also calculated the Euclidean distance between the sensors' output recorded during the closing and opening conditions, the two extreme condition. The calculated Euclidean distance permits to evaluate the range of sensors over the movement of interest. We define a threshold of 50% over the sensors' range.

4.2.2 Reading data from the CyberGlove: *cyberglove* ROS package

The communication with the CyberGlove system in ROS exploits the *cyberglove* package, that works as a driver for the device within ROS Fuerte distribution. The driver has been tested for compatibility with ROS Noetic release, the most recent ROS distribution. Minor changes to the original code were needed to ensure this compatibility.
The driver exploits the *cereal_port* package to read the serial port message, and decodes this message to obtain the raw output of the bend sensors. Proper parameters were set

to achieve data streaming at 100 Hz frequency. Finally, the driver streams the decoded data readings to the *cyberglove/raw/joint_states topic*.

4.2.3 Detection algorithm for opening-closing position of the human hand: *cyberglove_subscriber* ROS node

The decoding algorithm is implemented in the cyberglove_subscriber node. A subscriber allows the access to the CyberGlove data published on the cyberglove/raw/joint_states topic. It performs the calculation of the Euclidean distance between the actual sensors' output and the one recorded during the closing condition:

$$d(p,q) = \sqrt{(p-q)^2} \tag{4.1}$$

p and q are vectors filled with the sixteen sensors' output. In particular, p represents the vector acquired during the closing condition in the calibration process and q represents the vector of the live sensors' output.
If the calculated Euclidean distance overcome the subject specific threshold defined in the calibration process, the opening condition is recognized; conversely, in the opposite case.

4.2.4 The first version of Mia Hand ROS driver: *mia_subscriber* ROS node

The detection algorithm output is a boolean condition: if the algorithm recognizes the opening condition, the "false" boolean message is published on the */mia* topic; controversy in the opposite case.
The Mia node reads the message published on */mia* topic and, consequently, send the cylindric closing serial command to the robotic hand.

Figure 4.5: ROS architecture: the robotic hand teleoperation strategy.

4.3 Human arm - Robotic arm teleoperation

Human arm - Robotic arm teleoperation system exploits the pose of the palm, retrieved by the measurements of the inertial motion capture system, to control the Tool Center Point (TCP) of the robotic arm accordingly.
In order to make possible the teleoperation of the 6-axis robotic arm from the human

arm movements, different packeges work together in the ROS environment: here follows a detailed list of the softwares and packages developed or used to build the whole arm teleoperation architecture.

4.3.1 The Axis Neuron software

The hub is the central processing unit of the Perception Neuron motion capture system; it collects IMU sensors data and transfers them to the Axis Neuron software, a supporting application of the Perception Neuron suit in the Windows Operating System (OS).
Basically, the software receives the motion data from the hub, to visualize, process and export them. However, the software can also performs other operations.
In order to minimize magnetic interference, Axis Neuron software allows options like the calibration and the "zero button". The “zero button” functionality allows to reset the x and y coordinates to x = 0 m and y = 0 m when it is pressed. The reference z coordinate instead can be set through the software in single arm or upper body modes of the suit, and it will be set at the desired height; in full body mode, it is computed by the software according to leg sensor data and body links length (Figure 4.6).

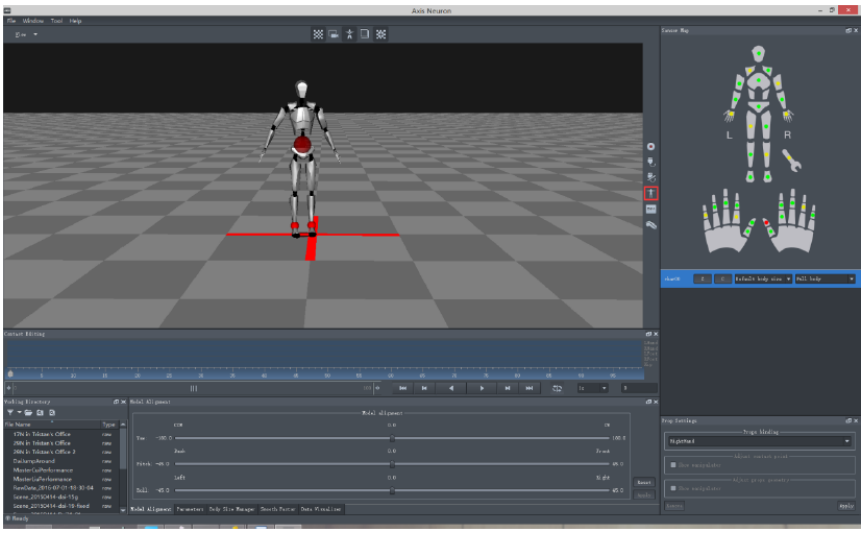


Figure 4.6: The figure shows the Axis Neuron software interface. The red box highlights the calibration button. Calibration is needed to correctly orient the sensors according to how they are worn.

The software reproduces human motion on a virtual animation model, thanks to the motion data received from the IMU sensors and a preset of anatomical constraints. In order to adapt to different body sizes of wearers, the software permits the insertion of the measured parameters of the body size, such as arm length, shoulder width and the height of the wearer. These parameters are set to create a subject specific virtual animation model and to increase accuracy in pose reconstruction. The skeleton model consists of 59 joints, which are ordered by the Axis Neuron software as reported in Figure 4.7.
After having processed the output of the IMUs sensors, the Axis Neuron software returns the motion data in two different formats: calculation and BVH data format.
The calculation data includes information about the position, velocity, quaternion, acceleration and gyroscopic data of each human bone. The BVH data includes the three

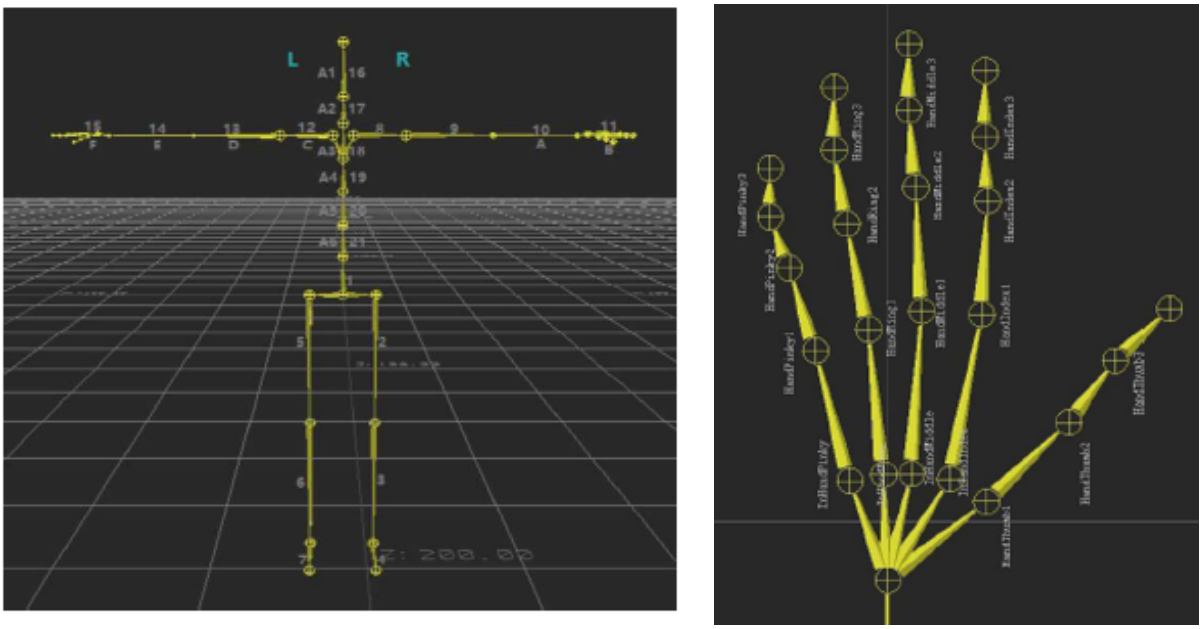


Figure 4.7: Skeleton graphs.

displacements and the three rotations data of each bone. We consider the BVH data format for our application.
In order to control the 6-axis robotic arm movement, we are interested only in the acquisition of the human right arm movements. For this reason, the Perception Neuron system is used in the upper body configuration, that appeared to be more reliable from a first qualitative evaluation. Based on the operator's upper limb motion data acquired from the IMUs, the human skeleton model is constructed: the considered kinematic chain includes references from hip, spine 0, spine 1, spine 2, spine 3, shoulder, upper arm, forearm, right hand. In the Axis Neuron software, the hip of the operator is set as the root node of the body's skeleton model, while the right hand coincides with the end node. There is a parent-child relationship between two adjacent nodes, and the node far away from the hip is considered to be the child node.

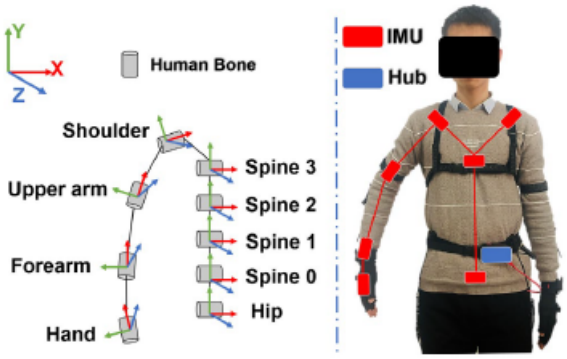


Figure 4.8: The skeleton model and the placement of IMUs on the human body. The number of bones does not correspond to the number of IMUs. [4]

4.3.2 Data streaming from Axis Neuron software to ROS: *PerceptionNeuronROSserial* application in Windows

The Axis Neuron software works on Windows OS, but the environment used to control the teleoperated system is ROS (Robot Operating System), a flexible framework for writing robot software, that runs in Linux OS. For this reason, an executable program called PerceptionNeuronROSserial is used to extract movement data from Axis Neuron and transfer them to ROS framework.
The Noitom company provides its SDK to parse the movement data streamed by the Axis Neuron software [24]. A Software Development Kit (SDK) is a set of tools provided by the manufacturer of a hardware platform, operating system, or programming language. SDKs permit developers to build specific applications for that platform, system or programming language.
The Axis Neuron software runs an internal server to send every motion frame to a virtual IP address; PerceptionNeuronROSserial application connects to this address to read the movement data. The PerceptionNeuronROSserial uses the NeuronDataReaderSDK to parse those data, and it outputs them in turn to a Linux OS PC where ROS is running, via a physical TCP/IP socket.
The TCP protocol enables two-way communication and ensures successful delivery of non corrupted data. In contrast, the UDP protocol is a one-way communication protocol, which does not execute mentioned error checks, but ensures faster and non delayed communication.
In summary, the PerceptionNeuronROSserial program first builds a connection to the Axis Neuron data server. Then, it receives the action data in BVH format through a callback function, which continuously fills a local buffer with the new motion frames. Finally, the PerceptionNeuronROSserial parses the movement data using NeuronDataReaderSDK and builds the proper ROS message format, that is published to the *rosserial_ server* node running in ROS.

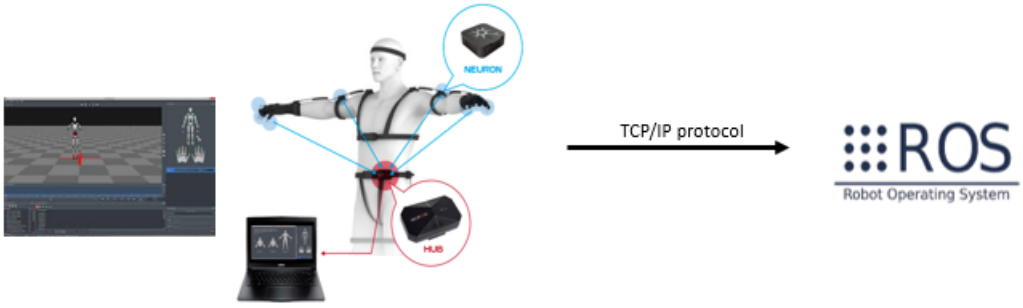


Figure 4.9: From Windows OS to Linux OS.

4.3.3 Importing movement data to ROS framework: *rosserial* ROS package

ROS *rosserial* package provides a protocol for sending messages over serial ports or network sockets. The rosserial package consists of a server and a client.
The *rosserial_windows* supports for communicating with Windows applications and it provides all necessary library files for a standalone ROS server in the Windows project PerceptionNeuronROSserial. The library provides some functionalities, like the connection with the ROS master and the interface to communicate with the *rosserial_server*.
The communication interface between the Windows and ROS environments provided by the rosserial_windows follows the same operating procedure of the publisher/subscriber in ROS: in order to publish messages, the Windows c++ program establishes connection with the duplicated ROS master, creates a publisher and publishes the data on the topic.
The *rosserial_server* creates a server which runs in ROS and handles the transmission of messages throught a TCP/IP communication protocol. Data are published by *rosserial_server* node to a topic called */perception_neuron/data*.

4.3.4 Writing body segments poses to the ROS *tf* system: *perc_neuron_tf_broadcaster* ROS package

The *perc_neuron_tf_broadcaster* node is part of the *perc_neuron_tf_broadcaster* package: *perc_neuron_tf_broadcaster* node subscribes to the topic */perception_neuron/data* to read the data acquired using the Perception Neuron system and sent to ROS. A homogeneous transformation matrix expresses the coordinate transformation between two frames in a compact form. If the frames have the same origin, it reduces to the rotation matrix; if the frames have different origins, the transformation matrix contain also the information about the distance between the origins of the two different frames. The *perc_neuron_tf_broadcaster* node converts the data into tf transforms: tf transforms can be seen as homogeneous transformation matrices, and represent the relationships between coordinate frames in ROS. Each transformation represents the pose of a skeleton joint with respect to the previous one.

$$ {}^{i-1}T_i = \begin{pmatrix} {}^{i-1}_{i}R & {}^{i-1}t_i \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} r_{xx} & r_{xy} & r_{xz} & t_x \\ r_{yx} & r_{yy} & r_{yz} & t_y \\ r_{zx} & r_{zy} & r_{zz} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

The position and orientation information contained in *tf* transforms is published on the */tf* topic using the *TransformStamped* message (position: t_x, t_y, t_z quaternion: x, y, z, w). In this way, the */tf* topic contains the data about the pose of each skeleton with respect to the previous one.
Some modification to the original *perc_neuron_tf_broadcaster* node were needed to achieve the desired data frequency: in fact, it is possible for us to work with a frequency of 120 Hz because of the choice of the usage of a number of IMU sensors smaller than 9. In order not to overload the system and guarantee a working frequency of 120 Hz,

we have chosen to publish on the /tf topic only the tf transforms relative to the human joints of interest in our application: hip, spine 0, spine 1, spine 2, spine 3, shoulder, upper arm, forearm, right hand. The original node considered all the 59 joints that compose the virtual skeleton model, instead.

4.3.5 Managing the teleoperation: the *pn2ur* ROS package

A ROS package *pn2ur* containing the *pn2ur_gain_node* ROS node was developed in C++ programming language to reconstruct the pose of the human right hand with respect to the hip in the operational space, and send it in input to the robot controller.
The node is structured as follows.

- calculation of the initial orientation offset between the UR5 tool center point (TCP) frame and the hand frame in the Perception neuron kinematic chain.
- ROS publisher to */ur_hardware_interface/script_command* topic to delivery a URscript command to move the robot to initial position and then start the teleoperation;
- a 120 Hz temporized cycle where:
  - the hand pose is reconstructed from the kinematic chain of the Perception Neuron and the computed orientation offset is applied;
  - the desired TCP pose is published to the robot controller;
  - the executed trajectory is read from the data stream of the pose of the slave robot.

Here follows a detailed explanation of the sections of *pn2ur_gain_node*.

Offeset calculation and initial position

In order to compute the offset, the mean initial position of the hand with respect to the fixed reference frame at the hip is reconstructed; more details about RightHand pose reconstruction and tf2 library used to manage the trasforms among reference frames are given in the following paragraph. Here, the procedure to compute the orientation offset and send the robot to the initial pose is described.
The right hand has a specific pose in the initial phase of the teleoperation, as does the end-effector of the robot. The RightHand (the human arm end-effector) reference frame and the tool0 (the robotic arm end-effector) reference frame are different in orientation and position. For this reason, we can't command to the robot directly the position and orientation extracted from $T_{base\_PN}^{RightHand}$, because this would lead to an incorrect robot end-effector pose in the operational space.
The orientation correction is made by calculating the homogeneous rotation matrix $R_{RightHand}^{tool0}$ before the start of the teleoperation, during a sort of calibration phase.

$$R_{base}^{tool0} = {R_{world}^{base}}^{-1} * \prod_{i=0}^{n} R_{UR5frames(i)}^{UR5frames(i+1)} \tag{4.2}$$

$$R_{RightHand}^{tool0} = R_{base\_PN}^{RightHand} * R_{base}^{tool0^{-1}} \tag{4.3}$$

The $R_{RightHand}^{tool0}$ contains the information about the orientation of the tool0 frame with respect to the RightHand frame, so it can be seen as an orientation offset. In order to correctly command the robot pose, the orientation offset is used to correct the orientation of the RightHand reference frame during the teleoperation phase.
The rotation matrix $R_{base\_PN}^{tool0}$ that represent the desired orientation to the robot controller is calculated using 4.4:

$$R_{base\_PN}^{tool0} = R_{base\_PN}^{RightHand} * R_{RightHand}^{tool0} \tag{4.4}$$

At the beginning of the teleoperation, the robot end-effector and the human end-effector can be in different positions in the operational space. For this reason, before the teleoperation starts, the `moveL(pose[x ,y, z, rx, ry, rz], velocity, acceleration, time_duration)` UR command is used to move the robot through a linear path in the initial position, corresponding to the position of the right hand with respect to the *base_PN* frame. The robot pose $T_{base\_PN}^{tool0}$, that is the human right hand pose corrected with the orientation offset, represents the input of the robot controller.

Reconstruction of the hand pose

The tf2 manages the relationship between coordinate frames in a tree structure buffered in time. The library lets the user transform points, vectors, etc. between any two coordinate frames at any desired point in time (Figure 4.10).
The Perception Neuron frames considered are:

PerceptionNeuronframes = {world, WorldPerceptionNeuron, Hips, Spine, Spine1, Spine2, Spine3, RightShoulder, RightArm, RightForeArm, RightHand}

The robot frames are:

UR5frames = {world, base_link, shoulder_link, upper_arm_link, forearm_link, wrist_1_link, wrist_2_link, wrist_3_link, tool0}

The human right hand pose is reconstructed with respect to a custom reference frame, named *base_PN*. The *base_PN* reference frame is constructed in such a way its axes are oriented as the ones of the base reference system of the robot.
RightHand pose in the *base_PN* reference system is calculated based on the homogeneous transformation matrices of each skeleton joint with respect to the previous one.
The $T_{base\_PN}^{RightHand}$ is calculated via forward kinematics as follows:

$$T_{base\_PN}^{RightHand} = T_{world}^{base\_PN^{-1}} * \prod_{i=0}^{n} T_{PerceptionNeuronframes(i)}^{PerceptionNeuronframes(i+1)} \tag{4.5}$$

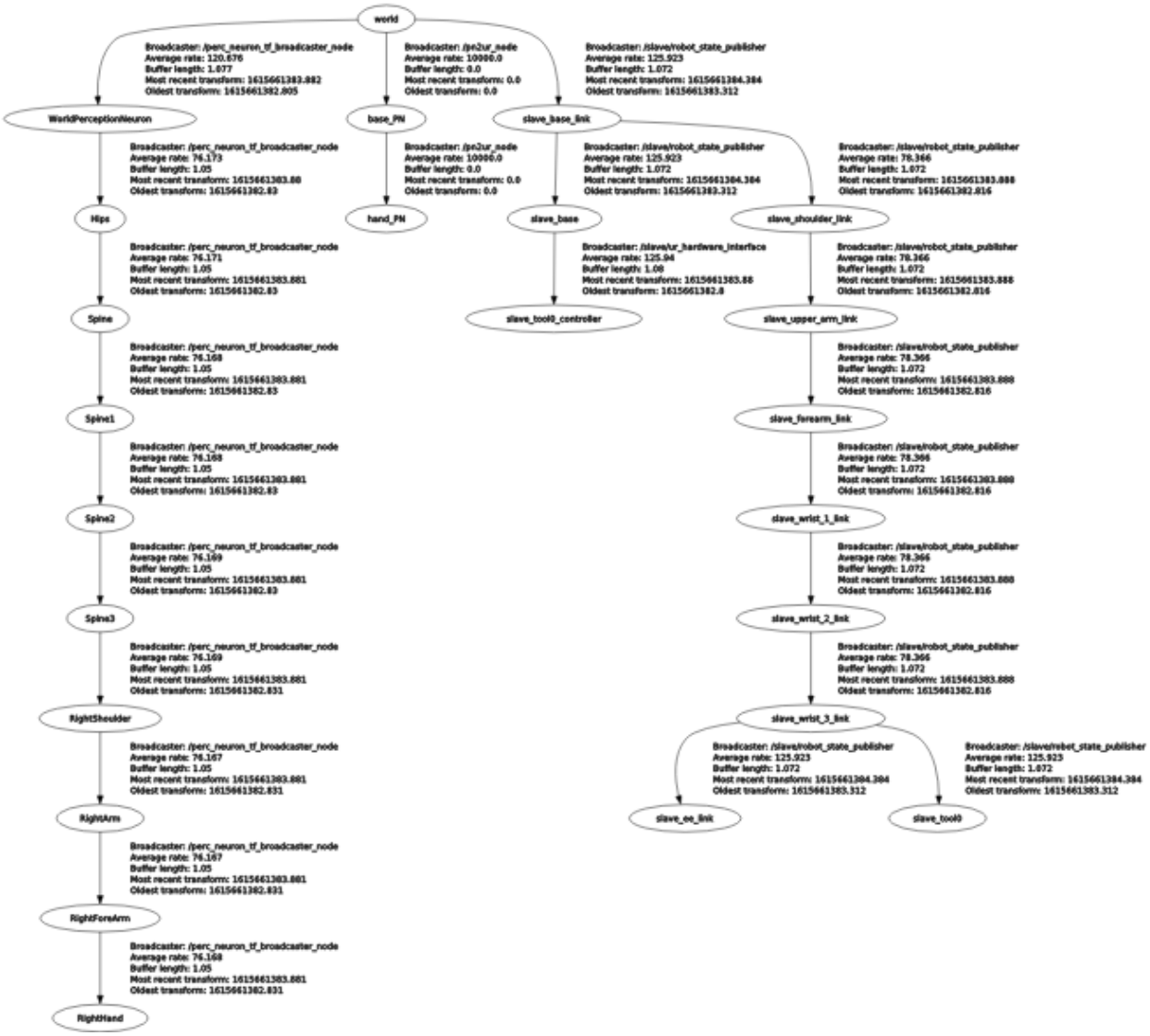


Figure 4.10: The Perception Neuron and UR5 trees.

To achieve the goal of pose reconstruction, we use the tf2 library, a second generation of the transform library. As previously said, the /tf topic contains the data about the position and orientation of each human joint with respect to the previous one. The mentioned topic records the same information about the robot joints.

The robot controller: *cartesian_motion_controller*

The selection of the proper robot controller among the one available in ROS_control was guided by the need to control the robot in its operational space: we want to send the trajectory point by point, in order to keep the possibility to check the compatibility of the trajectory with the workspace. This led us to choose the cartesian_motion_controller [25], a controller for following Cartesian target poses: it takes as input the end-effector pose point and through the inverse kinematics it reconstructs the joint pose. In this way,

the robot replicates the human right hand pose in the operational space.

4.3.6 Universal Robots Driver

To control the robot the Universal_Robots_ROS_Driver is used [26]. The driver is provided as open-source and it is compatible with different controllers managed by ros_control. To control the robot, a program node from the External Control URCap must be running on the robot interpreting commands sent from an external source.

Chapter 5

Assessment and discussion

5.1 Experiment

5.1.1 Participants

Six healthy participants (4 male and 2 female, all right-handed) aged 24 ± 6 years old took part in the experiment.
Informed consent in accordance with the Declaration of Helsinki was obtained from each participant before conducting the experiment. This study was approved by the local ethical committee of the Scuola Superiore Sant'Anna, Pisa, Italy. The methods were carried out following the approved guidelines.

5.1.2 Experimental set-up

The experimental set-up consisted of a 6-axis robotic arm used as a guide for the movement of the participants, and the teloperation system as described in Chapter 2 and Chapter 3.
Participants were asked to wear the inertial motion capture system and place their right hand on a holder attached to a collaborative robot arm (Universal Robot, model UR5 CB3). The robot that guides the movement was programmed through Polyscope 3.15 application to execute precise motions: translations along the three Cartesian axes, rotations around the three Cartesian axes and a random trajectory which is a combination of translations and rotations.

5.1.3 Experimental protocol

Participants are asked to distance all magnetic devices to minimize the magnetic interference with the inertial motion capture system. Successively, the participants worn the Perception Neuron system and the system was connected to the windows PC thanks to the USB cable.
In the Axis Neuron software, the hip was selected as contact point and the Upper Body as configuration. The correct height of the subject was set to have a more accurate reconstruction of the movement. The neurons were then calibrated executing the 3 steps calibration as reported in the Axis Neuron user manual: Steady Pose, A Pose and T pose.

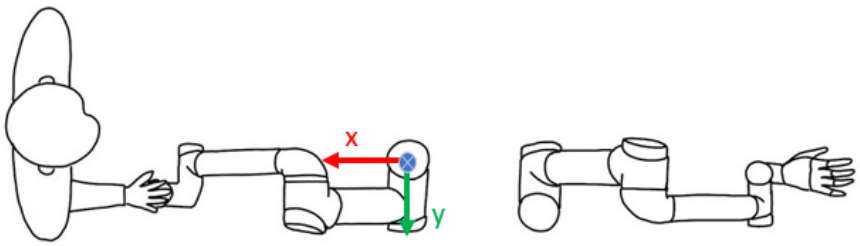


Figure 5.1: Experimental setup: on the left the robot programmed to perform precise movements and to guide the human right hand; on the right the slave robot which executes the poses reconstructed from the Perception Neuron motion system. It is reported the base reference system of the programmed robot.

In this test, the participants were seated comfortably on a chair in front of the programmed 6-axis robotic arm with their hand placed on the holder attached to the programmed 6-axis collaborative robotic arm (Figure 5.1). Once the subjects sat in the chair, the zero out error button was pressed to zeroing hip x and y coordinate.
Participants were asked to passively follow the movements performed by the programmed robotic arm to which they were linked thanks to a holder. Three different typologies of movements were performed: translations along the three Cartesian axes, rotations around the three Cartesian axes and a random trajectory which is a combination of translations and rotations.
Seven repetitions of the same movement were performed. A group of 7 repetitions composed a series. In total, three series were executed and calibration procedure was carried out between series in order to make them comparable. The calibration between the series included only two steps of the process: the A Pose and the T Pose. The test was performed at two different speeds: 250 mm/s and 1000 mm/s.

The *translations* along the three Cartesian axes involved the following steps:

– a translation of 30 cm from the starting position along the x-axis of the robotic base reference system in the positive direction;

– a pause of 1.5 s;

– a translation of 30 cm from the actual position along the x-axis of the robotic base reference system in the opposite direction;

– a pause of 1.5 s;

– a translation of 30 cm from the starting position along the y-axis of the robotic base reference system in the positive direction;

– a pause of 1.5 s;

– a translation of 30 cm from the actual position along the y-axis of the robotic base reference system in the opposite direction;

– a pause of 1.5 s;

– a translation of 30 cm from the starting position along the z-axis of the robotic base reference system in the positive direction;

– a pause of 1.5 s;

– a translation of 30 cm from the actual position along the z-axis of the robotic base reference system in the opposite direction;

– a pause of 1.5 s.

The total duration of a series was 2.5 min with a movement speed of 250 mm/s, 1.5 min with a movement speed of 1000 mm/s.

The *rotations* around the three Cartesian axes was performed as follows:

– a rotation of 45° from the starting position around the x-axis of the robotic base reference system in the negative direction;

– a pause of 1.5 s;

– a rotation of 45° from the actual position around the x-axis of the robotic base reference system in the opposite direction;

– a pause of 1.5 s;

– a rotation of 45° from the starting position around the y-axis of the robotic base reference system in the positive direction;

– a pause of 1.5 s;

– a rotation of 45° from the actual position around the y-axis of the robotic base reference system in the opposite direction;

– a pause of 1.5 s;

– a rotation of 45° from the starting position around the z-axis of the robotic base reference system in the positive direction;

– a pause of 1.5 s;

– a rotation of 45° from the actual position around the z-axis of the robotic base reference system in the opposite direction;

– a pause of 1.5 s.

The total duration of a series was 1.5 min with a movement speed of 250 mm/s, 1 min with a movement speed of 1000 mm/s.
The *trajectory* was chosen to combine translations and rotation movements. Each repetition of this movement was followed by a pause of 1.5 s. The total duration of a series was 50 s with a movement speed of 250 mm/s and 30 s with a movement speed of 1000 mm/s.

5.1.4 Data analysis

The aim of the presented protocol was the evaluation of the performances of the developed teleoperated hand-arm robotic platform. The performances have been evaluated in terms of *precision*, *accuracy*, *time delay* and *drift*. To reach this aim three different signals were recorded over the experiment time: the pose of the programmed collaborative robot, named ideal pose, the reconstructed pose from the Axis Neuron software, named *reconstructed pose*, and, finally, the *executed pose* by the slave robot, named executed pose. The evaluation of performances has been done by comparing the comparison between the ideal and the reconstructed pose and the ideal and the executed pose.

Accuracy For the evaluation of the accuracy, we had identified the static system conditions over a series. This conditions corresponds to the pause of 1.5 s between two opposite translations. We averaged the ideal, reconstructed and executed signals in the identified intervals, and we used those values in order to compute the accuracy in the following equations:

$$A_r(i-th\ rep) = \left(1 - \left|\frac{|reconstructedPose(i-th\ rep)| - |idealPose(i-th\ rep)|}{|\Delta idealPose(i-th\ rep)|}\right|\right) * 100 \tag{5.1}$$

$$A_e(i-th\ rep) = \left(1 - \left|\frac{|executedPose(i-th\ rep)| - |idealPose(i-th\ rep)|}{|\Delta idealPose(i-th\ rep)|}\right|\right) * 100 \tag{5.2}$$

Finally, a mean of the 7 obtained accuracy values has been calculated.

Drift The drift was evaluated from the linear regression of the previously calculated accuracy values. The angular coefficient of the calculated straight line represents an indicator of the drift. In absence of drift, a straight line parallel to the x-axis would be obtained.

Precision Knowing the average values of the reconstructed and executed pose in static conditions, the calculation of the precision was performed using the following equations for each axis:

$$P_r = \left(1 - \left|\frac{STD(reconstructedPose)}{mean(reconstructedPose)}\right|\right) * 100 \tag{5.3}$$

$$P_e = \left(1 - \left|\frac{STD(executedPose)}{mean(executedPose)}\right|\right) * 100 \tag{5.4}$$

Time delay The time delay was evaluated by means of the cross-correlation between the ideal and the reconstructed pose signals and between the ideal and the executed ones. For two discrete-time sequences, cross-correlation is defined as:

$$R_{xy}[n] = \sum_{m=-\infty}^{+\infty} x^*[m]y[n+m] \tag{5.5}$$

The delay in samples has been converted into time delay by exploiting the information about the sampling frequency of our signals.

Statistical analysis

This study was characterized by two factors, namely the series and movement velocities (250 mm/s and 1000 mm/s). Hence a two way repeated measures ANOVA was used to assess the effects of the series, the velocity and their interaction on the precision, accuracy and time delay.
The Kolmogorov-Smirnov test of normality was performed in order to verify that the hypothesis concerning the normality of the data was respected. This hypothesis must be verified before performing the ANOVA test. When the data violated the sphericity assumption, the Huynh-Feldt correction was used to adjust the degrees of freedom of the test. Between possible corrections, the Huynh-Feldt adjustment was chosen because of the smallness of the sample size. Statistical significance was defined for p value < 0.05.
For the drift evaluation, the t-test was performed. The considered null hypothesis was $H_0 = 0$. Statistical significance was defined for p value<0.05.
In the boxplots reported, the following notation is used:

- * : significant, p < 0.05;
- ** : very significant, p < 0.01;
- *** : highly significant, p < 0.001.

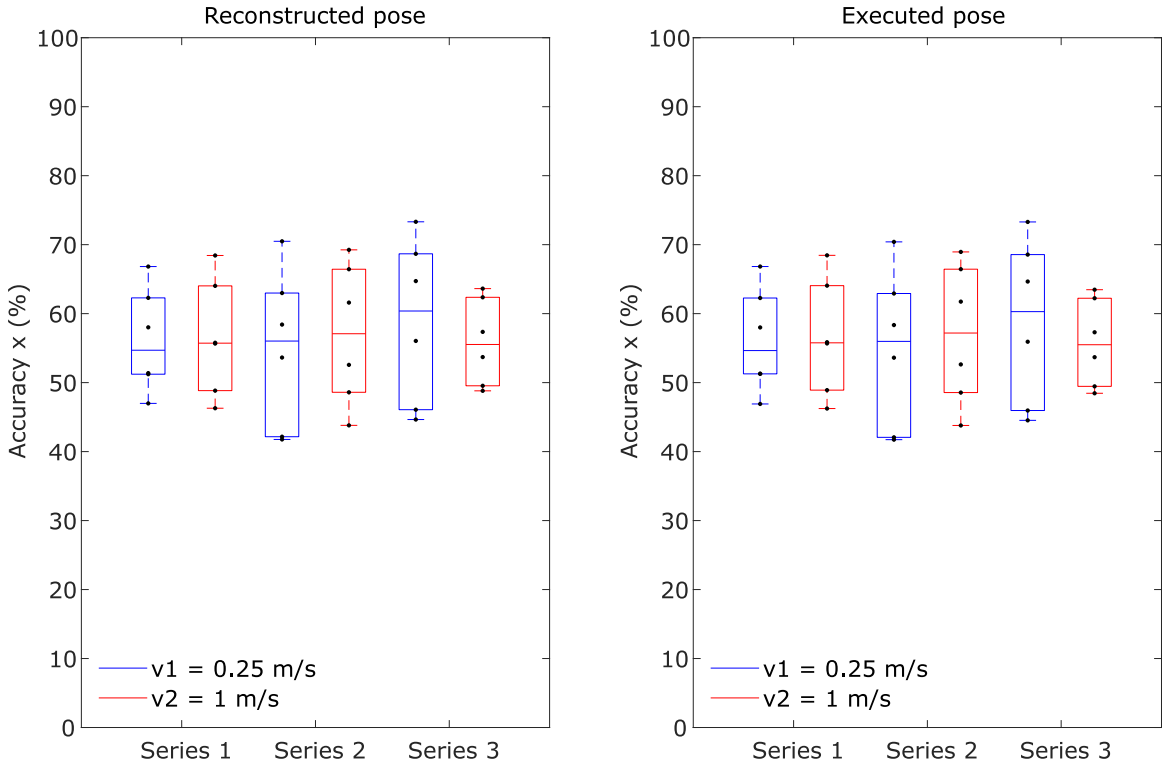
5.2 Results

The results will be reported throughout this section.

5.2.1 Accuracy

Translations

The system accuracy varies in function of the axis along the movement is performed. The accuracy is higher for movement carried out along the y-axis and the z-axis (Figures 5.3 and 5.4).



(a) (b)

Figure 5.2: The movements performed were translations along the x-axis. The boxplots show the accuracy for the reconstructed (5.2a) and the executed pose (5.2b). There is no effect of the series factor, velocity one and their interaction.

The average accuracy for the reconstructed pose and the executed one is the same and equal to:

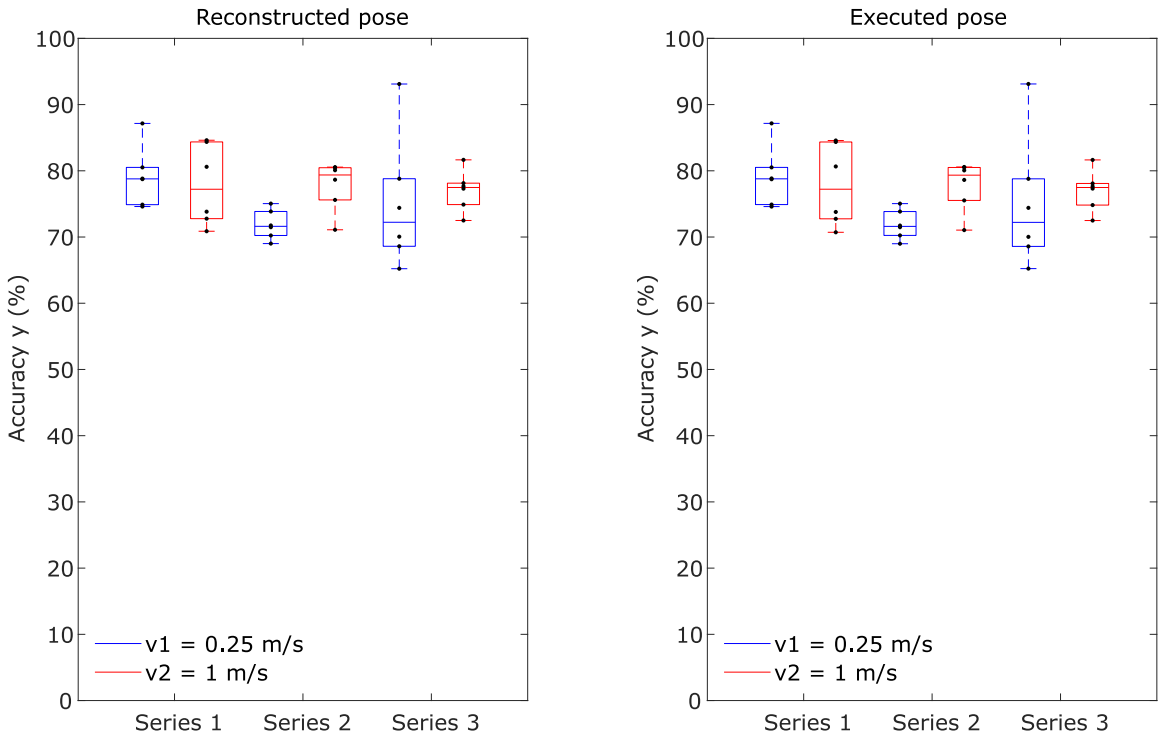
- x-axis: 56%
- y-axis: 76%
- z-axis: 73%

A statistical analysis was conducted. The ANOVA showed a non-significant influence of the series, velocity and their interaction on the system accuracy in each axis.

Rotations

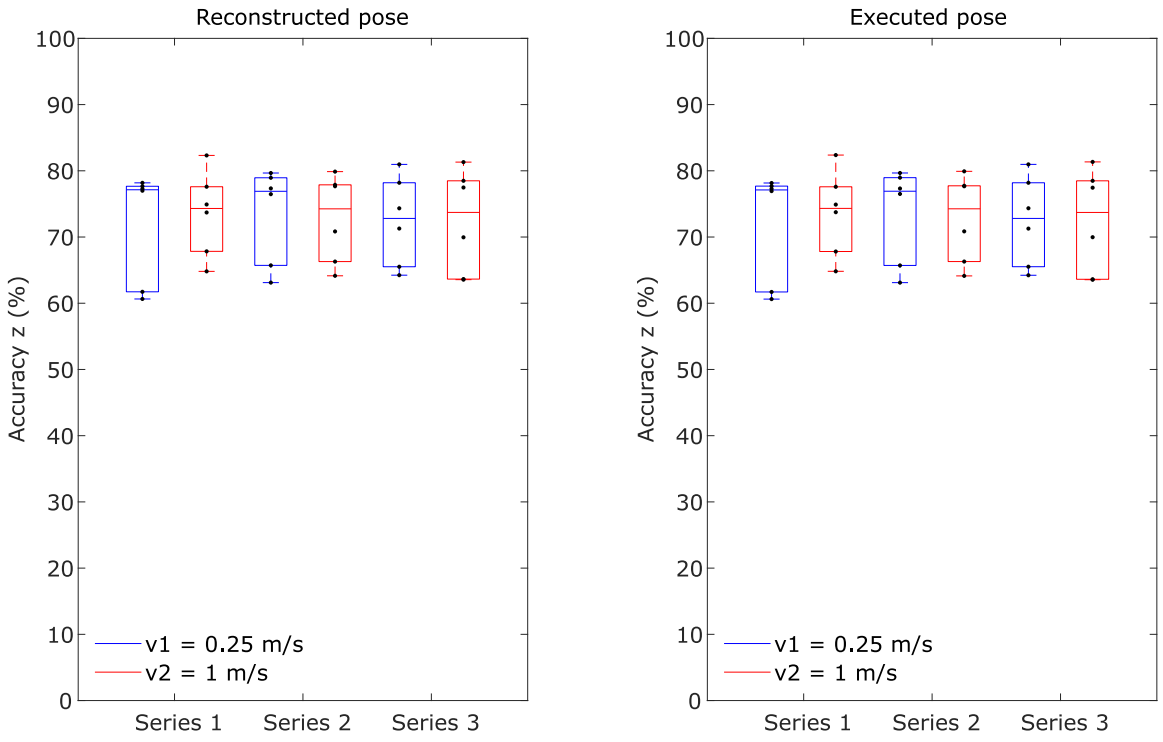
The accuracy of the system varies according to the axis around which the movement is performed. The accuracy is better for movement carried out along the x-axis (Figure 5.5). The average accuracy for the reconstructed pose and the executed one is the same and equal to:

- x-axis: 95%



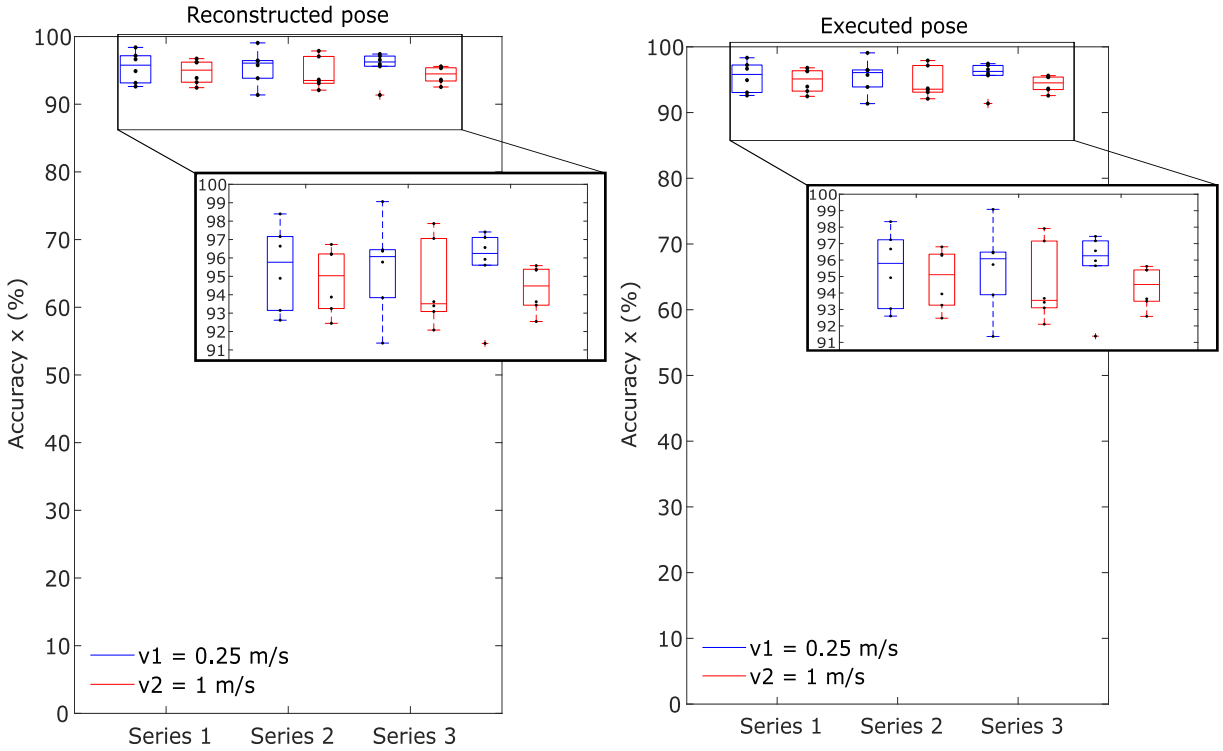
(a) (b)

Figure 5.3: The movements performed were translations along the y-axis. The boxplots show the accuracy for the reconstructed (5.3a) and the executed pose (5.3b). There is no effect of the series factor, velocity one and their interaction.



(a) (b)

Figure 5.4: The movements performed were translations along the z-axis. The boxplots show the accuracy for the reconstructed (5.4a) and the executed pose (5.4b). There is no effect of the series factor, velocity one and their interaction.

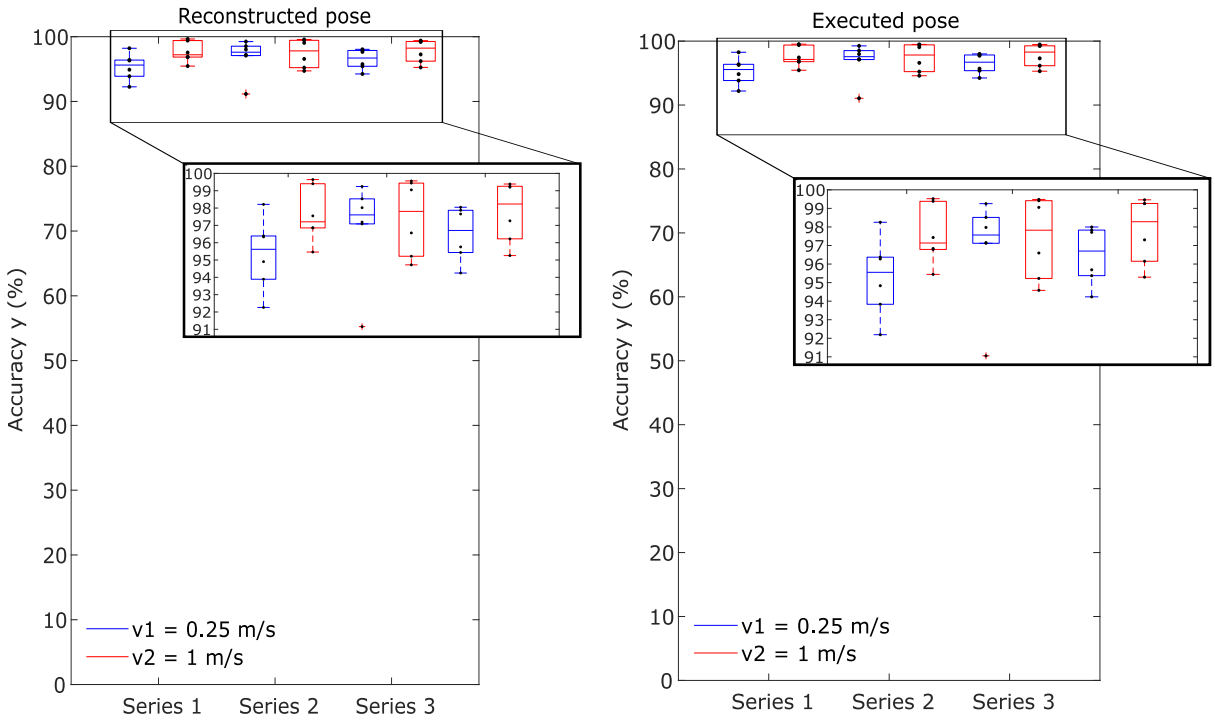


(a) (b)

Figure 5.5: The movements performed were rotations around the x-axis. The reported velocities corresponds to the velocities imposed to the movel command through the teach pendant. The equivalent angular velocity is 60°/s.The boxplots show the accuracy for the reconstructed (5.5a) and the executed pose (5.5b). There is no effect of the series factor, velocity one and their interaction.

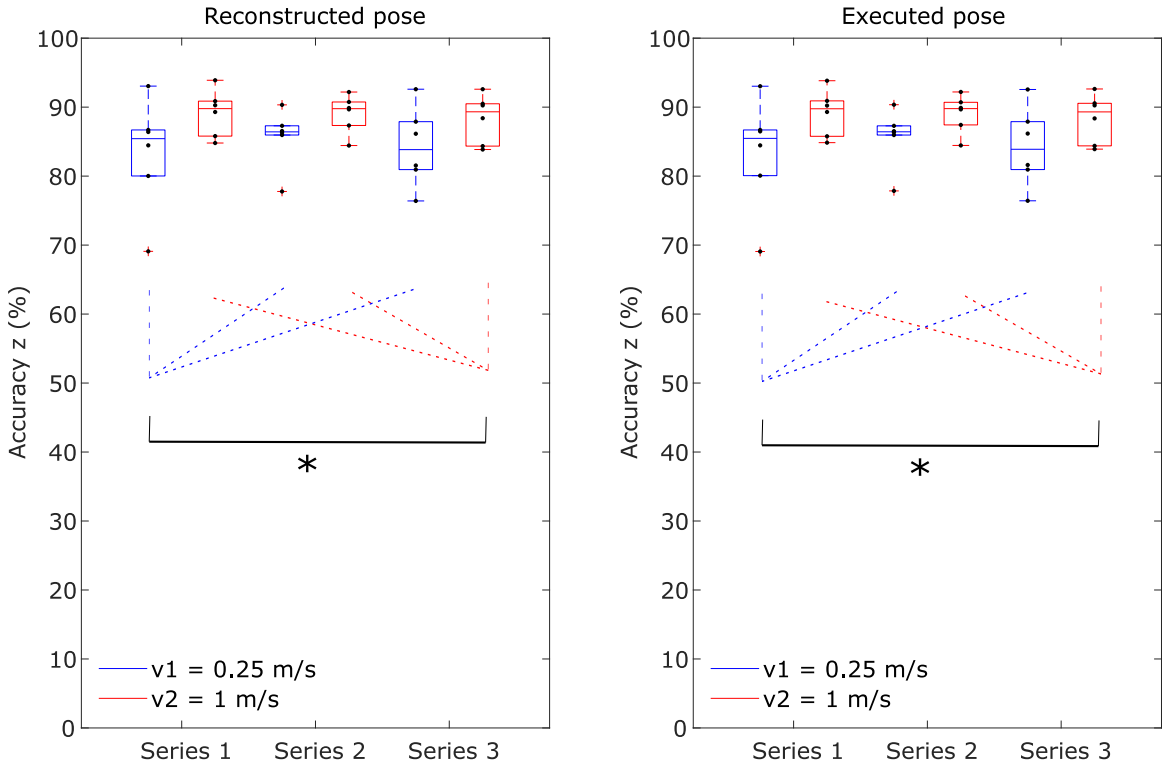
- y-axis: 97 %
- z-axis: 87%

As regards the translations along the z-axis, the ANOVA shows that the accuracy of the reconstructed pose and the executed one (respectively A_r and A_e) were affected by the velocity factor (repeated ANOVA, $F(1,5) = 8.707$, $p = 0.032$ for the reconstructed pose; repeated ANOVA, $F(1,5) = 8.748$, $p = 0.032$ for the executed pose), however, there was no influence of the series and there was no interaction between the two factors (Figure 5.7).
The post-hoc tests revealed an higher accuracy when the velocity of the programmed movement is equal to 1000 mm/s ($p < 0.05$) (Figure 5.7). We hypothesize that this can be due to the functioning of the sensors.



(a) (b)

Figure 5.6: The movements performed were rotations around the y-axis. The reported velocities corresponds to the velocities imposed to the movel command through the teach pendant. The equivalent angular velocity is 60°/s. The boxplots show the accuracy for the reconstructed (5.6a) and the executed pose (5.6b). There is no effect of the series factor, velocity one and their interaction.



(a) (b)

Figure 5.7: The movements performed were rotations around the z-axis. The reported velocities corresponds to the velocities imposed to the movel command through the teach pendant. The equivalent angular velocity is 60°/s. The boxplots show the accuracy for the reconstructed (5.7a) and the executed pose (5.7b). The post-hoc tests revealed a higher accuracy when the velocity of the programmed movement is equal to 1000 mm/s ($p < 0.05$).

5.2.2 Precision

Translations

The system precision varies in function of the axis along which the movement is performed. The precision is better for the translations performed along the z-axis (Figure 5.10).
The average precision for the reconstructed pose and the executed one is the same and equal to:

- x-axis: 94%
- y-axis: 94%
- z-axis: 99%

As regards the translation along the z-axis, the precision of the reconstructed pose was affected by the series (repeated ANOVA, $F(2, 10) = 6.349$, $p = 0.0017$), however, there was no influence of the velocity and there was no interaction between the two factors.
In the same way, the precision of the executed pose was affected by the series (repeated ANOVA, $F(2, 10) = 6.476$, $p = 0.0016$), however, as the previous case, there was no influence of the velocity and there was no interaction between the two factors.
However, both for the reconstructed pose and the executed one, the post-hoc tests revealed a no significant influence of the series factor.

Rotations

Both the reconstructed pose and the executed one results showed an higher precision in the case of rotation movements around the x and the y-axis. The average precision for the reconstructed pose and the executed one is the same and equal to:

- x-axis: 98%
- y-axis: 98%
- z-axis: 96%

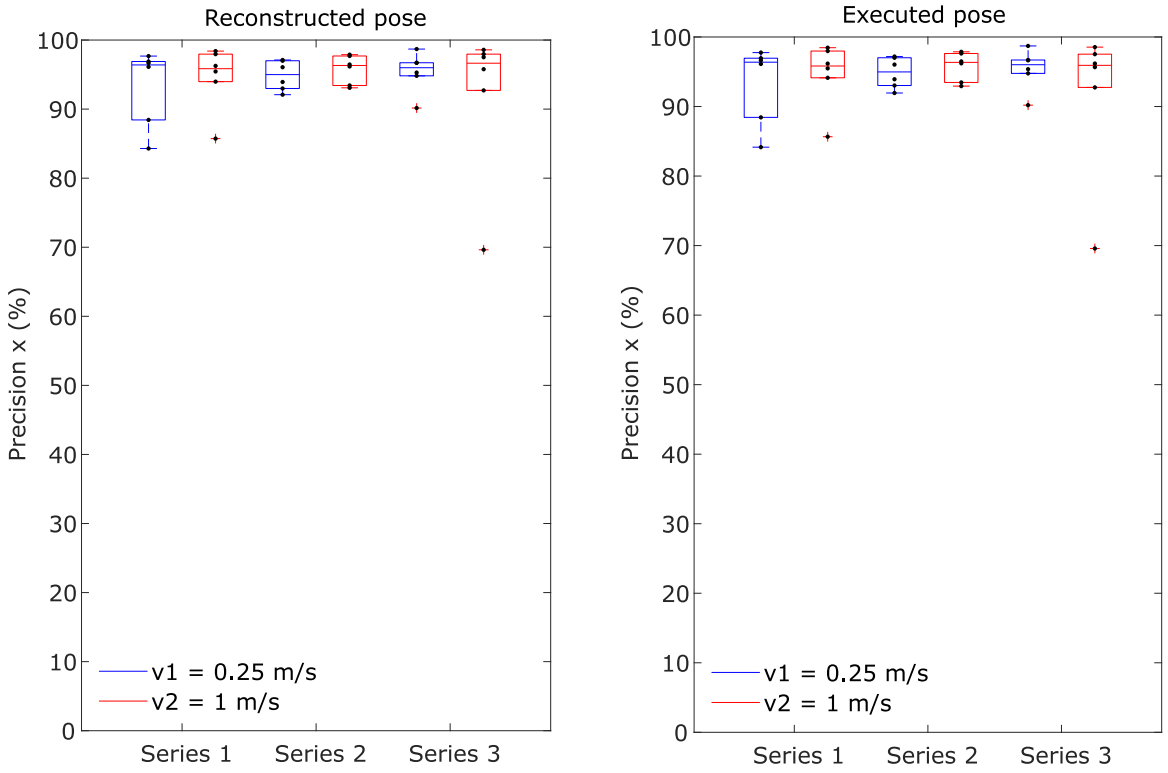
The ANOVA showed a non-significant influence of the series, velocity and their interaction on the precision for rotations around axes.

5.2.3 Time delay

Translations

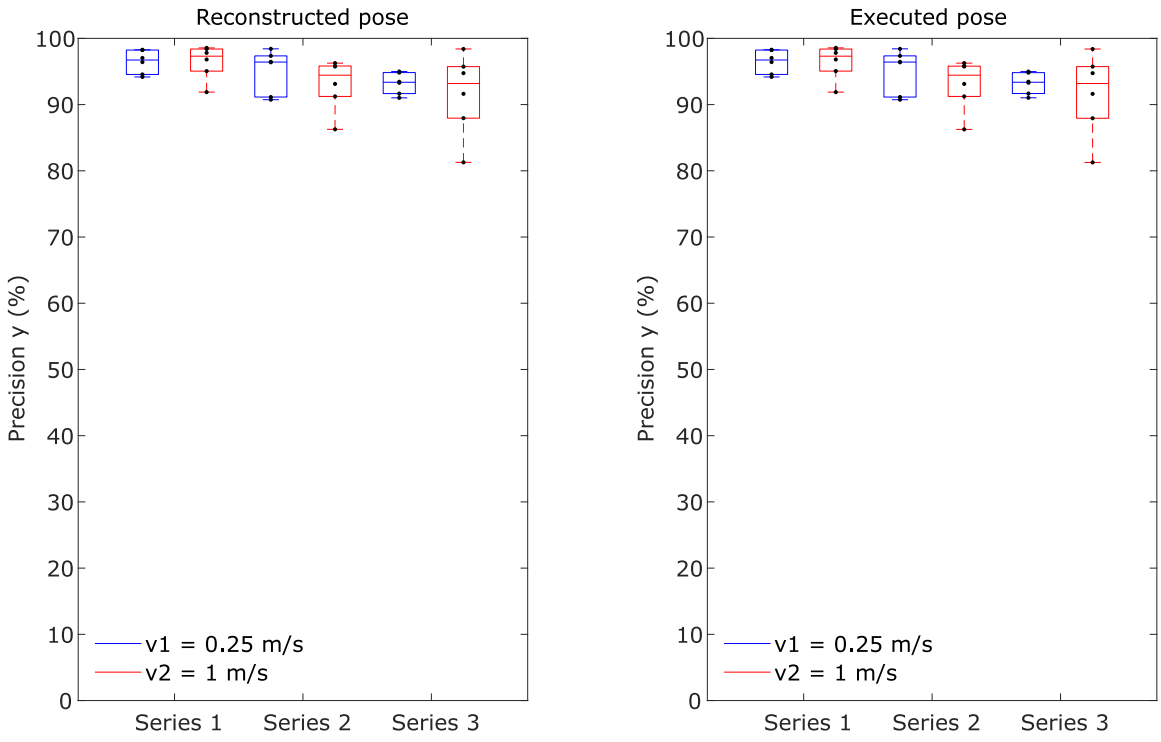
The average time delay of the reconstructed pose with respect to the ideal one is equal to:

- x-axis: 72 ms
- y-axis: 67 ms
- z-axis: 72 ms



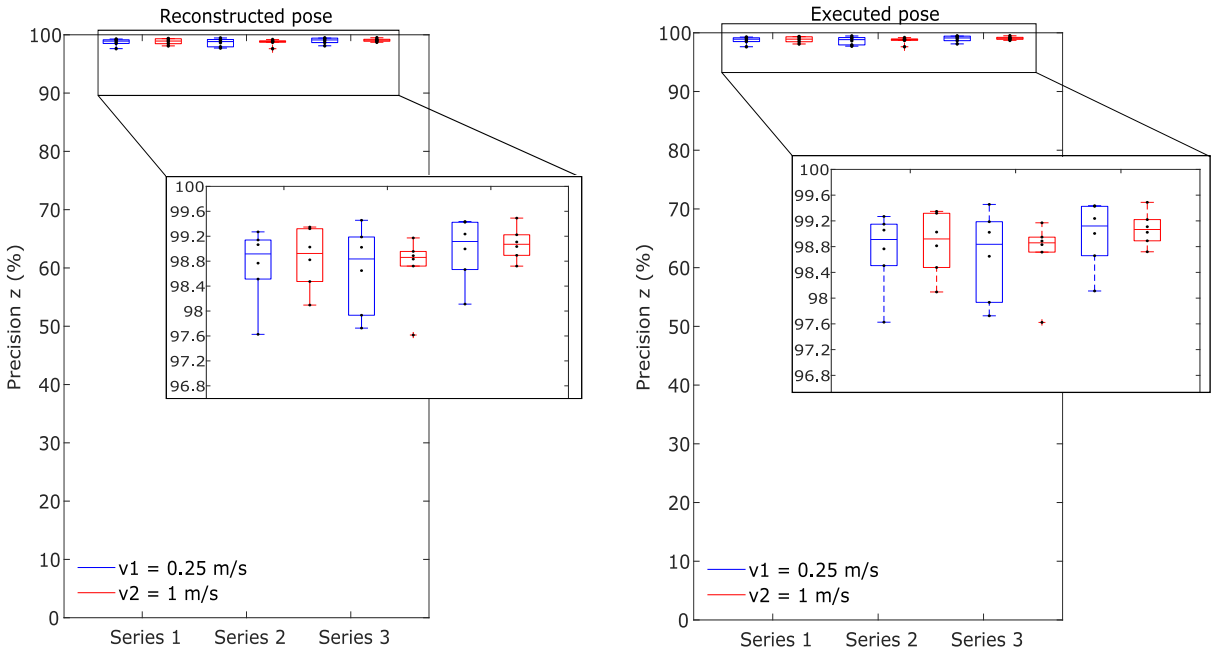
(a) (b)

Figure 5.8: The boxplots represent the precision on the translation along x-axis for the reconstructed (5.8a) and the executed pose (5.8b). There is no effect of the series factor and velocity one.



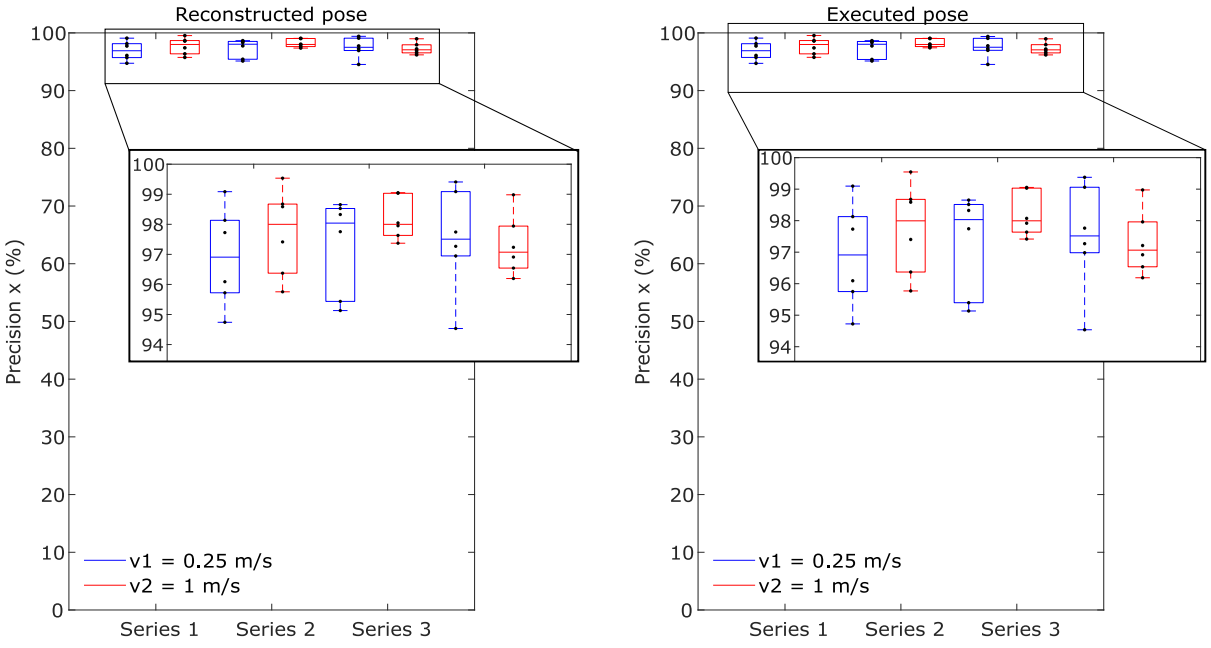
(a) (b)

Figure 5.9: The boxplots represent the precision on the translation along y-axis for the reconstructed (5.9a) and the executed pose (5.9b). There is no effect of the series factor and velocity one.



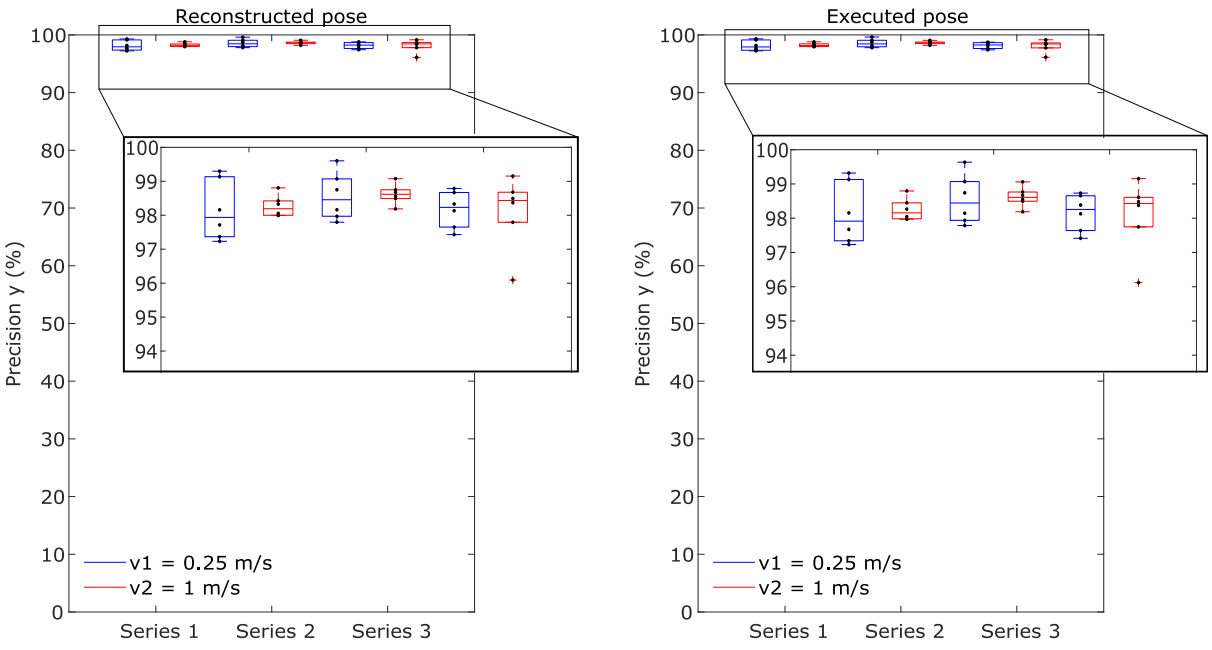
(a) (b)

Figure 5.10: The boxplots represent the precision on the translation along z-axis for the reconstructed (5.10a) and the executed pose (5.10b). Even if the ANOVA revealed an effect of the series factor, post-hoc tests revealed the p values higher than 0.05.



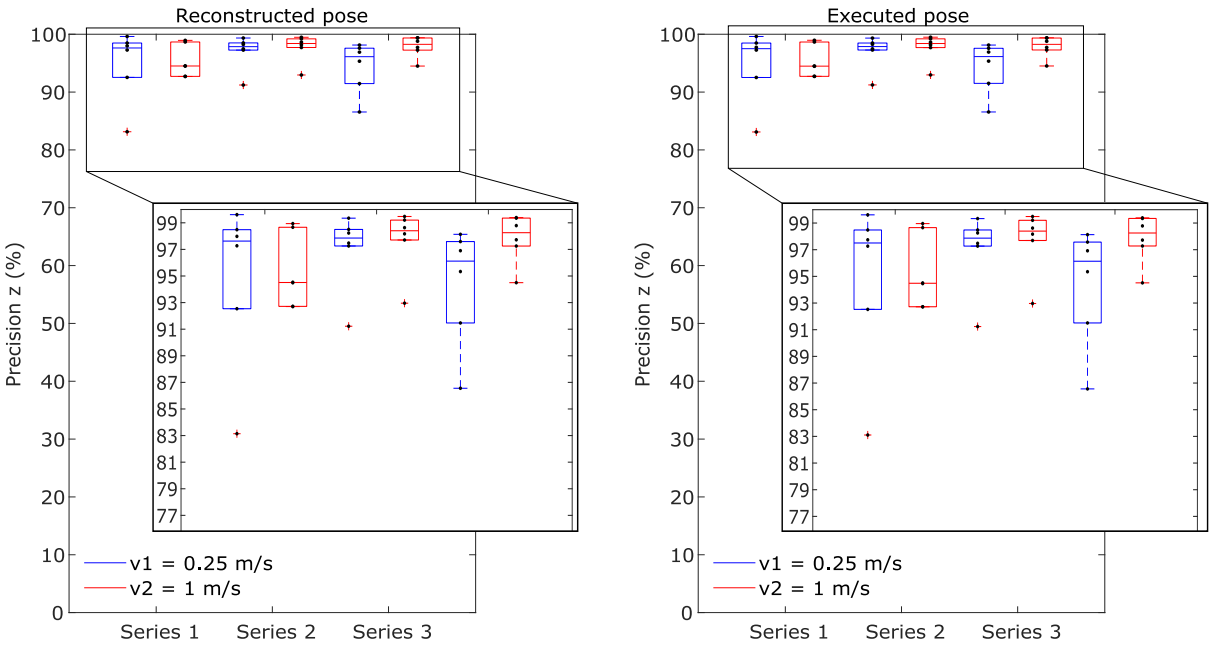
(a) (b)

Figure 5.11: The reported velocities corresponds to the velocities imposed to the movel command through the teach pendant. The equivalent angular velocity is 60°/s. The boxplots represent the precision on the rotation around x-axis for the reconstructed (5.11a) and the executed pose (5.11b). There is no effect of the series factor and velocity one.



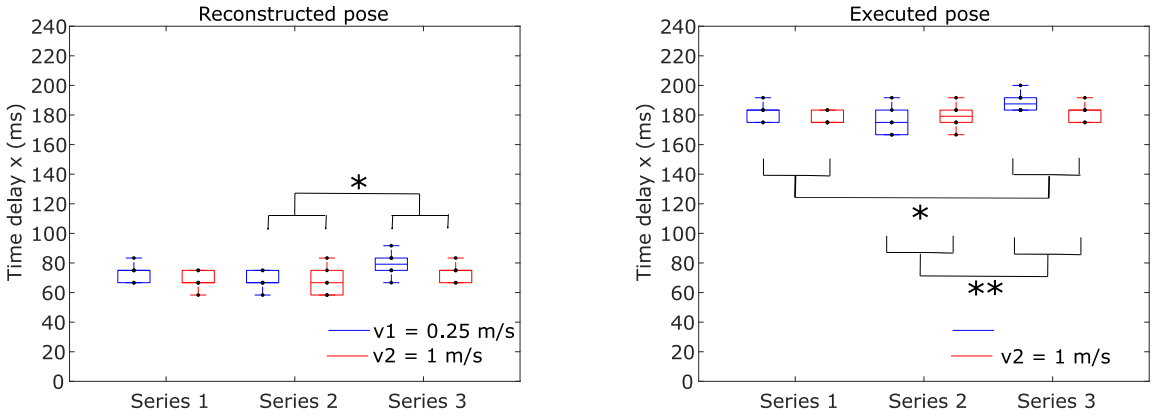
(a) (b)

Figure 5.12: The reported velocities corresponds to the velocities imposed to the movel command through the teach pendant. The equivalent angular velocity is 60°/s. The boxplots represent the precision on the rotation around y-axis for the reconstructed (5.12a) and the executed pose (5.12b). There is no effect of the series factor and velocity one.



(a) (b)

Figure 5.13: The reported velocities corresponds to the velocities imposed to the movel command through the teach pendant. The equivalent angular velocity is 60°/s. The boxplots represent the precision on the rotation around z-axis for the reconstructed (5.13a) and the executed pose (5.13b). There is no effect of the series factor and velocity one.



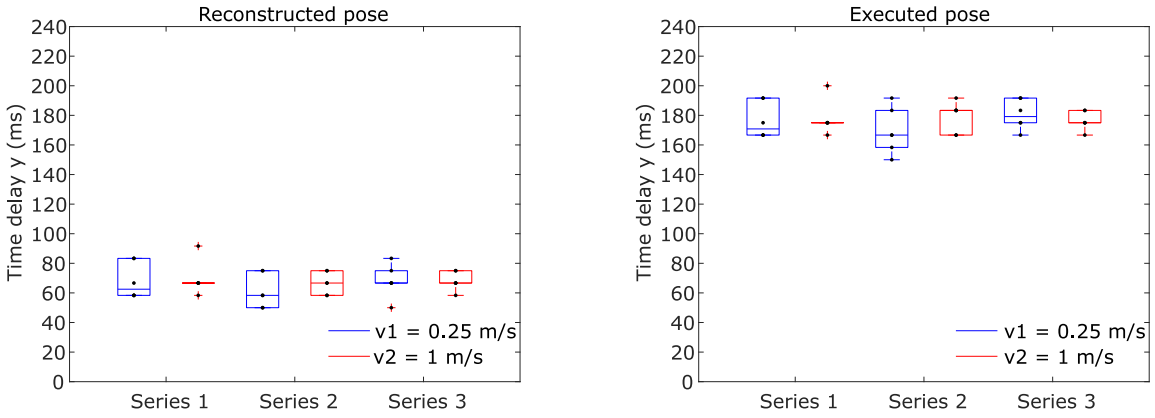
(a) (b)

Figure 5.14: The boxplots represent the time delay on the translation along x-axis for the reconstructed (5.14a) and the executed pose (5.14b). It is been highlight the effect of the series factor on time delay for the reconstructed pose and the executed one.

The average time delay the executed pose with respect to the ideal pose is equal to:

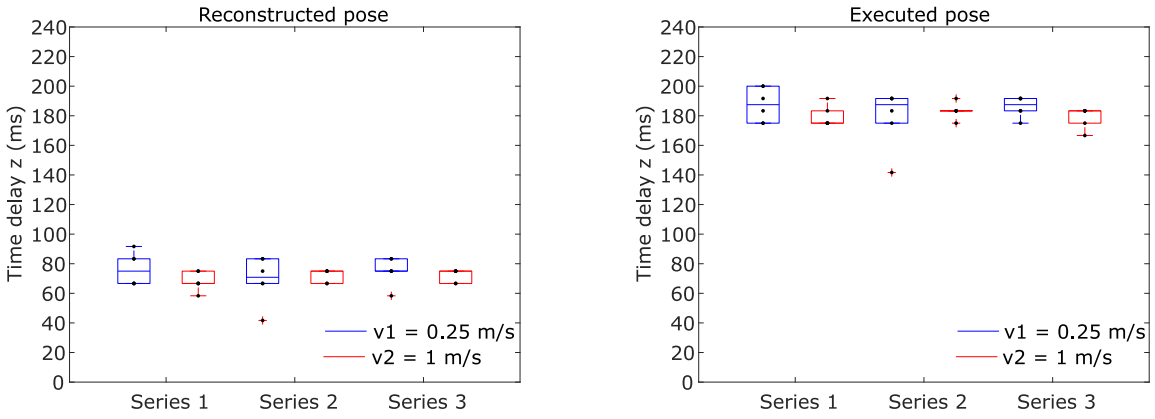
- x-axis: 181 ms
- y-axis: 177 ms
- z-axis: 182 ms

As regards the translation along the x-axis, the time delay of the reconstructed pose with respect to the ideal one (ΔT_r) was affected by the series factor (repeated ANOVA, $F(1.882, 9.411) = 5.714$, $p = 0.025$), however, there was no influence of velocity and there was no interaction between the two factors. The post-hoc test revealed an higher time delay between the reconstructed and the ideal pose in the third series compared to the second one ($p < 0.05$) (Figure 5.14a).
As regards the time delay of the executed pose with respect to the ideal one (ΔT_e) was affected by series (repeated ANOVA, $F(1.358, 6.791) = 10.319$, $p = 0.012$), however, there was no influence of the velocity and series. The post-hoc test revealed an higher time delay between the executed and the ideal pose in the third series compared to the first one ($p < 0.01$). In addition, there was a significant difference between the first and the third series ($p < 0.05$) and between the second and the third series ($p<0.01$) (Figure 5.14b).
Both for the reconstructed pose and the executed one the ANOVA showed an higher time delay for velocity 1 of series 3 and in both cases no increasing or decreasing trend has been observed, but the time delay fluctuates around an average value of 180 ms. We can hypothesize that this fluctuation may be due to an occasional lack of performance of the PC used during the execution of the protocol, which is responsible for both hardware control and data acquisition.
In order to verify this hypothesis we propose to perform further tests using a more performing PC for data acquisition.



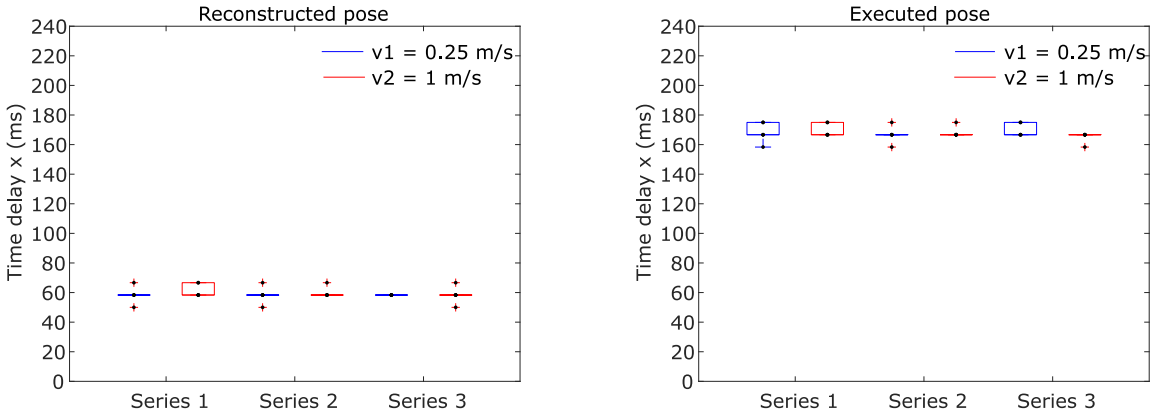
(a) (b)

Figure 5.15: The boxplots represent the time delay on the translation along y-axis for the reconstructed (5.15a) and the executed pose (5.15b). There is no effect of the series factor and velocity one.



(a) (b)

Figure 5.16: The boxplots represent the time delay on the translation along z-axis for the reconstructed (5.16a) and the executed pose (5.16b). There is no effect of the series factor and velocity one.



(a) (b)

Figure 5.17: The reported velocities corresponds to the velocities imposed to the movel command through the teach pendant. The equivalent angular velocity is 60°/s. The boxplots represent the time delay on the rotation around the x-axis for the reconstructed (5.17a) and the executed pose (5.17b). There is no effect of the series factor and velocity one.

Rotations

The average time delay of the reconstructed pose with respect to the ideal one is equal to:

- x-axis: 59 ms
- y-axis: 59 ms
- z-axis: 58.6 ms

The average time delay the executed pose with respect to the ideal pose is equal to:

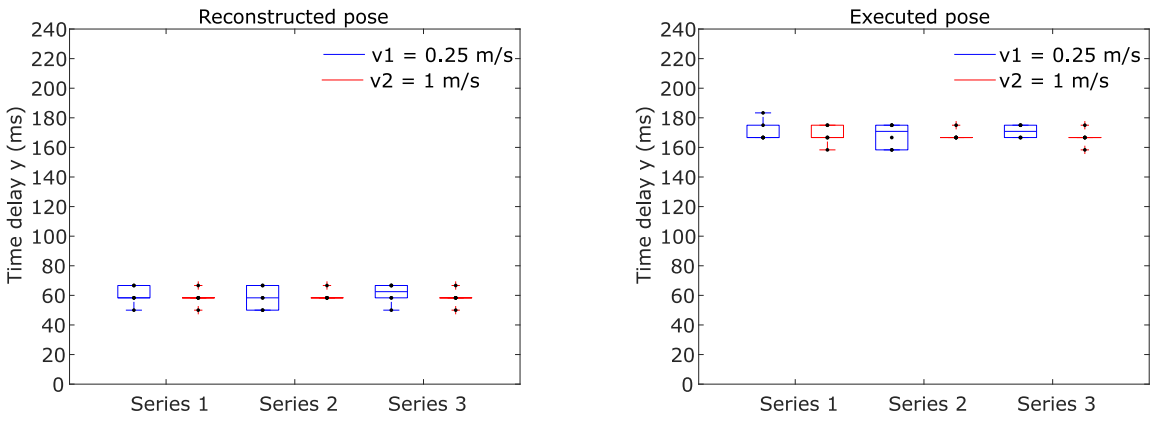
- x-axis: 167.8 ms
- y-axis: 168.8 ms
- z-axis: 168 ms

The ANOVA showed a non-significant influence of the series, velocity and their interaction on time delay.

5.2.4 Drift

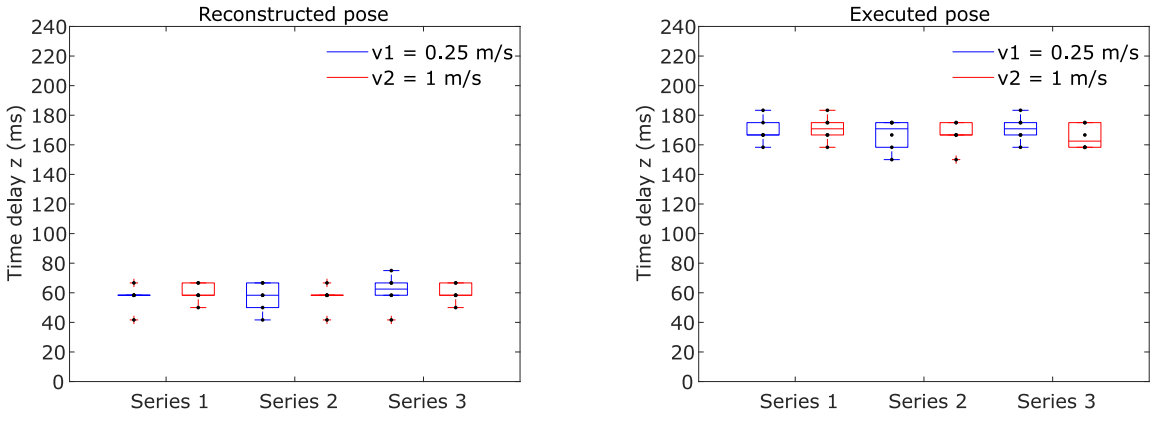
Translations

The t-test showed a significance only for the translations along the y-axis (Figure 5.21). Both for the reconstructed pose and the executed one, the significance was shown for the velocities v_1 and v_2 of the second series ($p < 0.05$) and for the velocity v_2 of the third series ($p < 0.05$).
We can hypothesize that this variability is due to a greater sensitivity of the inertial sensors to drift for movements involving the y-axis.



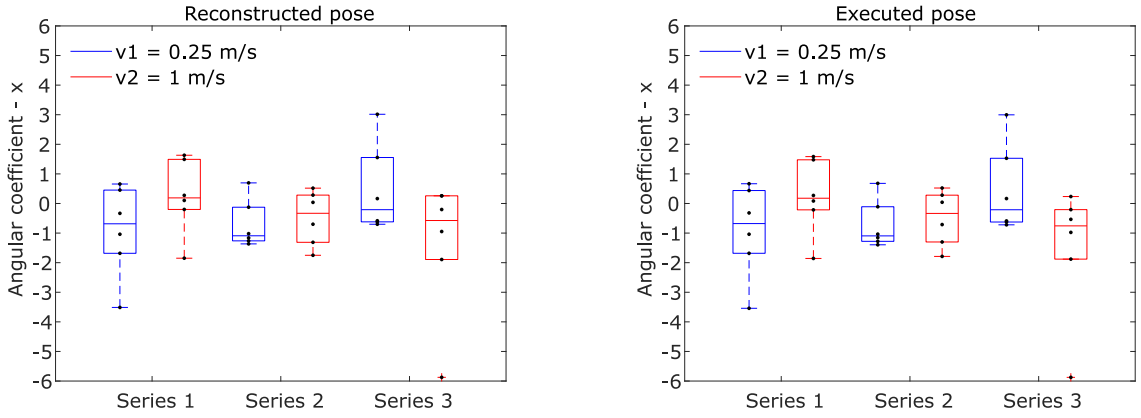
(a) (b)

Figure 5.18: The reported velocities corresponds to the velocities imposed to the movel command through the teach pendant. The equivalent angular velocity is 60°/s. The boxplots represent the time delay on the rotation around the y-axis for the reconstructed (5.18a) and the executed pose (5.18b). There is no effect of the series factor and velocity one.



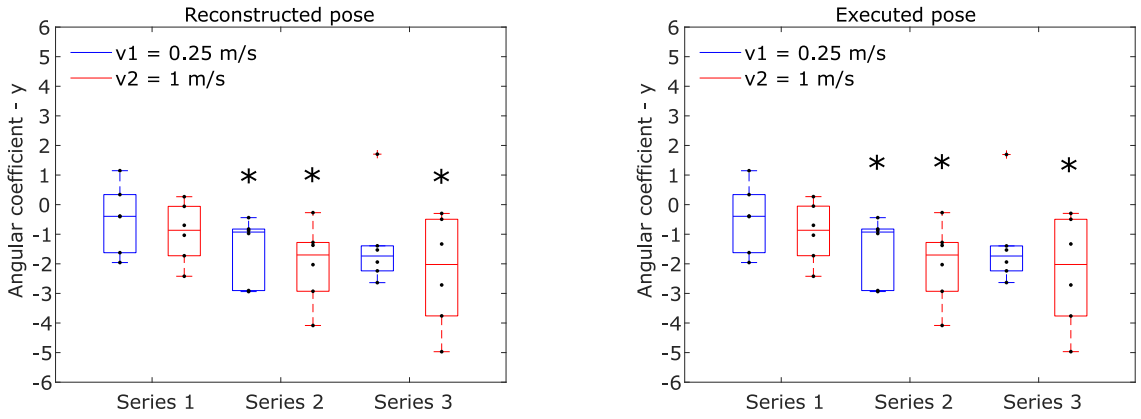
(a) (b)

Figure 5.19: The reported velocities corresponds to the velocities imposed to the movel command through the teach pendant. The equivalent angular velocity is 60°/s. The boxplots represent the time delay on the rotation around the z-axis for the reconstructed (5.19a) and the executed pose (5.19b). There is no effect of the series factor and velocity one.



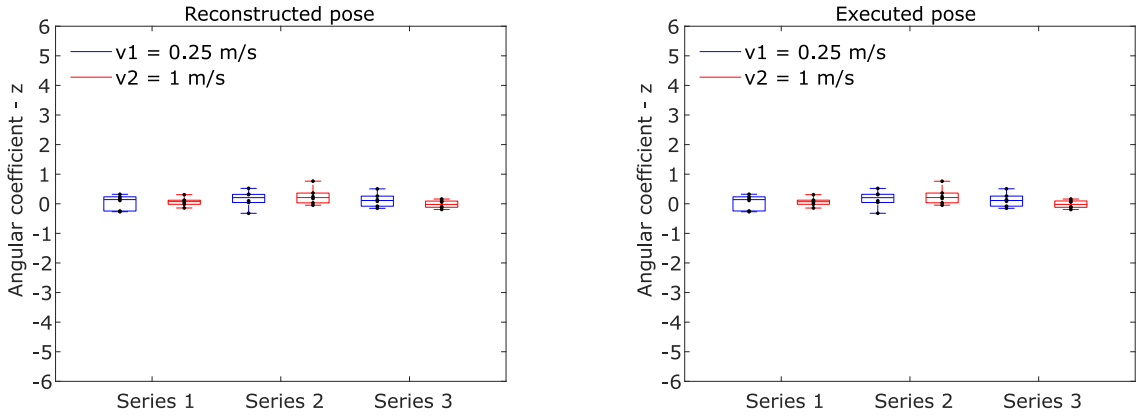
(a) (b)

Figure 5.20: The boxplots represent the angular coefficient on the translation along the x-axis for the reconstructed (5.20a) and the executed pose (5.20b). There was no significant difference from the null hypothesis for every experimental condition.



(a) (b)

Figure 5.21: The boxplots represent the angular coefficient on the translation along the y-axis for the reconstructed (5.21a) and the executed pose (5.21b). Both for the reconstructed pose and the executed one, the significance was shown for the velocities v_1 and v_2 of the second series ($p < 0.05$) and for the velocity of v_2 of the third series ($p < 0.05$).

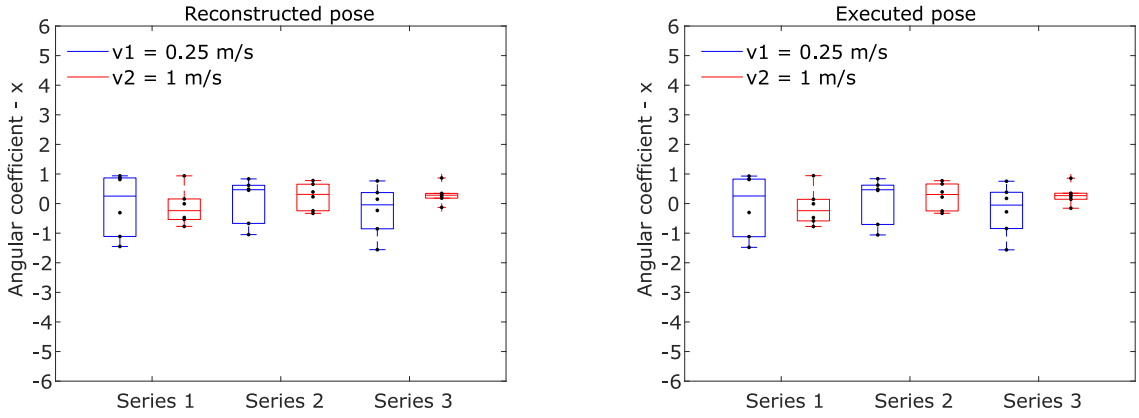


(a) (b)

Figure 5.22: The boxplots represent the angular coefficient on the translation along the z-axis for the reconstructed (5.22a) and the executed pose (5.22b). There was no significant difference from the null hypothesis for every experimental condition.

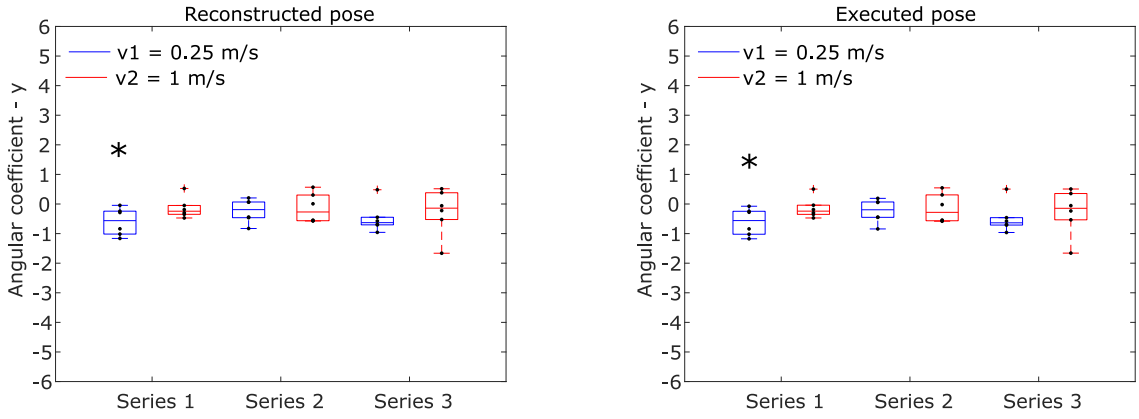
Rotations

The t-test showed a significance only for the translations along the y-axis (Figure 5.24). In both the reconstructed pose and the executed one, the significance was shown for the velocity v_1 of the first series ($p < 0.05$).
We can hypothesize that this variability is due to a greater sensitivity of the inertial sensors to drift for movements involving the y-axis.



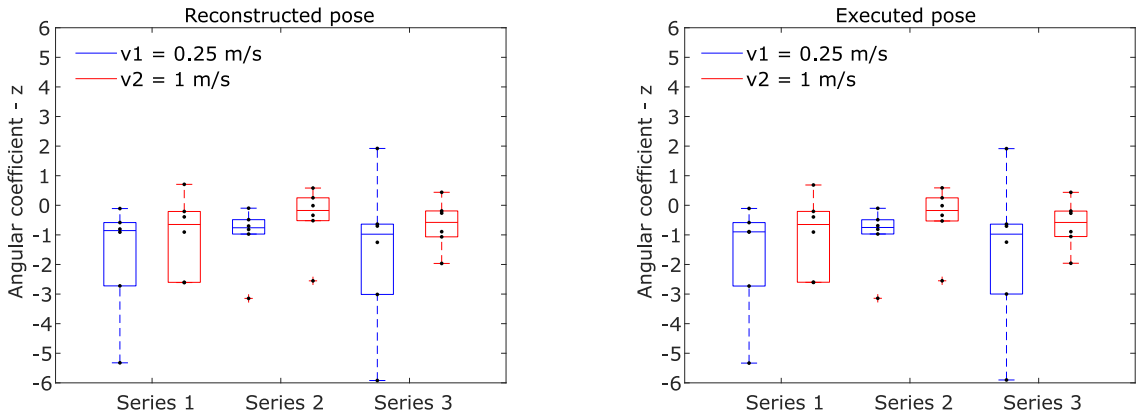
(a) (b)

Figure 5.23: The reported velocities corresponds to the velocities imposed to the movel command through the teach pendant. The equivalent angular velocity is 60°/s. The boxplots represent the angular coefficient on the rotation around the x-axis for the reconstructed (5.23a) and the executed pose (5.23b). There was no significant difference from the null hypothesis for every experimental condition.



(a) (b)

Figure 5.24: The reported velocities corresponds to the velocities imposed to the movel command through the teach pendant. The equivalent angular velocity is 60°/s. The boxplots represent the angular coefficient on the rotation around the y-axis for the reconstructed (5.24a) and the executed pose (5.24b). Both for the reconstructed pose and the executed one, the significance was shown for the velocity v_1 of the first series ($p < 0.05$).



(a) (b)

Figure 5.25: The reported velocities corresponds to the velocities imposed to the movel command through the teach pendant. The equivalent angular velocity is 60°/s. The boxplots represent the angular coefficient on the translation along the z-axis for the reconstructed (5.25a) and the executed pose (5.25b). There was no significant difference from the null hypothesis for every experimental condition.

Chapter 6

Conclusions

This work has set as its primary objective the development of a hand-arm teleoperation platform for the evaluation of shared autonomy algorithms.

The development of the system started from a first phase of hardware choice, that led me to critically evaluate the performance of different types of sensors designated for human motion analysis according to the application of interest.
To meet the need of a teleoperation system that did not constitute an obstacle to the movement of the operator during the teleoperation phase, the choice fell on wearable and non-bulky systems: the Perception Neuron motion capture system and the CyberGlove dataglove motion capture system. The first system was used to control the movement of the 6-axis collaborative robotic arm (Universal Robots, model UR5), while the output of the CyberGlove system allowed the control of the opening and closing of the robotic hand (Prensilia SRL, model Mia).
To achieve these goals, two different teleoperation algorithm have been implemented, one related to Mia robotic hand control and one related to the UR5 robotic arm control. Control algorithms have been developed in the ROS environment using the C++ programming language.

The first teleoperation algorithm is in charge of controlling the condition of opening and closing of the Mia robotic hand, starting from the condition of opening and closing of the operator's hand.
The idea behind the decoding algorithm is to calculate the Euclidean distance between the vector of sensor outputs recorded during the static phase of hand closure, and the outputs of sensors streamed by the CyberGlove during the teleoperation phase. All sensors are considered except, those related to measuring the degree of flexion/extension and abduction/adduction of the wrist, to obtain a response from the algorithm that is independent of these conditions. A threshold has been imposed on the calculated Euclidean distance to distinguish the closing condition from the hand opening condition. Based on the detected condition, the appropriate serial command is sent to Mia robotic hand.
A subject-specific initial calibration performed in MATLAB makes the algorithm robust and ensures a correct response from the robotic hand. In fact, the definition of the algo-

rithm was followed by a test phase on different subjects, aimed at a qualitative evaluation of its success and failure. The tests compared different wrist configurations, to verify the actual independence of the algorithm from particular wrist postures. The algorithm showed a failure rate of 0%.

Future developments A future goal is to adapt this algorithm to proportional control of the Mia robotic hand starting from the outputs of the CyberGlove sensors. From an in-depth study of the outputs returned by CyberGlove sensors, we would like to proportionally control three different types of Mia robotic hand grasps: cylindrical, lateral and pinch. To do this, a study should be conducted on the best way to discriminate the flexion/extension of the thumb and index independently, the flexion/extension of the index finger, middle and pinky and abduction/adduction of the thumb starting from the outputs of the CyberGlove sensors.

The second teleoperation algorithm is related to the control of the UR5 robotic arm, by mapping the centroid of the hand position and orientation to the TCP of the UR5 Arm. In this way, the robotic arm replicates the movements of the hand.
The algorithm was thought for a versatile teleoperation system, with the possibility to modulate the workspace, allowing different tasks to be carried out.
After the assembly of the entire teleoperation system, the performances in terms of accuracy, precision, time delay and drift were evaluated. To achieve this aim, an experimental protocol has been drawn up. The experimental protocol involved different subjects who were guided in the movements of translation and rotation of the hand through an additional robot that performed programmed movements; the pose reconstructed by the suit worn by the subjects was sent to the robot, that was able to follow the pose of the human right hand.
For the evaluation of performance through the metrics defined above, three different pose signals are acquired: the pose of the programmed robot that served as guide for movements (called ideal pose), the pose of the human hand in the operating space reconstructed with the IMU suit (called reconstructed pose) and the pose of the slave robot (called executed pose).
Overall, the results showed not too high system accuracy, but high precision. As concerns our application, this can be considered a very good result, as we believe that the value of our teleoperation system is given by its high precision: with a precise system, the poor accuracy can be eventually corrected thanks to the introduction of a corrective offset (or with the automatic mechanism of adaptation by the teleoperator on the basis of the visual feedback).
As concerns time delay, no particular changes have been observed within the experimental conditions. The time delay between the reconstructed pose and the ideal pose is around 70 ms, while the delay related to the pose performed compared to the ideal one is stable around 180 ms. For the purposes of manipulation tasks, the time delay imposed by the system is not perceived as a limit by the operator and reactivity can be considered satisfactory.
Drift appears differently on different axes. Based on the models that will be built from

the data collected during the execution of the experiment, it will be possible to subtract the drift from the acquired data. In addition, it will be possible to define a task duration that minimizes the drift effect.

Future developments We believe that the application of the developed teleoperation system in the research perspectives of our lab is at least twofold.
On one hand, the system allows for the evaluation of shared autonomy algorithms in teleoperation: the benefits of introducing robot autonomy in human robot collaboration has been stated, and we plan to perform analogous studies as concerns the preferences in the use of autonomous algorithms in teleoperation.
On the other hand, we plan to use the data that could be collected from the developed teleoperation in a teaching by demonstration experiment to teach a UR robot how to successfully and safely perform manipulation of deformable objects belonging to industrial tasks. In particular, the considered tasks have been defined within the April H2020 European project, that involves 15 European partners from 8 EU countries. April goal is to develop a new generation of robots to innovate ways to manage flexible and deformable materials for the production of products in European companies.
The team of the Human-Robot Interaction Laboratory of the Scuola Superiore Sant'Anna represents one of the partners of the project and works on the development of a library of automatic grasps for flexible materials applicable to different industrial use cases. The idea is to take advantage of the teleoperation system developed to let the robot learn how to grasp of flexible objects automatically after the manipulative choices of a human teleoperator who simulates the tasks.

Bibliography

[1] C. Passenberg et al. A survey of environment-, operator-, and task-adapted controllers for teleoperation systems. 2010.

[2] S. Lichiardopol. A survey on teleoperation. 2007.

[3] L. Basañez et al. Teleoperation. 2009.

[4] Honghao Lv et al. Teleoperation of collaborative robot for remote dementia care in home environments. 2019.

[5] G. Yang et al. Keep healthcare workers safe: Application of teleoperated robot in isolation ward for covid-19 prevention and control. 2020.

[6] M. H. M. Zaman. Dual-arm robot with mobile robot platform with master-slave configuration for teleoperation application. 2020.

[7] M. Schilling et al. Towards a multidimensional perspective on shared autonomy. 2016.

[8] Siddhartha S Srinivasa Anca D Dragan. A policy-blending formalism for shared control. 2013.

[9] Parham M. Kebria et al. Kinematic and dynamic modelling of ur5 manipulator. 2016.

[10] Verónica Gracia-Ibáñez et al. Across-subject calibration of an instrumented glove to measure hand movement for clinical purposes. 2016.

[11] International Federation of Robotics (IFR). Why service robots are booming worldwide – ifr forecasts sales up 12%, 2017. Retrieved from https://ifr.org/news/why-service-robots-are-booming-worldwide/ws/why-service-robots-are-booming-worldwide/.

[12] T. B. Sheridan. Human-robot interaction: Status and challenges. 2016.

[13] Róbinson Jiménez Moreno et al. Teleoperated systems: a perspective on telesurgery applications. 2012.

[14] M. Ferre et al. Advances in telerobotics. 2007.

[15] T. B. Sheridan. Telerobotics. 25(4):487–507, 1989.

[16] C. Melchiorri. Robot teleoperation. 2014.

[17] Z. H. Khan et al. Robotics utilization for healthcare digitization in global covid-19 management. 2020.

[18] Neta A. Moye Claus W. Langfred. Effects of task autonomy on performance: An extended model considering motivational, informational, and structural mechanisms. 2004.

[19] R.A. Prokopenko et al. Assessment of the accuracy of a human arm model with seven degrees of freedom. 2001.

[20] Salvador Cobos et al. Efficient human hand kinematics for manipulation tasks. 2008.

[21] DH parameters for calculations of kinematics and dynamics, Retrieved from https://www.universal-robots.com/articles/ur/application-installation/dh-parameters-for-calculations-of-kinematics-and-dynamics/.

[22] Ryan Sers et al. Validity of the perception neuron inertial motion capture system for upper body motion analysis. 2019.

[23] Universal Robots, Retrieved from https://www.universal-robots.com/it/prodotti/robot-ur5/.

[24] Perception Neuron, Retrieved from https://neuronmocap.com/software/neuron-data-reader-sdk.

[25] FZI Research Center for Information Technology, Retrieved from https://github.com/fzi-forschungszentrum-informatik/cartesian_controllers/tree/master/cartesian_motion_controller.

[26] Universal Robots, Retrieved from https://github.com/UniversalRobots/Universal_Robots_ROS_Driver.