Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2014

Filipe Daniel Lopes Serra

Interface Háptica para a Teleoperação de um Robô Humanóide
Haptic Interface for the Tele-operation of a Humanoid Robot

Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2014

Filipe Daniel Lopes Serra

Interface Háptica para a Teleoperação de um Robô Humanóide
Haptic Interface for the Tele-operation of a Humanoid Robot

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica de Filipe Miguel Teixeira Pereira da Silva, Professor Auxiliar do Departamento de Electrónica,Telecomunicações e Informática da Universidade de Aveiro e Vítor Manuel Ferreira dos Santos, Professor Associado do Departamento de Engenharia Mecânica da Universidade de Aveiro.

agradecimentos / acknowledgements

Queria deixar aqui um agradecimento a várias pessoas sem as quais não teria sido possível concluir este trajecto.

Primeiro queria agradecer à minha família, principalmente aos meus pais por todo o apoio e a ajuda ao longo destes anos, sem eles este objectivo não teria sido concluído.

Aos professores Filipe Silva e Vítor Santos por toda a ajuda, orientação, motivação e apoio dados ao longo deste ano.

Aos meus amigos por toda amizade e pela paciência demonstrada ao longo destes anos.

Ao pessoal do LAR por toda a ajuda, camaradagem e disponibilidade demonstrada em ajudar sempre que fosse preciso.

A todos um enorme Muito Obrigado.

Resumo

A aprendizagem por demonstração aplicada a um robô é uma abordagem poderosa em que um robô adquire exemplos de demonstrações humanas. Estes exemplos podem ser obtidos das mais variadas formas, tais como a gravação do estado e dos comandos enviados para o robô enquanto o mesmo é teleoperado por um operador humano ou guardar os dados enquanto um robô observa um humano executando um determinado comportamento. A teleoperação é o método mais directo de transferência de informação. No entanto, continua a faltar uma interface activa e bidireccional em que o robô e o operador possam interagir para melhorar o desempenho de uma determinada tarefa. O objectivo desta dissertação é desenvolver uma interface háptica para a teleoperação dos membros inferiores de um robô humanoíde real em tarefas de equilíbrio postural. No centro desta interface encontra-se o uso de um joystick háptico Phantom Omni que fornece ao operador humano informação sobre o estado em que se encontra o robô, as suas capacidades físicas e/ou limitações. Esta informação é transformada em realimentação de forças e irá assim ajudar o operador a guiar o robô correctamente na execução de um determinado comportamento. No fim da fase de demonstração procede-se ao registo de toda a informação sensorial disponível (*e.g.*, posições/velocidades angulares, forças de reacção no solo, dados inercias) e dos comandos numa arquitectura de software baseada em ROS (Robot Operating System). Este conjunto de dados pode vir a ter um papel importante na implementação de algoritmos de aprendizagem, permitindo ao robô reproduzir uma determinada tarefa sem ajuda externa.

Abstract

Robot learning from demonstration is a powerful approach which allows the robot to acquire training examples from human demonstrations. These demonstration examples can be obtained in different ways, such as recording state-action pairs whilst the robot is tele-operated by a human supervisor or recording the data as the robot observes the human teacher executing the desired behaviour. The tele-operation approach provides the most direct method for information transfer, but what is missing is an active and bidirectional interface in which both actors, the human teacher and the learner robot, can interact in order to improve the overall system's performance during the execution of a given task. The goal of this dissertation is the development of a haptic interface for the tele-operation of the lower limbs of a real humanoid robot during postural balance tasks. At the core of this co-adaptive design is the use of a Phantom Omni haptic device that provides the human teacher with information about the robot's state (*e.g.*, robot's balance), its physical capabilities and/or limitations. This information from the environment is translated in force feedback to allow the user to guide and correct the executed behaviour. At the end of the demonstration phase, data logging of all the sensory information available (*e.g*, joint positions/velocities, ground reaction forces, inertial data) and the control commands supplied by the human operator are carried out by a suitable software architecture based on ROS (Robot Operating System). These recorded datasets may be useful in future works for implementing a learning algorithm, enabling the robot to reproduce the task without assistance.

Contents

List of Figures

List of Tables

Acronyms

ROS	Robot Operating System
COP	Center of Pressure
USB	Universal Serial Bus
DOF	Degree of Freedom
PCB	Printed Circuit Board
SOP	Small Outline Package/Plastic
PDIP	Plastic dual Inline Package
PHUA	Humanoid Project from the University of Aveiro
ALDURO	Anthropomorphically Legged and Wheeled Duisburg Robot
LfD	Learning from Demonstration
SPI	Serial Peripheral Interface
XML	Extensible Markup Language

Chapter 1

Introduction

The world of robotics is probably one of the most engaging and challenging. Since the beginning of the mankind, the dream was to create a fully autonomous robot that could be as similar as possible to the human being. This robot would be able to help humans in difficult tasks or in the fulfillment of their needs. Despite all the work and research, this dream is far from being achieved. In particular, balance and biped locomotion remain open research problems. The locomotion of full-body humanoid robots is a complex problem due, to the high number of degrees-of-freedom (DOF) to control, the lack of precise models and the non-closed form for robot control.

From a software point of view, the main difficulty is to pre-program sophisticated motions for a broad range of tasks the robot is expected to perform. On the other side of the spectrum, teaching robots appears as a crucial demand in order to automate manual programming. In this context, robot Learning from Demonstration (LfD) is a powerful approach in which a robot acquires training examples from human demonstrations. These demonstration examples can be obtained in many different ways, each one with their own advantages and weaknesses. For example, recording state-action pairs whilst the robot is tele-operated by a human supervisor provides the most direct method of information transfer, but it is rarely used with many DOF since the movements may be difficult to control via a joystick. A comprehensive survey of LfD techniques applied for robot control can be found in [11].

This dissertation work was developed within the scope of the Humanoid Project at the University of Aveiro (PHUA), joining since 2004 the Department of Mechanical Engineering and the Department of Electronics, Telecommunications and Informatics. The main objective has been the development of highly integrated humanoid platforms based on standard components and open software for studies in locomotion, navigation, learning and perception for autonomous humanoids.

1.1 Motivation

A topic that has not yet been much addressed in the existent LfD approaches is how to exploit co-adaptation between the human teacher and the learner robot. Haptic tele-operation seems to be a promising approach for teaching and sensing in which the human operator provides functional guidance and corrections, while being aware of what the robot is doing. The motivation for the development of such bidirectional interface is to understand the role of haptic feedback to improve the process of transferring motion skills to humanoid robots, from teaching new skills to achieve optimization goals. In this line of thought, this dissertation presents a new approach for teaching motion skills to humanoid robots that relies on the physical interaction between human and robot through haptic feedback. In this sense, the proposed framework, referred to as Tele-kinesthetic Teaching [12], enters in the domain of haptic tele-operation.

1.2 Objectives

The main goal of this dissertation is to develop an haptic interface for the tele-operation of a real humanoid robot using a Phantom Omni device. This task will be dedicated to the development of a set of computational tools allowing the human operator, with the help of force feedback, to guide, correct or optimize the execution of a given task, interactively and gradually. At this stage, the haptic control involves simple postural tasks, such as a crouching motion and balance on two legs subject to external perturbations on the support surface. An interface to control the humanoid robot using a haptic device with force feedback has already been tested [9]. This interface allowed controlling the humanoid robot in simple tasks, such as lateral and vertical movements. The force feedback was only used as output variable for analyses purposes, but without any interference in the executed task.

The second objective is to gain insight on the fundamental factors that will contribute to improve the overall system's performance through the repetition of a set of experiments over time. In order to ensure an intuitive and natural interaction, it will be necessary to define the best way to map the DOFs of the haptic device to those of the robotic system. Equally important is to assure that the commands sent to the robot reflect the human intention. Another dimension of the interface design is to adopt the best strategies for translating information about the state of the robot into force feedback (the force rendering problem).

Finally, some demonstration examples obtained with the assistance of a human operator will be provided. After a training phase, where the user acquires skills in performing the demonstrations, the robot must collect and record all the sensory information (*e.g.,* joint positions, ground reaction forces) and the commands guiding the execution of a specific task.

1.3 State of the Art

This dissertation is based on two essential concepts: (1) the recording of demonstration examples (*i.e.*, state-action pairs) obtained whilst the robot is tele-operated by a human teacher, and (2) the use of haptic feedback to provide the human teacher with state feedback on the robot's performance. This section will describe some representative research works on learning from demonstration and haptic control.

Learning by Demonstration

In learning from demonstration, training robots and obtaining the teaching examples can be made in many different ways, namely kinaesthetic teaching, tele-operation and observational learning.

In the kinaesthetic teaching method, also known as tactile teaching, the human operator physically guides the robot through the desired task [2]. In [1], an approach using kinaesthetic teaching is presented where the operator controls the humanoid robot by moving directly its arm. This experiment is repeated several times during the demonstration phase and, using the recorded data, the humanoid robot is able to perceive the desired task and reproduce it without any external help. This method has the advantage of avoiding the constraints created by the correspondence map between the controller and the robot. Figure 1.1 shows an example of kinaesthetic teaching.

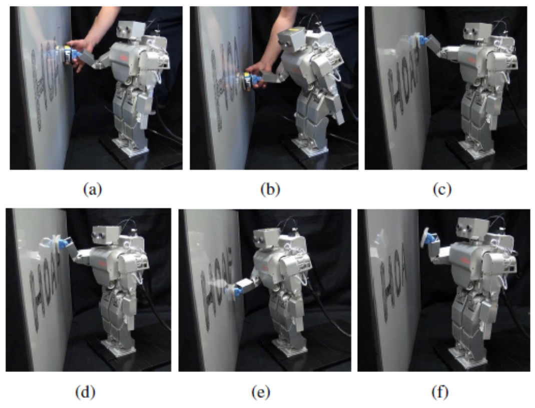


Figure 1.1: Learning by Kinaesthetic example [1].

Tele-operation is a learning method where the human operator controls the robot using an input device (*e.g.,a joystick or a haptic device*. An example of this technique is found in [2]. In this research, the operator uses the haptic device to control the movement of the robotic arm and the buttons to open or close the gripper of the robot. In the learning phase, the operator uses the joystick to perform the intend tasks and collect data to teach the robot in the following phase. Similarly to the kinaesthetic teaching this type of demonstration has the problem of mapping between the joystick and robot workspaces. In cases of high DOF's, this can be a serious and difficult problem to solve. In Figure 1.2 an example o tele-operation teaching is shown.

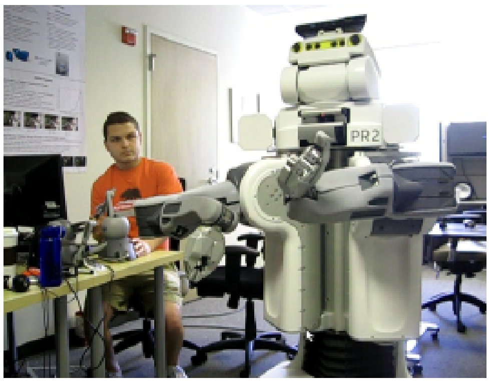


Figure 1.2: Teleoperating Example[2].

In observational learning, also known as imitating teaching, the human exemplifies a certain task by showing the various steps required to perform it [13]. The robot will imitate the human operator and perform the task by itself. This demonstration method allows the robot to generalize and apply the learned movement for different tasks. In Figure 1.3 an example of a demonstration using the observation method is shown.

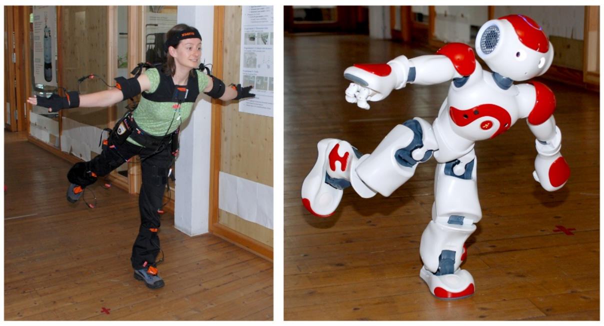


Figure 1.3: Observing Example [3].

Haptic Control

Although the use of haptic devices is recent, they begin to play a crucial role in many research areas, such as robot tele-operation (*e.g.,* in rescue missions, handling dangerous materials, etc), medicine (*e.g.,* robotic surgery, medical devices, etc) and virtual reality.

Medicine is probably the area where the use of haptic devices is further advanced. There are already several prototypes that perform surgeries using this type of devices [14][4], but only recently force feedback was introduced. In [15], the system consists of a master-slave architecture where the movement done by the robot (slave) is controlled by an operator via a surgeon console (master). This technique allows to perform complex operations. The use of force feedback technology can help and improve the doctor's performance in a surgery. This technique brought unimaginable progress in this field, since the doctors can act according to what they feel (the force feedback is generated by the impact and proximity of the robot's end-effector to the tissue). Thus, the success of the surgery can be improved and tissue damage can be prevented. It is now possible to have the best of both worlds: the precision provided by the use of an arm/robotic operator to do the operation, and the control of the operation done by an expert doctor feeling like he was actually in contact with the patient's body. An example of haptic surgery can be seen in Figure1.4.

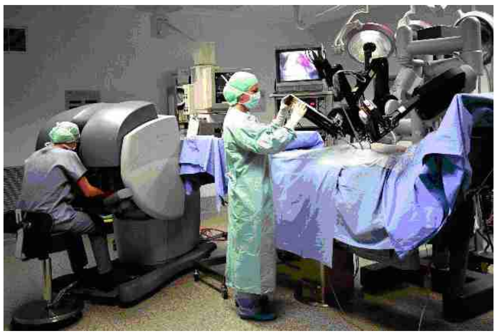


Figure 1.4: Haptic surgery example[4].

A different kind of haptic joystick is presented in [16], showing the importance of force feedback. Using a joystick to control a robot shows importance in imminent danger or for example, to do complex and difficult situation/operations. In this case, we are in the presence of a DataGlove with sensors that retrieve information to control the robot. In the robotic arm, there are a set of sensors which, when pressed, transmit information to the glove wore by the operator. This information is then converted into force and applied in the operator. Because of that it is possible for the human operator to control robotic arm and hand, "feeling" all the movement, including what the robot grabs. A better performance can be obtained and the operator does not need to be in the field to touch or grab equipment. Figure 1.5 shows an example of the interface used.

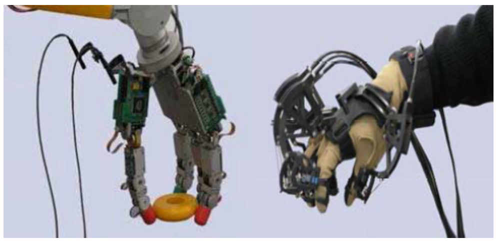


Figure 1.5: DataGlove.

The objective of work in [5] is to control a four-legged robot using a joystick with force feedback. The robot ALDURO consists of a platform with a cabin for the operator, a power plant and four legs. With the joystick, the robot can be controlled by two different ways: position control and speed control. The force feedback is used to achieve two purposes:

- To help the operator to position the joystick
- To provide the workspace information, *i.e*, when the robot is close to the limit one counter force is generated.

The use of force feedback protects the robot of moving beyond its limits and help to control it [5]. The ALDURO robot can be seen in Figure 1.6.

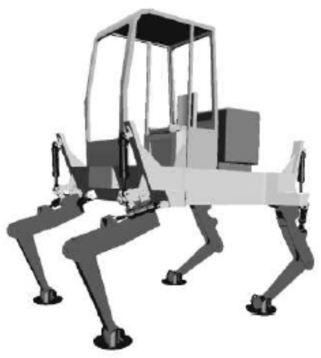


Figure 1.6: ALDURO Robot [5].

1.4 Structure of the document

The dissertation is organized as follows: Chapter 2 introduces the hardware and software technologies used to implement the haptic tele-operation system. Chapter 3 presents the data processing algorithms used to transform human's commands into robot's actions and the strategies for translating information about the state of the robot into force feedback. Chapter 4 describes the software architecture of the haptic interface, namely the different ROS-based processes and how they interact with each other. Chapter 5 presents the experiments conducted to demonstrate the validity of the haptic feedback for teaching balance skills in humanoid robots. Finally, chapter 6 discusses the conclusions reached in this dissertation and proposes suggestions for future work.

Chapter 2

Infrastructure and Experimental Setup

This chapter describes the hardware and software technologies used to develop the haptic tele-operation system. The main hardware components consist of a real humanoid robot, the Phantom Omni haptic device and the Load Cells sensors. The haptic interface consists of several modules running in the host computer and the software architecture is based on the ROS (Robot Operating System), Fuerte version, middleware using a Linux/C/C++ programming environment.

2.1 Hardware Technologies

2.1.1 The PHUA Humanoid Robot

Started in 2004, the PHUA project culminated, in 2009, with the development of a highly integrated humanoid platform oriented for research in biped locomotion, navigation in real world environments and multi-modal perception for autonomous humanoids. This robot consists of a small-sized whole-body robotic platform based on standard components and open software. From a hardware point of view, this research platform has a great potential in terms of movement abilities with 27 degrees-of-freedom (DOF), low-level actuator control for both position and velocity, and a number of useful onboard sensors, such as video cameras, inertial sensors and load cells on both feet. A hybrid actuation system allows combining active HITEC servomotors with passive elastic components. The communications with the humanoid robot, both for control actions and for sensory readings, is performed through an external computer using a bidirectional RS-232 protocol. The humanoid platform is represented in Figure 2.1 together with the distribution of the 27-DOFs by the robotic structure. Table 2.1 summarizes the servomotor models installed at each joint of the robotic structure.

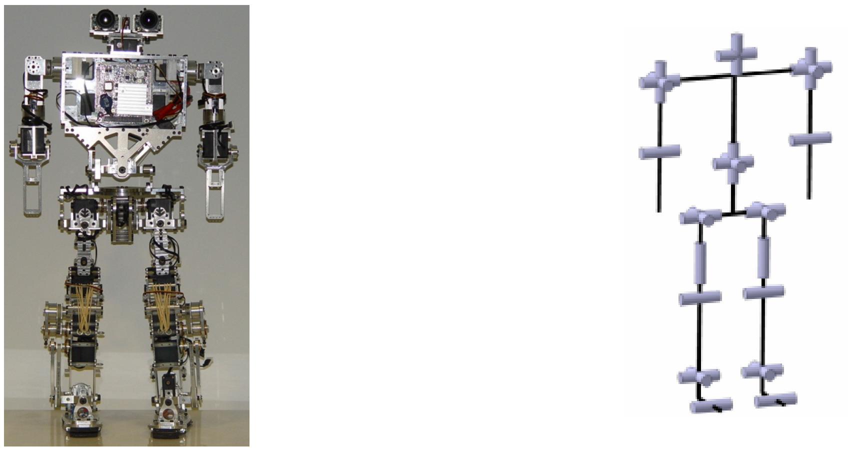


Figure 2.1: Physical humanoid robot (left) and distribution of DOF by the full-body structure (right) [6].

Joint	Number of DOFs	Servomotor
Toe	1(x2)	(none)
Ankle	2(x2)	HSR-5980SG
Knee	1(x2)	HSr-5980SG
Hip	3(x2)	HSR-5980SG

Trunk	3	HSR-5980SG(x2)
		HSR-8498SG(x1)

Shoulder	3(x2)	HSR-5980SG HSR-5498SG HS-82MG
Elbow	1(x2)	HSR-5498SG
Neck	2	HSR-5498SG HS-5056MG
Total	27	--

Table 2.1: PHUA platform's degrees-of-freedom and installed servomotors [6].

2.1.2 Load Cells

The humanoid robot is equipped with four load cells placed in the corners of the feet (see Figure 2.2 (Left Side)). The load cells (see Figure 2.2 (Right Side)) are force transducers that measure the reaction between two plates, in this case, both feet and the ground. The cells used in this work have a range of 5 and 10 pounds of force.

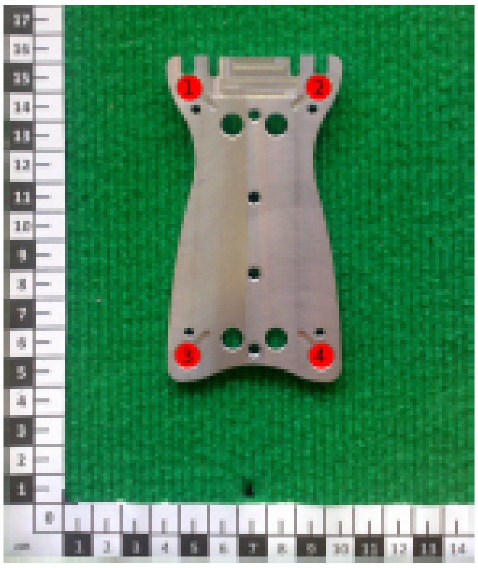
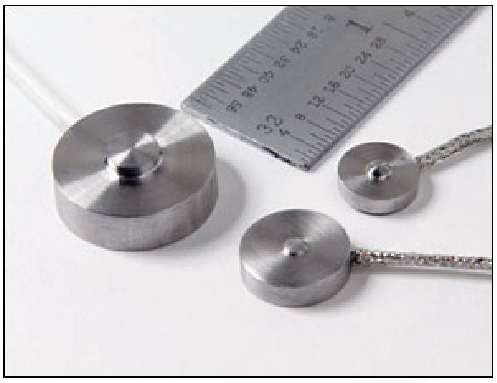



Figure 2.2: Bottom of the foot [7] (Left Side), Load Cells (Right Side)

To correctly read the values of the load cells some signal conditioning is needed. That work was done before using PDIP (Plastic Dual Inline Package) components and a Arduino Fio in [9]. The main problem is that the circuit board can not be accommodated within the humanoid robot because of its size. Another problem is the fact that the Arduino Fio needs an USB Serial Adapter to work. It is not possible to communicate and power the board using just a USB interface. Therefore, a new PCB was designed and built to solve this problem. All components used in [9] were replaced by SOP components and the Arduino used is a Arduino Nano (see Figure 2.3 and Table 2.2).

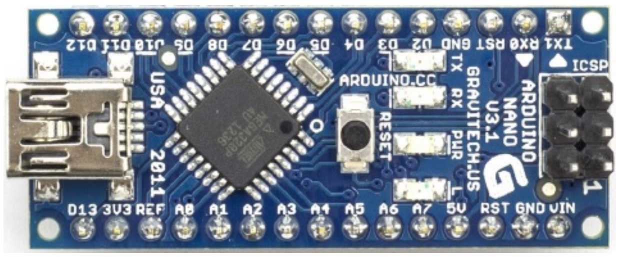
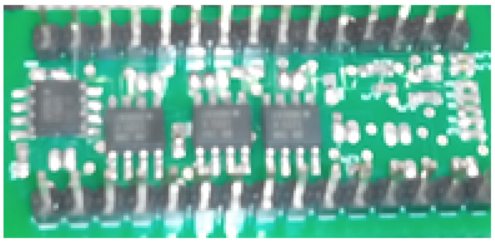



Figure 2.3: Arduino Nano (Left Side), New breadboard (Right Side).

Microcontroller	ATmega328
Input Voltage	7-12 V
Digital I/O Pins	14
Analog Input Pins	8
DC Current per I/O Pin	40 mA
Flash Memory	32 KB (ATmega328)
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz
Dimensions	0.73" x 1.70"

Table 2.2: Arduino Nano Characteristics.

The Arduino Nano is a small breadboard based on the ATmega 328. This Arduino can be powered externally (6 - 20 v, pin 30) or using a mini-USB connection (pin 27). Contains 14 digital pins that can be used as input or output, and operates at 5 volts. Pins 0 and 1 are used for receiving and transmitting serial data. External interrupts are allowed in the digital pins 2 and 3 and PWM in the pins 3 to 11. The remaining pins are used to SPI (Serial Peripheral Interface). The Arduino Nano has 8 analogue pins, providing 10 bit of resolution and able to read from ground to 5 volts.

The new board is smaller: the new size is 4.37 cm x 2,195 cm instead of 6.051 cm x 2.712 cm and the route and clearance are now 0.2032 mm (8 mil) and 0.1524 mm (6 mil) instead of 0.25 mm (10 mil) and 0.2032 mm (8 mil).

The board's new size is beneficial to pack the boards with the Arduino Nano in the humanoid robot. This will permit a better control of space and will decrease the number of outside cables.

2.1.3 Haptic Device

The idea behind an haptic device is to provide the users the force feedback information that represents a "feeling of the situation". The haptic device will be useful for tele-operation tasks where visual information may not be sufficient. In this work, it is used the Phantom Omni haptic device manufactured by SensAble Technology. This is a pen-style haptic device based on serial mechanisms (see Figure 2.4) providing 6-DOF motion (3-DOF translational motion and 3-DOF orientational motion) and capable of rendering three dimensional forces(*i.e.*,, 3-DOF force feedback device). The communication with the host computer is achieved through an IEEE 1394a FireWire port. Table 2.3 summarizes the most important technical specifications of the haptic device Phantom Omni.

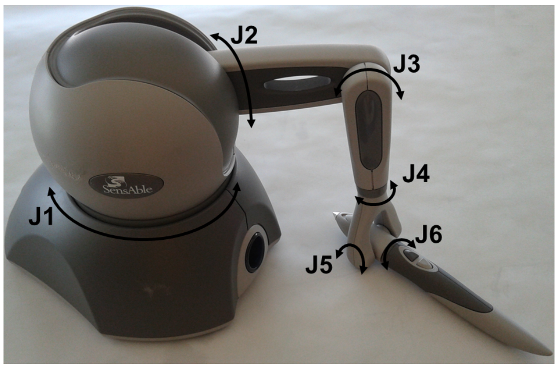


Figure 2.4: Phantom Omni DOF's.

Model	**The PHANTOM Omni Device**
Force feedback workspace	6.4 x 4.8 H x 2.8 D in >160 W x 120 H x 70 D mm
Footprint **Physical area the base of device occupies on the desk**	6 5/8 " x 8 D in 168 W x 203 D mm
Weight (device only)	3 lb 15 oz
Range of motion	Hand movement pivoting at wrist
Nominal position resolution	>450 dpi 0.055 mm
Backdrive friction	<1 oz (0.26 N)
Maximum exertable force at nominal position	0.75 lbf. (3.3 N)
Continuous exertable force	>0.2 lbf. (0.88 N)
Stiffness	X axis >7.3 lb/in (1.26 N/mm) Y axis >13.4 lb/in (2.31 N/mm) Z axis >5.9 lb/in (1.02 N/mm)
Inertia (apparent mass at tip)	0.101 lbm. (45 g)
Force feedback	x, y, z
Position sensing ———————- **[Stylus gimbal]**	x, y, z(digital encoders) —————— [Pitch, roll, yaw (5% linearity potentiometers)]
Interface	IEEE-1394 FireWire® port: 6-pin to 6-pin
Supported platforms	Intel or AMD-based PCs
OpenHaptics SDK compatibility	Yes

Table 2.3: Phantom Omni by Sensable™ characteristics [10].

2.2 Software Tools

2.2.1 Robot Operating System (ROS)

This project is developed under the Robot Operating System (ROS) software that helps to create robot applications by providing a wide range of libraries and computational tools. It is open-source software that provides functionalities such as hardware abstraction, device drivers, graphic tools, debugging tools, libraries of software commonly used and message passing between processes. The ROS software allows to split the problem in different parts. This point is important to the Humanoid project because it allows to divide each part of the robotic control in separated modules (or packages) that communicate with each other but can work independently. To do so, each module is programmed recurring to C++ or python and some executable/nodes are created.

ROS is distributed under BSD license (permissive free software license) and has the advantage of having a large on-line community, a very well documented code and solutions to help developers to create their code faster. This makes ROS a very flexible platform, capable of being implemented in any robot [17]. A brief description about the different elements that compose the ROS environment is given below.

Nodes

In ROS, a node is a process which performs computation. Each one has a specific function, for example, a node may control the haptic joystick, another may control the ServoMotors and so on. A node can have subscribers and publishers and communicate with another using topics, services, and parameter server. Therefore, the use of nodes brings two important benefits: the ability to isolate potential problems and reduce the complexity [18].

Topic

Each node exchanges messages through a communication channel/bus that, in ROS, is known as a topic [19]. On the one hand, whenever a node is ready to receive some data subscribes a specific topic. On the other hand, whenever a node generates new data, it publish it on a certain topic. Each topic can have multiple publishers and subscribers. Using a tool called *rostopic* is possible to view specific information about a topic. In the list below, examples of the use of this tool are shown.

- *rostopic bw* Display bandwidth used by topic list
- *rostopic echo* Print messages to screen
- *rostopic find* Find topics
- *rostopic hz* Display publishing rate of topic

- *rostopic info* Print information about active topics
- *rostopic list* Show active topics
- *rostopic pub* Publish data to the topic
- *rostopic type* Print topic type

Messages

A message is a data structure with different fields that supports standard and array primitive types (*e.g.,* integer, floating point, boolean, etc). The class that implements the message is automatically generated from a file with the .msg extension. Each message consists of a header with a time stamp and a frame ID. In the following list an example of a .msg file is shown.

- *Header header*
- *int16 speed_g_static*
- *int16[13] speed_wanted*
- *string data*
- *float64[13] joint_now*

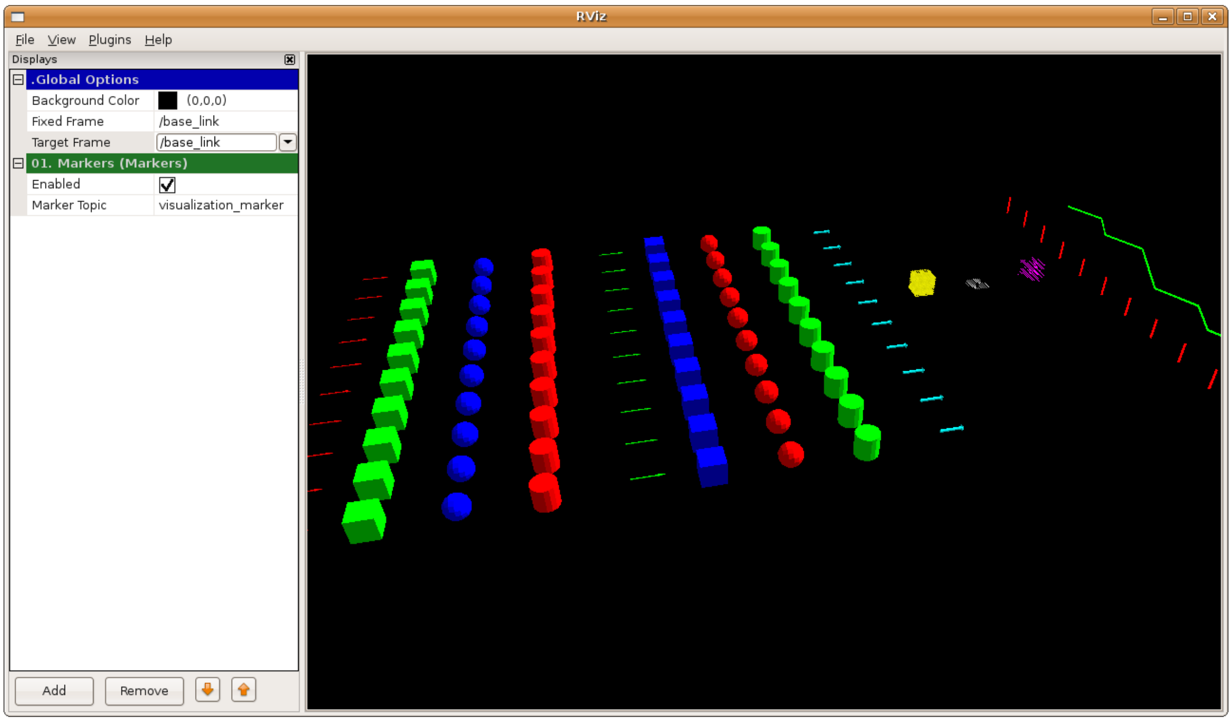
Launch file

An important part of ROS are the launch files. Instead of launching each node individually, an XML file with a .launch extension can be created to launch all the required nodes. An example of a .xml file is shown below :

```
     <?xml version="1.0"?>
<launch>

    <!-- This is a launches Humanoid node-->
    <node name="humanoid_node" pkg="humanoid_control" type="humanoide">
<!--
        <remap from="/device" to ="/device_mapper/serial_com1"/>
-->

        <param name="op_mode" value="6"/>
        <param name="port" value="/dev/ttyUSB0"/>

    </node>
</launch>
```

Rviz

Rviz is one of the most used tools for debugging in ROS. Rviz is a graphic tool to see what is happening in a ROS system. It is possible to subscribe topics and see data inside them in a graphical way. In Figure 2.5, an example of this tool [20] can be seen.



.Global Options
Background Color (0,0,0)
Fixed Frame /base_link
Target Frame /base_link
01. Markers (Markers)
Enabled ✓
Marker Topic visualization_marker

Figure 2.5: Rviz example.

Bag

An important tool when working with the ROS environment is the rosbag [21]. Rosbag is a tool that can subscribe one or more ROS topics and record the data of each subscribed ROS topic. Besides recording, the rosbag tool also plays the data, making it possible to test new algorithms even without the robot. The following list shows examples of how to use this tool.

- *rosbag record* Record a bag file with the contents of the specified topics
- *rosbag info* Display a summary of the contents of the bag files.
- *rosbag play* Play back the contents of the given bags
- *rosbag check* Determine if the a bag is playable in the current system
- *rosbag fix* Repairs a bag using registered rules

- *rosbag filter* Convert a bag file using the given Python expression
- *rosbag compress* Compress the bag files
- *rosbag decompress* Decompress the bag files
- *rosbag reindex* Reindex the bag files

2.2.2 OpenHaptics Toolkit

The OpenHaptics Toolkit is a programming solution to build haptical applications developed in C/C++ by SensAble™. This toolkit is available for different operating systems and includes:

- QuickHaptics micro API, allows to quickly and easily write applications or adds haptics to existent applications. The fact that has a built-in geometry parsers and intelligent parameters makes possible to create haptic scenes with small code.
- Haptic Device API (HDAPI), provides low-level access to the haptic device and enables haptic programmers to render forces directly.
- Haptic Library (HLAPI), provides high-level haptic rendering and is designed to be familiar to the OpenGL API programmers, simplifying haptics and graphics threads.
- Device Drivers (PDD), support all shipping Phantom devices.
- Source Code Examples, several examples to illustrate commonly used functionalities of Open Haptics.
- Programmer's Guide, explains the OpenHaptics toolkit and introduces the architecture of the toolkit.
- API Reference, contains references pages to all the QuickHaptics, HDAPI and HLAPI functions.

The typical application structure under the OpenHaptics is illustrated in Figure 2.7. The relation between the QuickHaptics micro API and the existing HD and HL layers is shown in Figure 2.6.

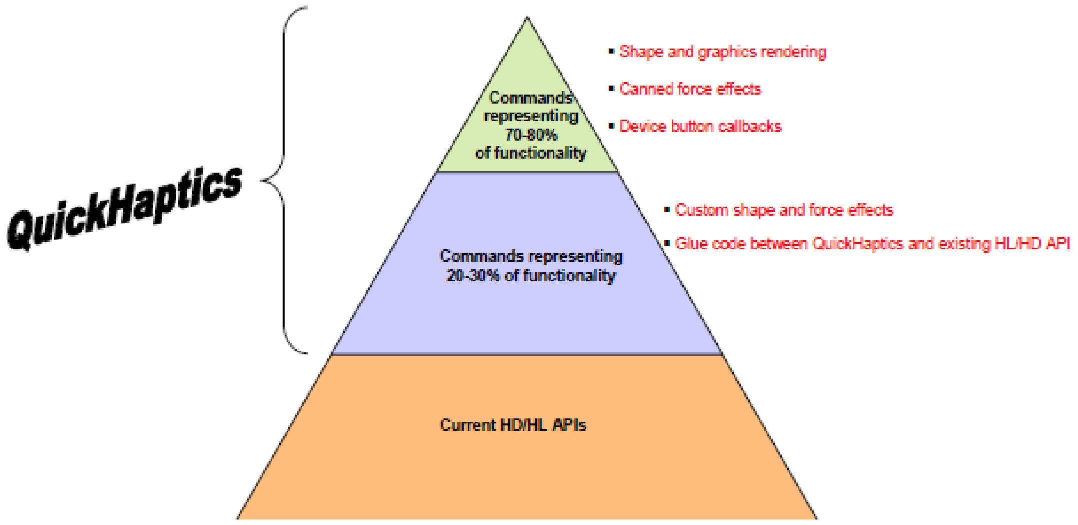

QuickHaptics
Commands representing 70-80% of functionality
Commands representing 20-30% of functionality
Current HD/HL APIs
▪ Shape and graphics rendering
▪ Canned force effects
▪ Device button callbacks
▪ Custom shape and force effects
▪ Glue code between QuickHaptics and existing HL/HD API

Figure 2.6: OpenHaptics overview [8].

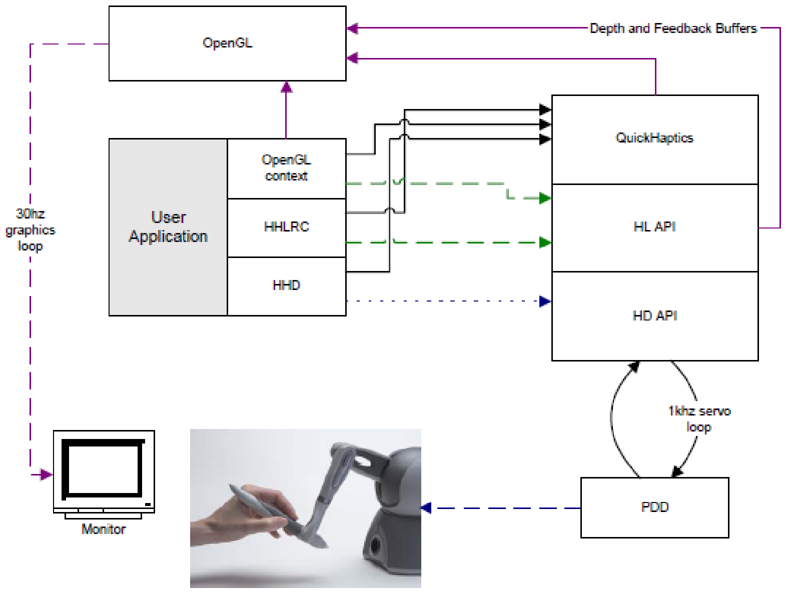


Figure 2.7: Relationship between QuickHaptics, HD and HL layers of OpenHaptics [8].

Chapter 3

The Haptic Tele-operation Paradigm

This chapter describes the haptic tele-operation approach for teaching balance skills to humanoid robots through force feedback. The intention is to include the human teacher in-the-loop by providing him/her with state feedback on the robot's performance. Based on the evaluation of performance, the human teacher may provide functional guidance and corrections on the robot's behaviour. In this context, haptics tele-operation refers to an active and bidirectional interface between the human teacher and the robot learner. The bilateral nature of the implemented interface enables the human teacher to provide motion commands to the robot and, simultaneously, to receive force feedback about the robot's current state.

Figure 3.1 illustrates the control architecture of the haptic tele-operation system and the interconnection among the main hardware components. One of the main computational components of the interface is responsible for the kinematic mapping between the DOFs of the haptic device to those of the humanoid robot. The second important component is related with the strategy for translating information about the robot's state into force feedback, usually referred as the force rendering problem. It becomes challenging to adopt the best strategies for translating information about the state of the robot into force feedback. Here, the goal is to provide the human teacher with the right perception of what is happening to the robot in terms of balance state.

Another dimension of the problem is the need for a training phase. The human teacher needs training with respect to what force rendering algorithms are most effective, what sensorial information accurately reflects the real-word dynamics and how the robot interprets the human intention. Force feedback from the environment, performance metrics to help the human teacher understand the level of improvement achieved and the repetition over time are fundamental factors that will contribute to simplify the learning process and/or to improve the overall system's performance. Despite the importance of a training phase with multiple users, the study of the interface usability is missing since it is beyond the scope of this work.

In summary, this chapter is divided into two distinct parts. The first part presents the data processing algorithms implemented to transform commands from the human teacher into control actions applied to the robot. The second part is dedicated to analyze the data obtained from the force sensors located in the feet of the humanoid robot and how they are transformed into force feedback.

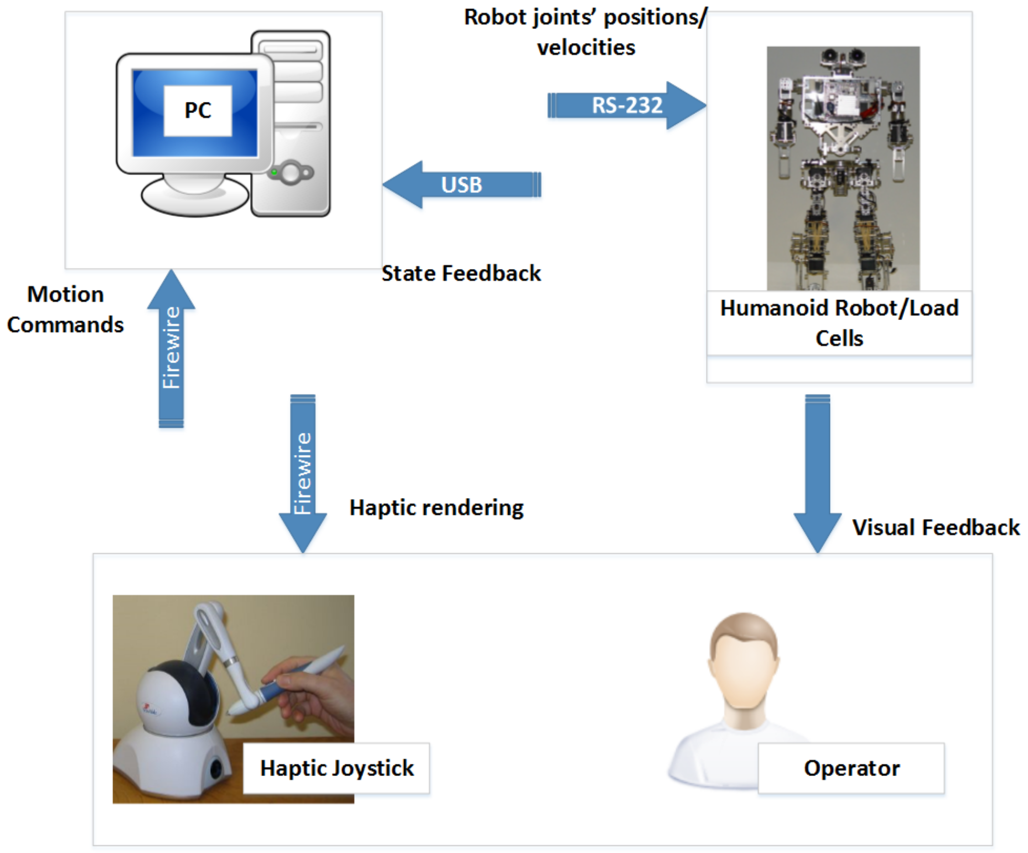


Figure 3.1: The haptic loop of the tele-operation system. The haptic device senses the human operator input, such as position and orientation, and the system applies the input to the tele-operated humanoid robot. The response of the humanoid robot is computed through haptic rendering and relayed to the human operator. The ideal result is that the human operator feels that he/she is interacting directly with the humanoid robot. Depending on the haptic interface and application, this loop must be repeated a few hundreds of Hertz or higher frequencies around 1 kHz.

3.1 Pre-processing of joystick raw-data

3.1.1 Haptic Workspace and Force Limits

The haptic device used in this work is a SensAble PHANToM Omni (more recently Geomagic Touch) joystick popularized by haptic applications in virtual reality [22]. The Omni is a pen-style haptic device, with sensors and actuators, based on a 6-DOF serial mechanism, providing information about the position (the first 3-DOF) and the orientation (the last 3-DOF) of its end-effector. In addition to sense the stylus motion in six DOF, this haptic device can apply forces in the x, y and z directions to the stylus tip (Figure 3.2 shows the reference coordinate system). In the bilateral interface, motion becomes the output from the user and force becomes the input to the user (*i.e.*, the teleoperation system provides information to and requires commands from the user).

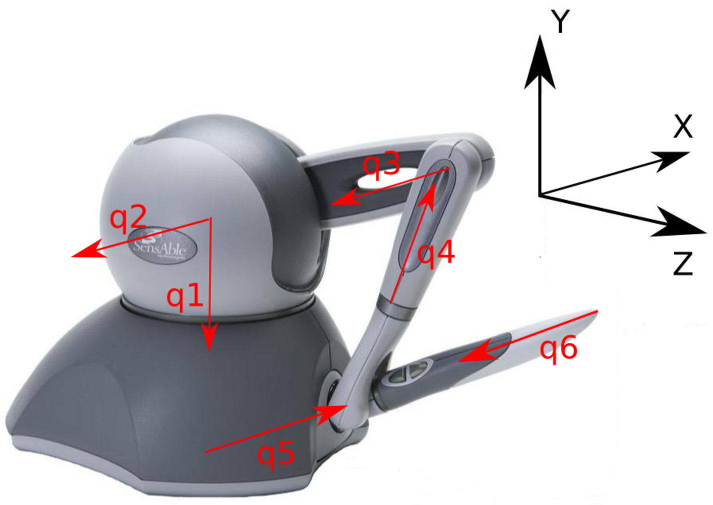


Figure 3.2: Phantom Omni device with the Cartesian axis

Given the physical limits of the rotational joints and the link lengths of this haptic device, the workspace available for the operator is also limited. Tables 3.1 and 3.2 depict the limits in the translational and rotation motions, respectively. The limits can also be found using the function *hdgetDoublev* from the Open Haptics library [23].

	$x(mm)$	$y(mm)$	$z(mm)$
$Maximum$	210	205	130
$Minimum$	−210	−110	−85

Table 3.1: Maximum workspace dimensions

	$J4(mm)$	$J5(mm)$	$J6(mm)$
$Maximum$	$-0,765$	$-1,864$	$-0,562$
$Minimum$	$-5,534$	$-4,186$	$-5,723$

Table 3.2: Maximum Orientation workspace

Besides the 6-DOF input, the Omni device provides 3D force feedback in position. To accurately render the robot's state, the haptic device must be able to render forces over a large dynamic range, in both frequency and magnitude. The maximum and the continuous force allowed for the Omni device are presented in Table 3.3 (values available by the same function *hdgetDoublev*). Although the device allows the application of forces of 3.3 N, it can only support a continuous exertable force (24 hrs.) of 0.88 N (a permanent damage may occur if the value is exceeded).

	$Maximum(N)$	$Continuous(N)(24hrs)$
$Force$	3.3	0.86

Table 3.3: Maximum and Continuous force permitted by the device

In addition to the large difference in values, another dimension of the problem is the lack of uniformity throughout the target workspace. As a result, the usable workspace for the haptic device with force feedback is reduced to the values shown in Table 3.4

	$x(mm)$	$y(mm)$	$z(mm)$
$Maximum$	80	60	35
$Minimum$	-80	-60	-35

Table 3.4: Usable workspace of the device

3.1.2 Data Filtering

Using a haptic joystick as the input device, there are two different modes for describing the desired robot motion: position control and velocity control. Position control mode means that the joystick displacement is interpreted as a desired position. This is typical for controlling stationary robots with a limited workspace. In velocity mode, the joystick data is interpreted as a desired velocity imposed to the robot. Since the intention is to control a stationary robot with a limited workspace, position control makes more sense. Thus, the user operation on the joystick is transmitted to the humanoid robot in the form of a position command.

However, the joystick raw-data may be unsuited if directly transferred into control actions to be applied to the humanoid robot. Therefore, user control in position must be filtered to smooth the actions sent to the robot controller. For the experiments carried out, two types of data filtering were implemented. First, low-pass filtering was used to remove human's hand vibrations or instability when holding the joystick. Figure 3.3 illustrates the result of applying a moving average filter on the data acquired with the joystick and after processed in Matlab.

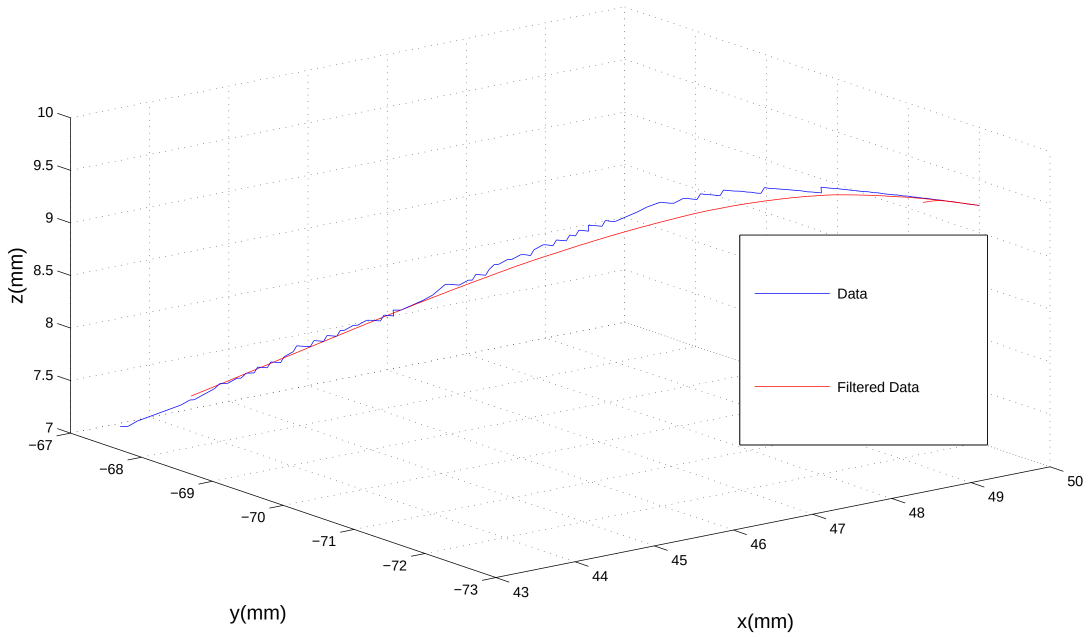
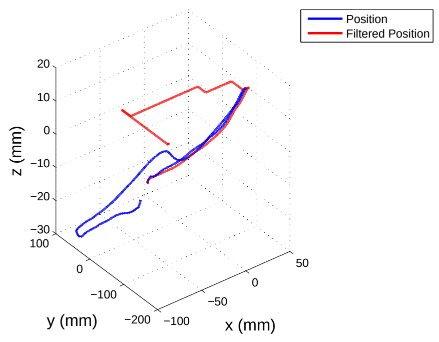


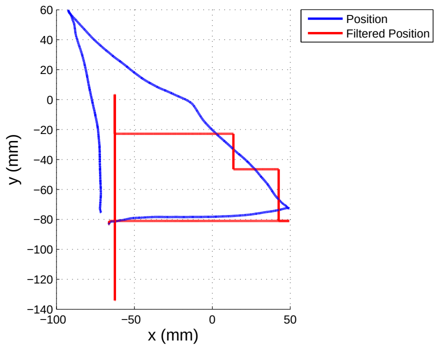
Figure 3.3: Original raw-data vs. filtered-data.

Second, path filtering was used to impose specific paths such as up-down, front-back or left-right. This apparent constrain reflects, in general, more precisely the human intention in teleoperation applications, especially when a hybrid solution is considered (*i.e.,* the filter is only applied to certain components of the movement, but not others). A second advantage can be envisioned during operation whenever the user is exposed to abrupt movements of the humanoid platform. Whenever these perturbations are passed to the driver's hand via force feedback, they could cause unintentional control pulses that can be mitigated by filtering the joystick raw-data.

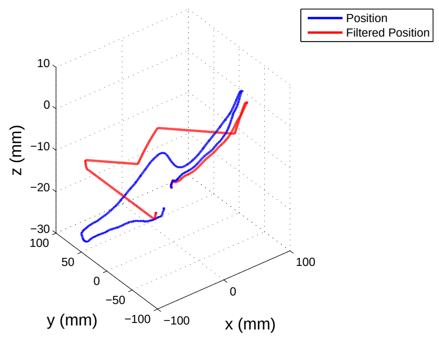
In this work, this second part of the data processing consists of applying an arctangent function to the data in each individual plane (*i.e.*, x-y, x-z and y-z planes). Once the quadrant is known, the movement is modified according to an external parameter provided a priori by the user. For example, accepting from the user only limited slope resolutions in multiples of 90°. In this particular case, the final movement is restricted to occur along the x-, y- and z-axis. Figure 3.4 shows the result of several tests performed in 3D space with different resolutions of slopes in the trajectories. It should be noted that for resolution angles lower than $\frac{\pi}{6}$ the difference between the original raw-data and the filtered-data is negligible.



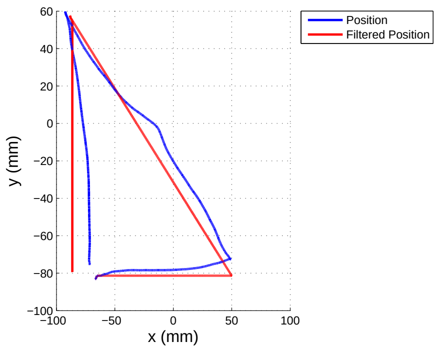
(a) 3D Plan for a parameter of $\frac{PI}{2}(90°)$



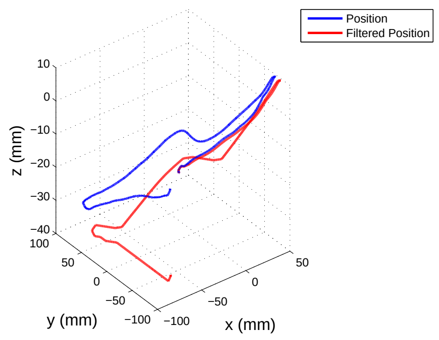
(b) XY Plan for a parameter of $\frac{PI}{2}(90°)$



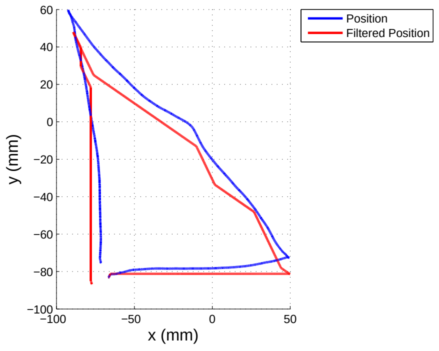
(c) 3D Plan for a parameter of $\frac{PI}{4}(45°)$



(d) XY Plan for a parameter of $\frac{PI}{4}(45°)$



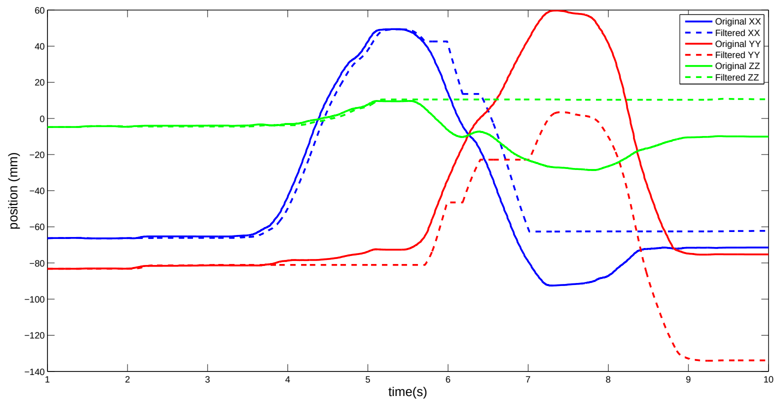
(e) 3D Plan for a parameter of $\frac{PI}{6}(30°)$



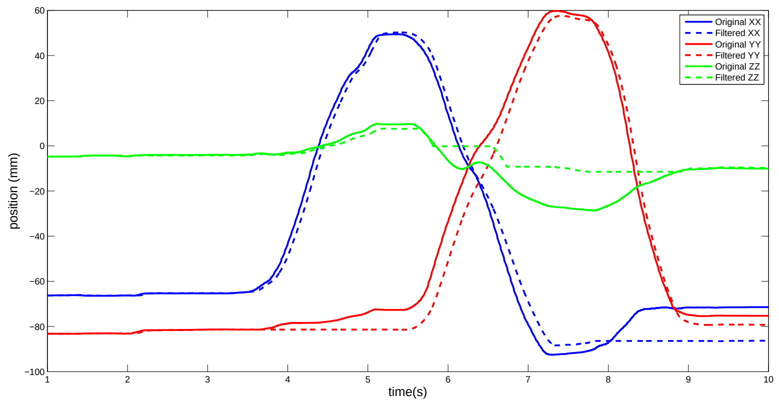
(f) XY Plan for a parameter of $\frac{PI}{6}(30°)$

Figure 3.4: Original Data Vs Filtered Data , XY and 3D plan

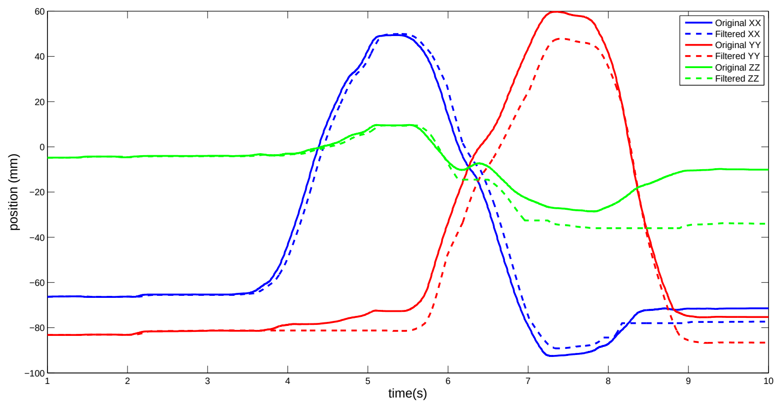
Another perspective of the same problem is depicted in Figure 3.5 where the time courses of the three coordinates are represented. The filtered-data (dotted curves) are now limited to principal directions, being visible that they are affected by time delay.



(a) Coordinate variation over time for a parameter of $\frac{PI}{2}(90°)$



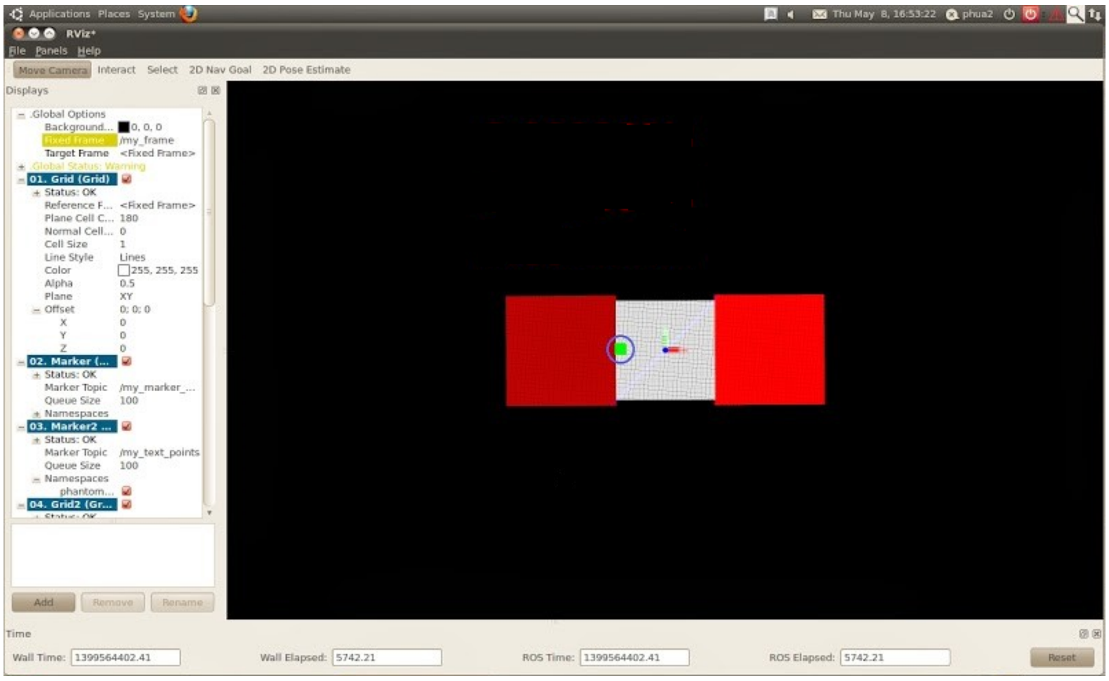
(b) Coordinate variation over time for a parameter of $\frac{PI}{4}(45°)$.



(c) Coordinate variation over time for a parameter of $\frac{PI}{6}(30°)$.

Figure 3.5: Original Data Vs Filtered Data over time.

After the MATLAB experiments, the next step is to convert and test all the code in ROS/C++ environment. A Rviz application was developed to test all the code before using it in ROS/C++ environment. The Rviz tool has the ability to create markers (see Chapter 2 section Robot Operating System). In this case, the marker chosen was a Cube whose position is directly linked to the information coming from the Phantom Omni joystick (see Figure 3.6). It was possible to understand that the movement was properly modified by changing the joystick's position and analyzing the outcome in Rviz.


Applications Places System
RViz*
File Panels Help
Move Camera Interact Select 2D Nav Goal 2D Pose Estimate
Displays
Add Remove Rename
Time
Reset

```
− .Global Options
    Background...   0, 0, 0
    Fixed Frame     /my_frame
    Target Frame    <Fixed Frame>
+ .Global Status: Warning
− 01. Grid (Grid)   ✓
  + Status: OK
    Reference F...  <Fixed Frame>
    Plane Cell C... 180
    Normal Cell...  0
    Cell Size       1
    Line Style      Lines
    Color           255, 255, 255
    Alpha           0.5
    Plane           XY
  − Offset          0; 0; 0
        X           0
        Y           0
        Z           0
− 02. Marker (...   ✓
  + Status: OK
    Marker Topic    /my_marker_...
    Queue Size      100
  + Namespaces
− 03. Marker2 ...   ✓
  + Status: OK
    Marker Topic    /my_text_points
    Queue Size      100
  − Namespaces
        phantom...  ✓
− 04. Grid2 (Gr...  ✓

Wall Time: 1399564402.41   Wall Elapsed: 5742.21   ROS Time: 1399564402.41   ROS Elapsed: 5742.21
```

Figure 3.6: Example of the used Rviz interface.

This data filtering is mostly concerned with position data, but is also used with the orientation as well. The result of applying a moving average filter to the raw-data from the last three DOF (orientation of the tip) is shown in Figures 3.7 to 3.9. The orientation was tested using the Rviz application. After testing the different rotations of the cube, it has been proven that the use of all the 3 DOFs at the same time is difficult and can lead to continued instability. Therefore, for further work only one DOF can be active at the same time. The active one is determined by calculating the Standard deviation of 200 orientation points, the degree in the highest value is considered the dominant.

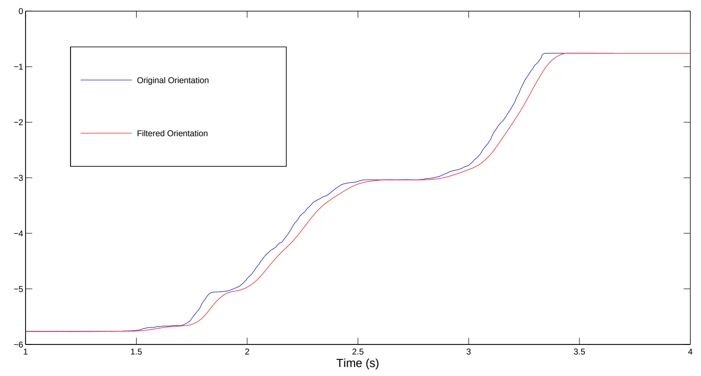


Figure 3.7: Joint 4 original vs filtered data.

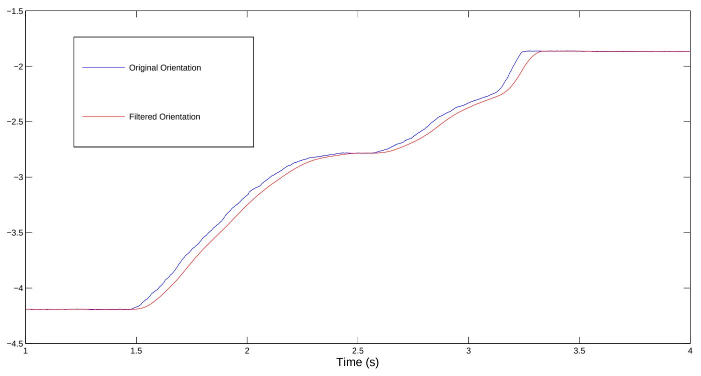


Figure 3.8: Joint 5 original vs filtered data.

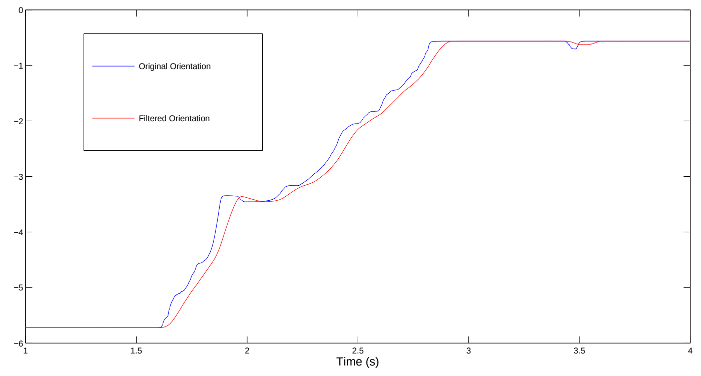


Figure 3.9: Joint 6 original vs filtered data.

3.2 Kinematic Mapping between Haptic Device and Humanoid Robot

Assuming the humanoid robot is controlled from the filtered data arriving from the haptic device, this section considers the mapping from the haptic device to the humanoid robot (two robots). In order to provide a natural and intuitive human-robot interaction, it is necessary to define the best way to map the DOFs of the haptic device to those of the robotic system. In this context, it is important to remember that the haptic device moves in its own workspace, while the robot moves in another workspace somewhat different. Before discussing how the haptic and the robot are connected, it must be highlighted that the two devices are not always coupled, *e.g.,* before the system is turned on.

Typically, the humanoid robot starts from an initial standing position (upright). When the system is turned on, the haptic's tip is moved to the zero position and the connection is established by pressing a keyboard key in the host computer. One of the joystick's buttons can be pressed to define a new "zero" position allowing the user to rest without affecting the humanoid robot behaviour. In that case, the beginning of the movement requires that the stylus tip be placed in the last configuration to avoid a shift between the two robots (*i.e.,* to assure that the two workspaces perfectly overlap). This means the haptic mechanism and the humanoid robot are constrained to start at the same location and the system does not allow offsets between them.

In order to define the connection between the two robots, it is worth noting that the haptic device and the humanoid robot are kinematically dissimilar, both from the point of view of number of DOF and link lengths. As result, connecting them at a joint level does not seem to be appropriate or even feasible; instead, it was decided to connect them at their tips (Cartesian tip positions) in a position-position mode. Then, the hip position can be defined by:

$$\tau_{hip} = \mu.\tau_{haptics} \tag{3.1}$$

where μ represents the motion scale, $\tau haptic$ is the haptic position measured relative to the haptic device (see Figure 3.2) and τhip is the hip position measured relative to the robot view (discussed in Chapter 5). In this work, the scale was set to map the two workspaces as best as possible. However, it is expected that nonlinear or time-varying mappings may be useful to best utilize the humanoid robot workspace.

In the implemented mode, the human operator controls the robot's hip position what implies that an inverse kinematics algorithm should be used for translating the haptic commands into robot actions (joint angular displacements). The kinematics equations describing the relation between the Cartesian and the joint spaces of the lower limbs are presented in [9]. The angle values obtained by the kinematics algorithm need to be transformed into a range of values between 600 and 2400 before being sent to the HITEC servomotors. Each individual servomotor includes a local internal PID position controller that assures a position error around 1°. The communications between the host computer and the servomotors are performed via a RS-232 bidirectional serial interface. The RS-232 communication parameters and the structure of the data frame are depicted in Table 3.5 and 3.6, respectively. The communication data frame is composed by seven bytes, where the first five are the start byte, the command, two command's parameters and the checksum, and the last two are used to get a response from the servomotor, such as the current joint angular position [24].

Parameter	Value
Baudrate	19200
Data bits	8
Stopbits	2
Parity	none
Handshake	none

Table 3.5: Communication Parameters [6].

byte	1	2	3	4	5	6	7
Controller	StartByte	Command	Param1	Param2	Checksum	0x00	0x00
Servomotor	High-Impedance					Return1	Return2

Table 3.6: Communication Frame [6].

3.3 Haptic Rendering

3.3.1 Source of feedback

A key feature in the haptic tele-operation approach is the transmission of some of the proprioception of the robot to the user. Haptic rendering is the process of computing the force feedback based on measurements of the robot's state to be applied on the human's hand. By taking into account that balance is a major concern, it is assumed that the most relevant sources of feedback are the ground reaction forces measured from the four load cells installed on the feet corners. The normal reaction forces allow estimate the centre-of-pressure (CoP) that determines the dynamic stability conditions of the humanoid robot. The COP is calculated based on the information obtained from the load cells using the following equation:

$$C\bar{O}P = \frac{\sum_i \bar{r_i} \times f_i}{\sum_i f_i} \tag{3.2}$$

where $\bar{r_i}$ is the coordinate of the load cell in the feet and f_i is the correspondent force.

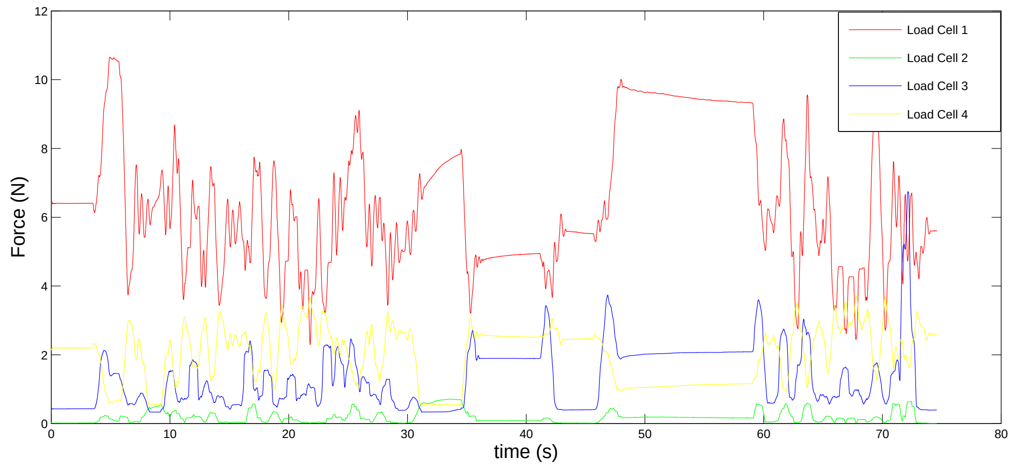


Figure 3.10: Information generated by the Load Cells example.

Figure 3.10 illustrates the information generated by the load cells of one foot for a simple example motion. From these data, the coordinates of the COP can be calculated through Eq. 3.2 and represented in two different ways: the global spatial evolution of the COP (see Figure 3.11) and the time course of the coordinates along the x and y axis (see Figure 3.12). As can be noted, when the robot is placed in the home position (*i.e.*, all servos with a zero angle), the x and y components of the COP is close to zero.

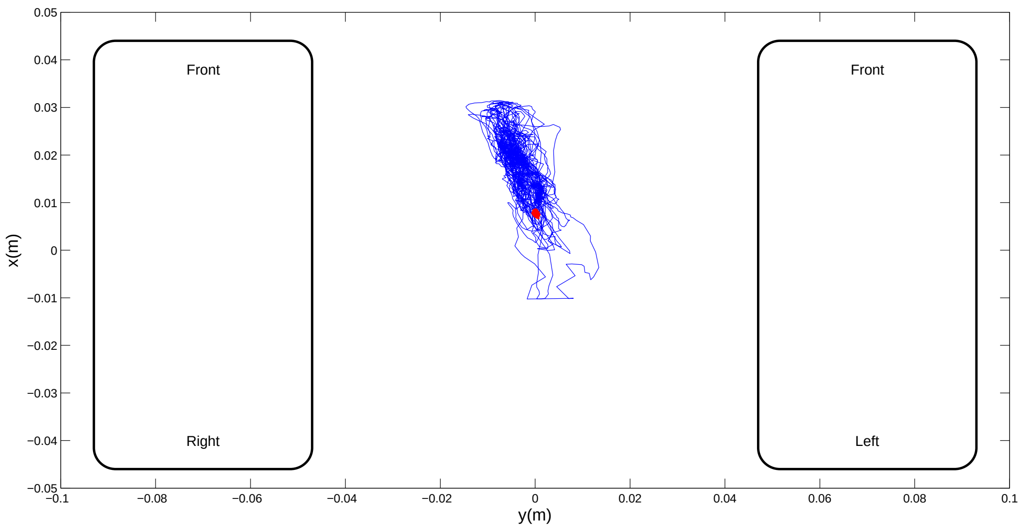


Figure 3.11: Generated COP example for both feet.

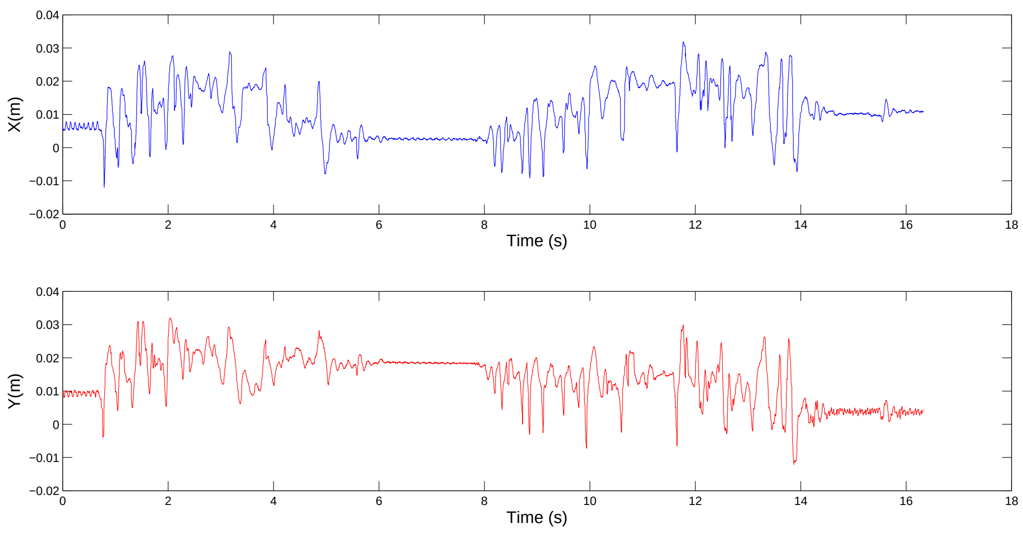


Figure 3.12: Generated COPX and COPY example.

3.3.2 Force Generation

Beyond the sensing step, the computational process of haptic rendering involves, in each cycle, the force calculation that is then fed back (actuation) on the human's hand. As mentioned before, the teleoperation system provides the user with force feedback based on what the robot is feeling in terms of COP coordinates (a balance index), informing him/her on the proximity of the limits of stability by means of a repulsive force (pushing the end-effector away). In line with this, the joystick synthetizes forces in the x and z components, while the force in the y component is constant aiming to counteract the force of gravity.

Aiming to provide the human operator with the sense of robot's balance, force feedback can be calculated in many different ways, using linear and nonlinear functions. First of all, it seems a good rule to avoid linear functions since small incremental alterations could be no longer felt due to user's habituation. Second, a deadzone should be considered to allow small displacements of the COP can be eliminated. Having this in mind, a sigmoid-type function (Richard's curve) is chosen, allowing for an easy parametrization of the S-shaped curve:

$$F = A + \frac{(K-A)}{(1+0.5*\mathrm{e}^{-B*(COP_u-M)})^{\frac{1}{0.5895}}} \tag{3.3}$$

The six parameters are defined as follows: A/(K) is the lower/(upper) asymptote that defines the minimum/(maximum) force exerted on the user, B affects the growth rate and M the deadzone. Figure 3.13 depicts different curves showing the effect of varying the growing rate B and 3.14 shows the effect of applying different deadzones for a fixed B, in this case 1.5. After a few experiments, it was found that the shape of this curve is particularly appropriate for the intended applications (described in detail in Chapter 5). In order to maintain a desired (and constant) COP, growth should be fastest at the start, while the maximum force feedback value is approached much more gradually.

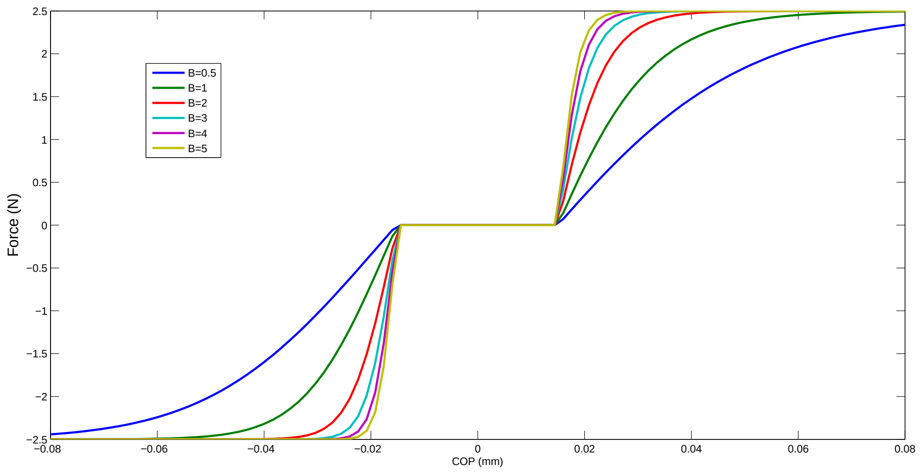


Figure 3.13: Sigmoid function for different B parameters.

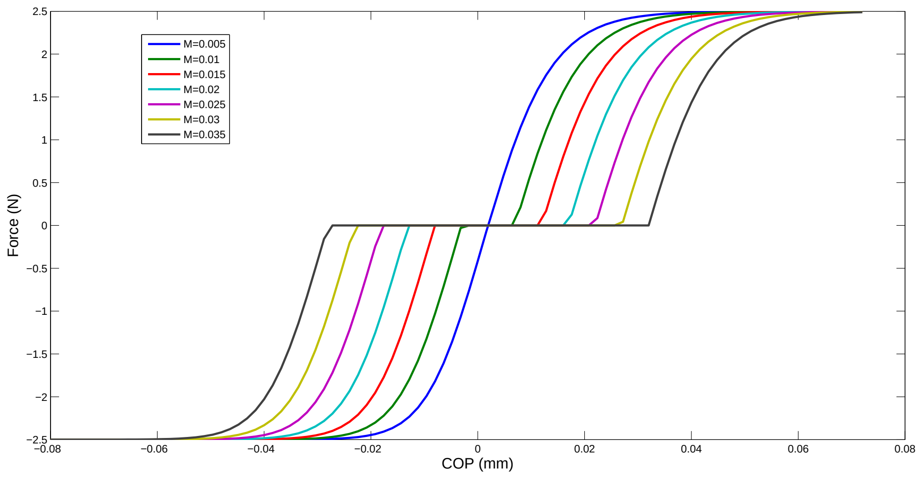


Figure 3.14: Sigmoid function for different M parameters.

This function shows a non-linear behaviour in the whole movement. Even with a small COP variation the operator can feel a higher or a small counterforce, simply by reducing or increasing the parameter B. Due to this characteristic, this function was the chosen one to be used during the experiments performed in this work.

Chapter 4

Software for Control and Data Logging

In this chapter the different processes that compose and work together to create the architecture behind this interface will be described. An important aspect that will be discussed is the data logging system that collects and records all the sensory information (*e.g.,* joint positions and ground reaction forces), as well as the commands guiding the execution of a specific task.

The ROS distributed environment makes possible to divide the different modules by categories depending on its function. For example, in this work the different modules are divided as follows:

- *Phantom_control* module belongs to the bases category since this module allows communication with the hardware device. More precisely, it is responsible for the connection between the joystick device and the host computer. This module subscribes the information needed to process the force feedback and publishes the data for later control of the humanoid robot.

- *Phantom_filter* module belongs to the utils category since performs data filtering. This module subscribes the *phantom_control* data, handles the data (*i.e.,* apply a filter to the data) and publishes it.

- *Humanoid_control* module belongs to the bases category, being responsible for the communication between the host computer and the humanoid robot. This module subscribes the data published by the *phantom_filter* module and uses that data to control the servomotors of the humanoid robot.

- *Pressure_cells* module belongs to the sensory category, being responsible for collecting sensory data from the load cells, calculating the COP and publishes it.

- *Haptic_force* module belongs to the perception, being responsible for interpreting the COP information and transform that data in force to be used by the *phantom_control* module.

Figure 4.1 illustrates the interconnection among the different modules developed under the ROS platform. A great advantage of this distributed architecture is the decentralized execution of the several tasks involved, making it easier to detect and correct software bugs.

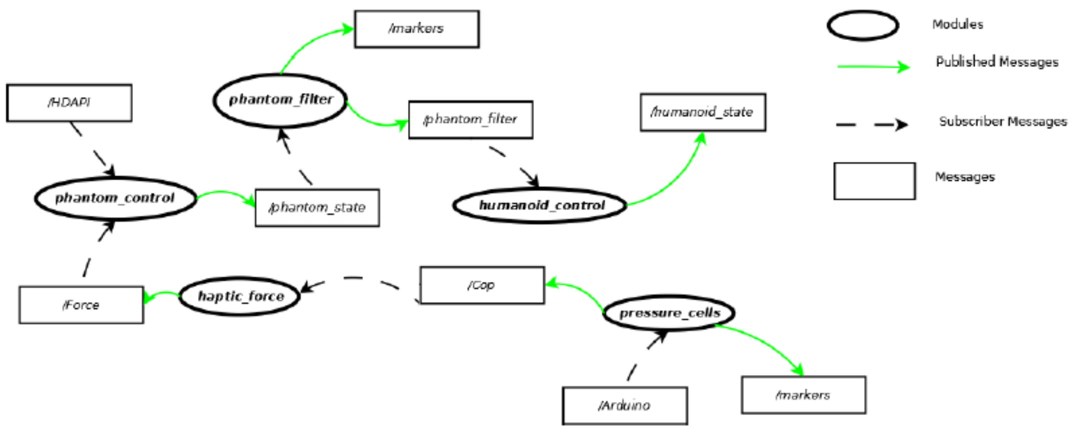


Figure 4.1: Interconnection between the different modules.

4.1 *Phantom_Control* module

This is the module responsible for retrieving the different data from the joystick and control the device force feedback. This module works by using just one node. This node can be divide into 2 distinct phases. The 1 st part begins by initializing the parameters that compose and will be published by the device :

- *position ([1 x 3] vector)*
- *rotation ([1 x 3] vector)*
- *gimbals ([1 x 3] vector)*
- *temp ([1 x 3] vector)*
- *buttons ([1 x 2] vector)*
- *thetas ([1 x 7] vector)*
- *home (boolean)*

- *home_position ([1 x 3] vector)*

The *position* gives the Cartesian coordinates X, Y and Z of the tip in mm. The *rotation* gives the angle from the base joints, the gimbals the angles from the tip joints and all these angles are stored in the *thetas* vector. The *temp* gives the temperature of the motors in the base of the device and the *buttons* refers to the state of the buttons from the device. Lastly, the *home* and *home_position* are used to set the home position.

Once the parameters are initialized, the device starts and an asynchronous scheduler is set to manager the servo loop. This scheduler initiates the communication with the haptic device and gets the information about the position, joint state, buttons state and temperature. If button 1 is not pressed, a force is created to send the device to the home position. This force will continue after the device is in the home_position to ensure it stays that way. If button 1 is pressed the force used is the one published by the *haptic_force* module. The thetas vector is then updated and the communication with the device stopped. In Figure 4.2, the scheme of the 1 st part is shown.

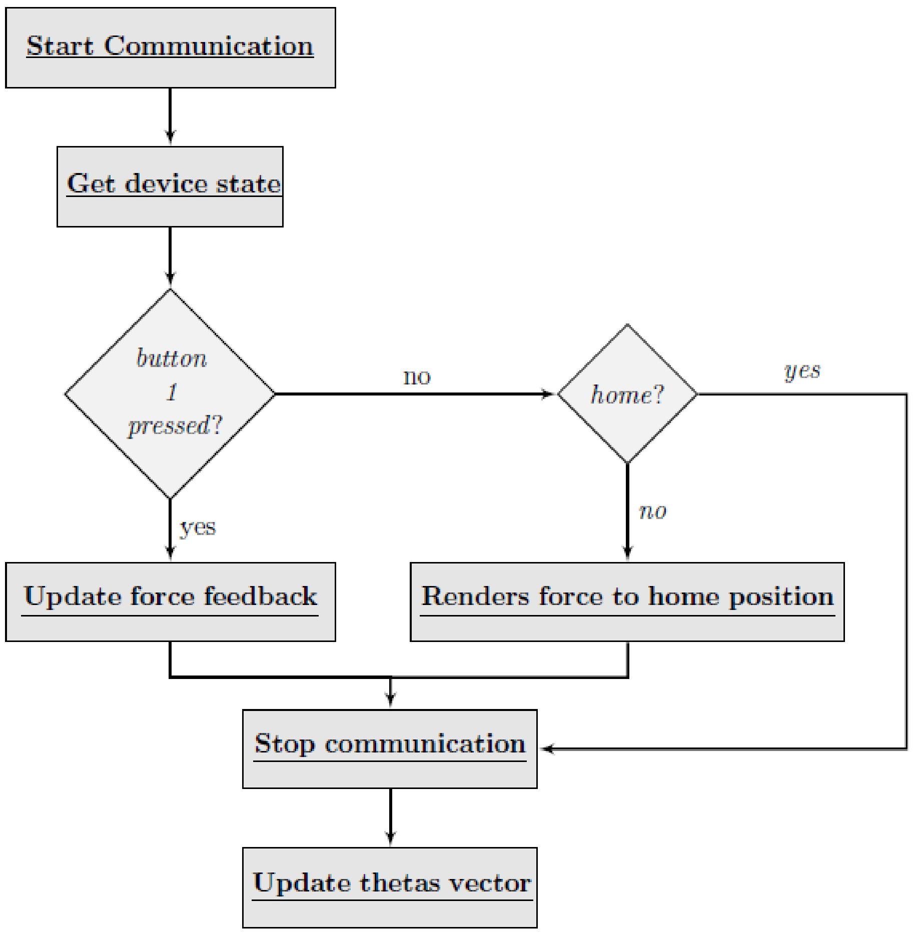


Figure 4.2: *phantom_control* scheduler's callback scheme.

The 2 nd part of the node, starts by enabling the force output and starts the scheduler. When starting the node if is unable to do it due to an error, the process is stopped. If the node starts the device does an automatic calibration, sets the *phantom_control* as advertiser (*i.e.,* this module is set to publish a certain topic during the time that ROS is active), subscribes the force and sets the frequency of the main loop. Each subscriber triggers a Callback (*i.e.,* function that will be evoked during the time that the ROS is active and as an argument the message of the subscribed topic), where the state of force is updated. Finally, in the main loop the new state is published and if the button 2 is pressed a new home position is set. In Figure 4.3, the scheduler's callback scheme is shown.

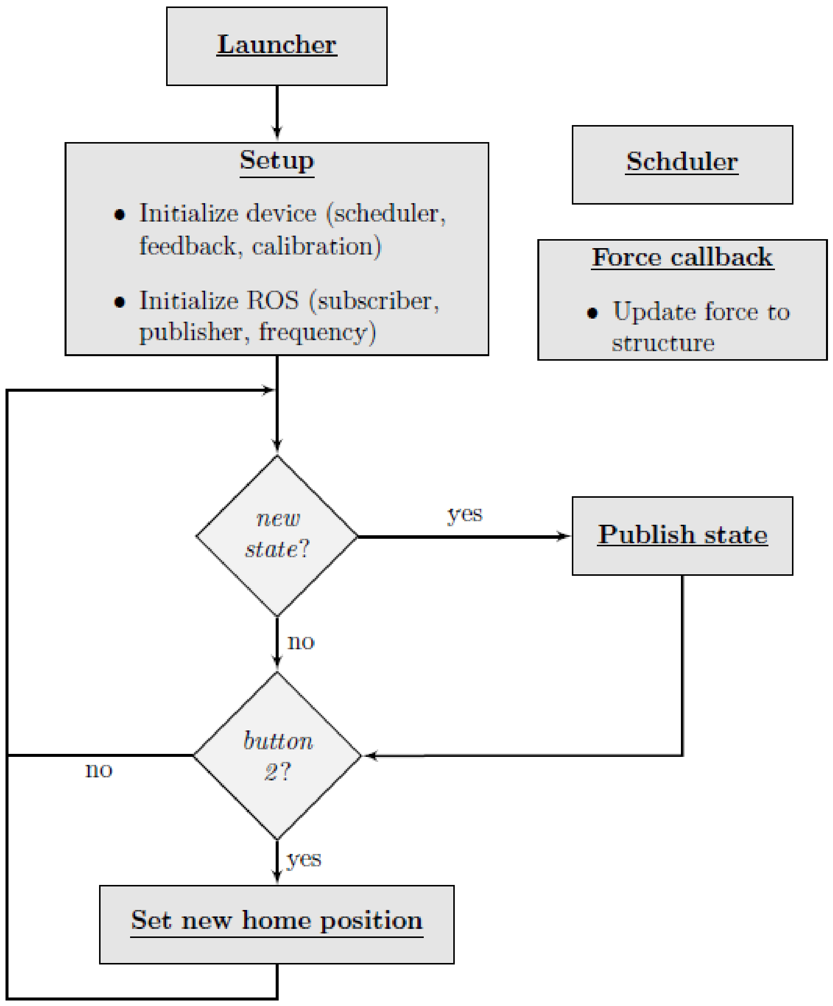

Launcher
Setup
• Initialize device (scheduler, feedback, calibration)
• Initialize ROS (subscriber, publisher, frequency)
Schduler
Force callback
• Update force to structure
new state?
yes
Publish state
no
button 2?
no
yes
Set new home position

Figure 4.3: phantom_control scheduler's callback scheme.

4.2 *Phantom_Filter* module

This module is based on the study conducted in the Chapter 3, Pre-processing of joystick raw-data. This module is essential for the correct operation of the humanoid robot and aims to filter the data from the joystick.

The node responsible for this is called *PhantomFilter*. This node starts by initializing the ROS environment and advertising two markers. These markers are the ones that will be used in the *Rviz* interface shown in Chapter 3. Besides publishing the markers, the node also subscribes the data from the *phantom_control* and publishes the new data. Each subscribed topic triggers a callback and inside this function the *phantom_control* data is updated. The variables that compose the data are:

- float64[3] Position
- float64[3] Rot
- int16[2] Buttons
- float64[3] force
- float64[3] Home_pos

The above fields have the same characteristics of the data published by the *phantom_-control* module with the difference that this data is filtered and this process of filtering and publishing only happens after a buffer reaches a number of points defined by user.

In Figure 4.4 a schematic explains the main operation of this node is shown.

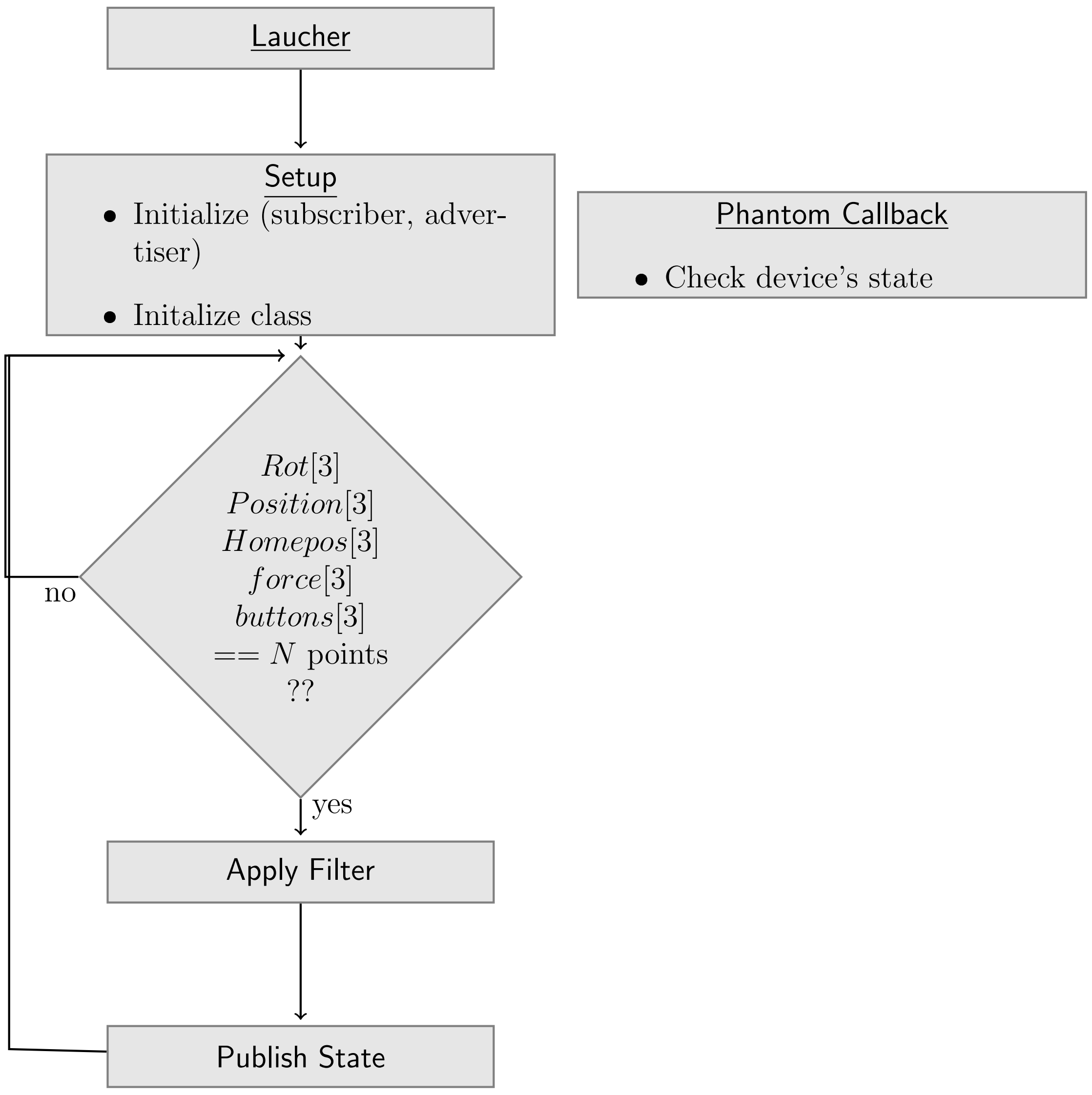


Figure 4.4: Program structure of phantom_filter node.

4.3 ***Humanoid_Control*** module

This module is responsible for the communication between the PC and the servomotors which control the humanoid robot.

First, in [6] a software has been implemented to control the servomotors using the RS-232 protocol. In [9], a class was created to help command these servomotors. This class has the capability to convert degrees into servomotors values and sends this information to the correct servomotor by knowing the ID.

The *Humanoid_Control* is the node responsible for controlling the servomotors. This node starts by initializing the ROS environment, setting the serial port communication, subscribing the message from the *phantom_filter* node and defining as advertiser the state of the humanoid robot. Before entering the main loop the humanoid robot is set to the home position. In the callback generated by the *phantom* subscriber, the phantom data is updated and its Cartesian Coordinates are transformed into joint angles using the inverse kinematics.

Finally, in the main loop the new angle values are compared to the old ones, if they are different in one degree the new state is published (speed and position). Each state is composed by the following fields:

- int16[13] speed_wanted
- int16[13] speed_now
- float64[13] joint_now
- float64[13] joint_wanted

The speed_wanted and joint_wanted represent the desire value for the next humanoid iteration. The speed_now and joint_now the real state of the humanoid robot.

In Figure 4.5, the structure of this node can be viewed.

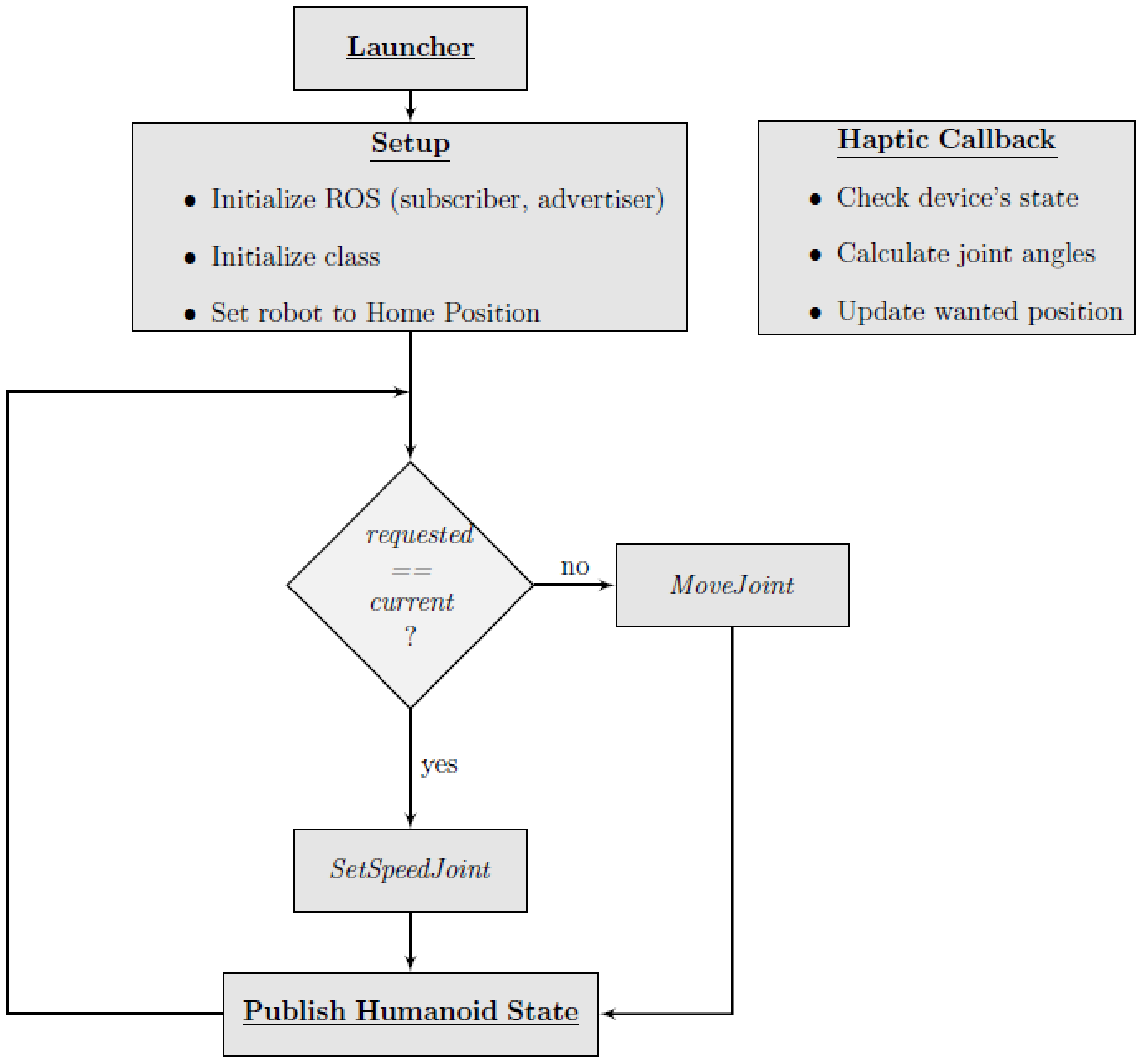


Figure 4.5: Program structure of humanoid_control node.[9].

4.4 *Pressure_Cells* module

The main objective is to retrieve information from the Arduino Nano and publish it in ROS format. This module has the function of retrieving sensory data and calculating the COP individually and globally for each foot.

In order to do this, two different nodes running in parallel are needed: One is responsible for retrieving the sensory information from the load cells and the other for computing the different COP (COPy, COPx and COP).

The first node is called *arduino* and has the function of read the values of the ADC. This node accepts as parameters configuration the serial port of each Arduino, if display or not the markers, and the identification of the corresponding load cells. Once the configuration is set, the ADC values are retrieved, the calibration routine is triggered and the load cells information is published.

The second node is responsible for calculating the COP values. This node accepts as parameter the information of which cells are communicating, sets the current frame identification and decides if it will show or not the visualization markers. After initializing the ROS environment, 3 advertises are set, one for each foot and one global. The loop starts until the variable *ros::ok* is true, check if the feet are in the same plan and if a callback run in the last iteration (*ros::ok1, ros::ok2*). If those parameters are true the COP is calculated and published. The different fields published by this module can be seen in the following list:

- float64 COPx
- float64 COPy
- float64 sen1
- float64 sen2
- float64 sen3
- float64 sen4

COPx and COPy represent the components of the global COP. The variables sen1 to sen4 are the values retrieved from the four load cells of each foot.

In Figure 4.6, the schematic of the force_cop node can be seen.

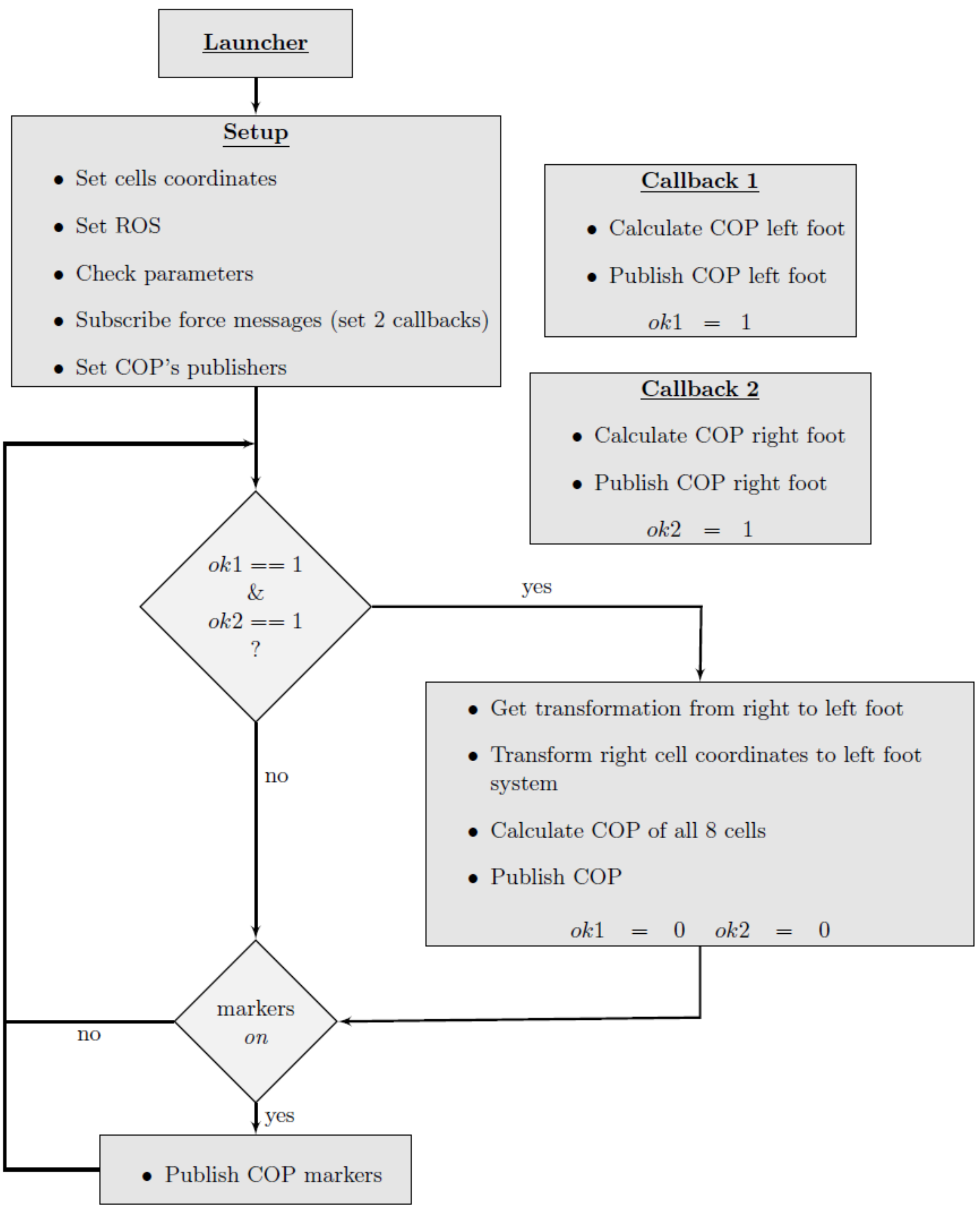


Figure 4.6: Simplified schematic of force_cop node.[9].

4.5 *Haptic_Force* module

The idea of this module is to turn sensory data, already in COP format into Force.

The node responsible for all of that is called *force_gen*. This node has a simple structure: after starting ROS, subscribes the COP message, sets the force message as advertiser and, before starting the main loop defines the loop's rate. Then a callback is triggered and, inside this callback, all the transformation described above occurs. Everytime the callback is run the variable corresponding to the COP, updates. The new COP value is used to calculate the force, once there are three different force generation (see Chapter 3, Haptic Force Rendering), the function that will be used is chosen by the operator. Each time the program is inside the main loop the new force values are published. The field published by this module is:

- float64[3] force

In Figure 4.7, a diagram explain the main structure of this node is shown.

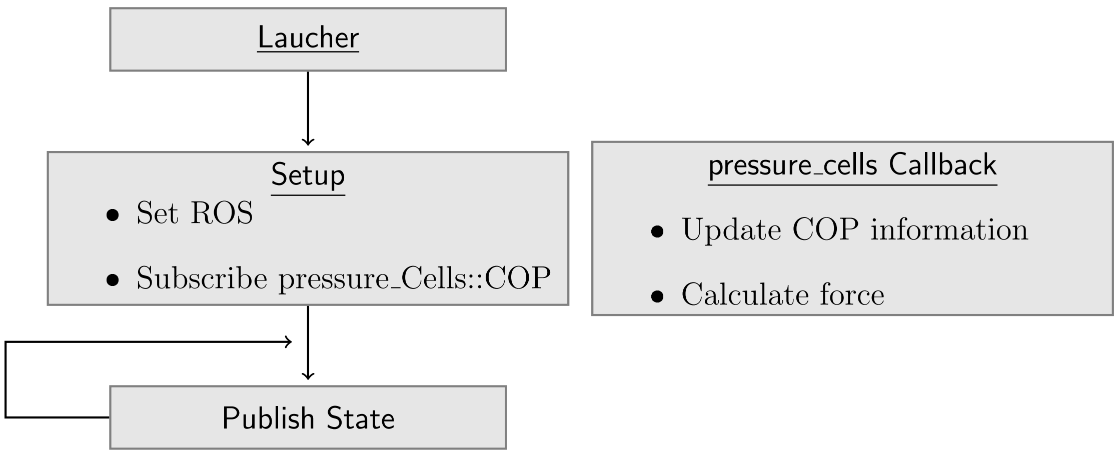


Figure 4.7: Program structure of force_gen node.

4.6 *Data Logging*

The main idea behind this work is to use a physical humanoid robot that, while tele-operated by a human subject using the haptic joystick, collects and records all sensory information and the control commands guiding the execution of a specified task. These datasets logged from the human-robot interaction can be later used for learning purposes. This task has been simplified in view of the variety of tools developed in ROS allowing to store, analyse and visualize message data. In this context, a bag is a file format in ROS created by the *rosbag* tool which subscribes to one or more ROS topics. These bag files can also be played back in ROS to the same topics they were recorded from, or even remapped to new topics.

Using bag files within ROS is, generally, no different from having ROS nodes send the same data. In order to avoid problems with time stamped data stored inside message data, the *rosbag* tool includes an option to publish a clock that corresponds to the time the data was recorded in the file. The bag file format is very efficient for both recording and playback, as messages are stored in the same representation used in the network transport layer of ROS.

Data logging was performed to the topics published by the five modules. All the variables are recorded using a simple node that subscribes the different topics and are saved in a .txt file. The variables of interest are the following:

- Time-stamp
- User actions on the Joystick (raw- and filtered-data)
- Feet force sensors
- Joint angular data
- Synthesized forces

Chapter 5

Teleoperation Experiments

This chapter describes some representative tele-operation experiments carried out with the real physical robot, while the human operator receives haptic feedback in the form of a force synthesized in the joystick. The experimental results aim to evaluate the utility of using a haptic joystick with force feedback in specific tasks involving robot's balance, as well as to illustrate the major challenges when transferring skills from humans to robots. In these experiments the robot's torso was removed to simplify the experiments and the user's challenges.

5.1 Experiments and Metrics

The proposed tele-kinesthetic teaching approach is evaluated in different scenarios that can be distinguished by the level of human supervisory control (semi- *vs.* full-control) and the influence of visual *vs.* haptic feedback. Given the large number of robot's DOF to control with a single joystick, the question posed was how to provide a natural and intuitive interface. As explained in Chapter 3, the location of the joystick's tip maps into the Cartesian coordinates of the hip section. In line with this, an inverse kinematics algorithm is used for translating the haptic commands into robot actions (*i.e.,* angular displacements of the joints).

Figure 5.1 represents the relative orientation of the x-, y- and z-axis of the reference coordinate systems associated with the haptic device Sp and the humanoid robot Sh. It is assumed a fixed alignment of the two coordinate systems during the bilateral control.

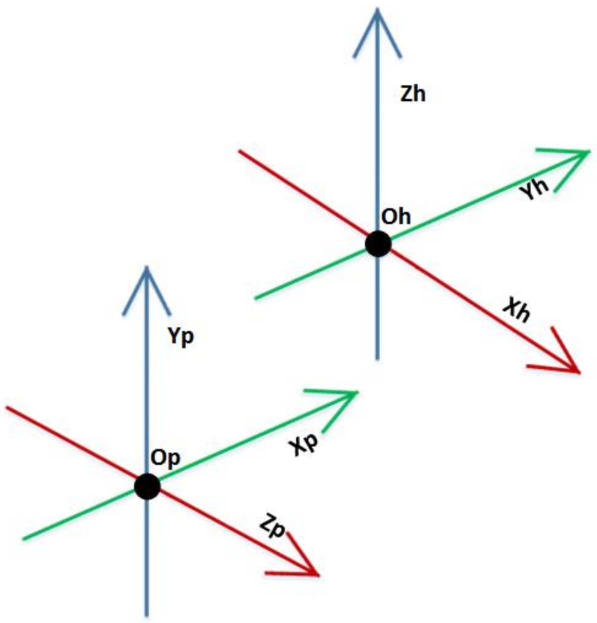


Figure 5.1: Reference coordinate systems of the haptic device and humanoid robot.

The key component of the system is the haptic interface allowing the tele-operation of the humanoid robot based on force feedback (in addition to visual feedback). Taking into account these experiments, it was decided to use the sigmoid-type function in order to synthetize the feedback forces. In any of the experiments, the outcome can be measured in terms of the real path of the COP since the stability is a priority goal. In line with this, two measures of statistical dispersion of the COP samples are used to help understand the level of improvement achieved when the experience is repeated over time.

- The first performance measure is the standard deviation estimated from the data samples as follows:

$$M1 = \sqrt{\sum_{i=0}^{N} \frac{(COP_u(i) - mean(COP_u))^2}{Ntotal}} \tag{5.1}$$

  where Ntotal represents the number of samples and the mean (COP_u) the mean of all the COP points.

- The second performance measure considers the COP at the initial configuration as the desired one and calculates the mean distance from this reference:

$$M2 = (\sum_{i=0}^{N} \frac{COP_u(i) - COP_u(0)}{Ntotal}) \tag{5.2}$$

  where Ntotal represents the number of samples and $COP_u(i)$ the different COP points.

The M1 and M2 performance measures are calculated for the x and y coordinates of the centre of pressure. For merging the two components into one single value, a global index is defined as the Euclidean norm (length of the vector) on the 2-D space. To complete the set of performance metrics, the area occupied by the COP is also calculated considering it a geometric figure (*i.e.,* the COP lateral and frontal extreme limits are considered as the limits of the geometry figure and the 'area' Matlab function is applied). For the proposed experiments, a small area is desired since it reflects the robot's tendency to stay in balance.

5.2 Robot's Balance Subject to External Perturbations

In the first experimental scenario, it is assumed that the human operator has the absolute control on the robot's actions. The experiment described here consists of using an industrial robot (FANUC) to create instabilities on the floor plate supporting the humanoid. The disturbance applied will be the same for all experiments and it is reflected in the angular movement performed by the support plate along the axis of interest in the time interval from 10 to 30 seconds (*i.e.,* during the first 10 seconds the support plate remains in a horizontal level). In response to these unexpected external perturbations, the user should try to keep the robot in balance by compensating the force sensations that are transmitted by the haptic device. In any case, the robot's feet are kept in contact with the ground at a constant distance from each other. Figure 5.2 shows the setup used for the experiments with the FANUC industrial robot.

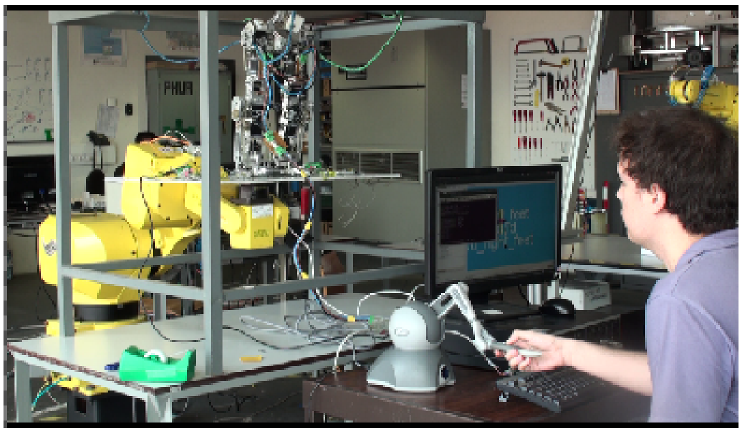


Figure 5.2: The setup for experiments with the FANUC industrial robot.

5.2.1 Balance in the Lateral Plane

In this experiment, the FANUC is used to create a lateral disturbance in the support plate (see Figure 5.3). In order to simplify the control actions, the robot's legs remain straight during the experiment, while the ankle and hip joints perform symmetric movements to maintain the hip section is a stable horizontal posture. The main goal is to compare the situation where the human operator just relies in visual feedback to that in which simultaneously relies in visual and haptic feedback. In terms of force feedback the parameters of the sigmoid function are the following: growing rate B=1.5 and deadzone M=0.015 mm.

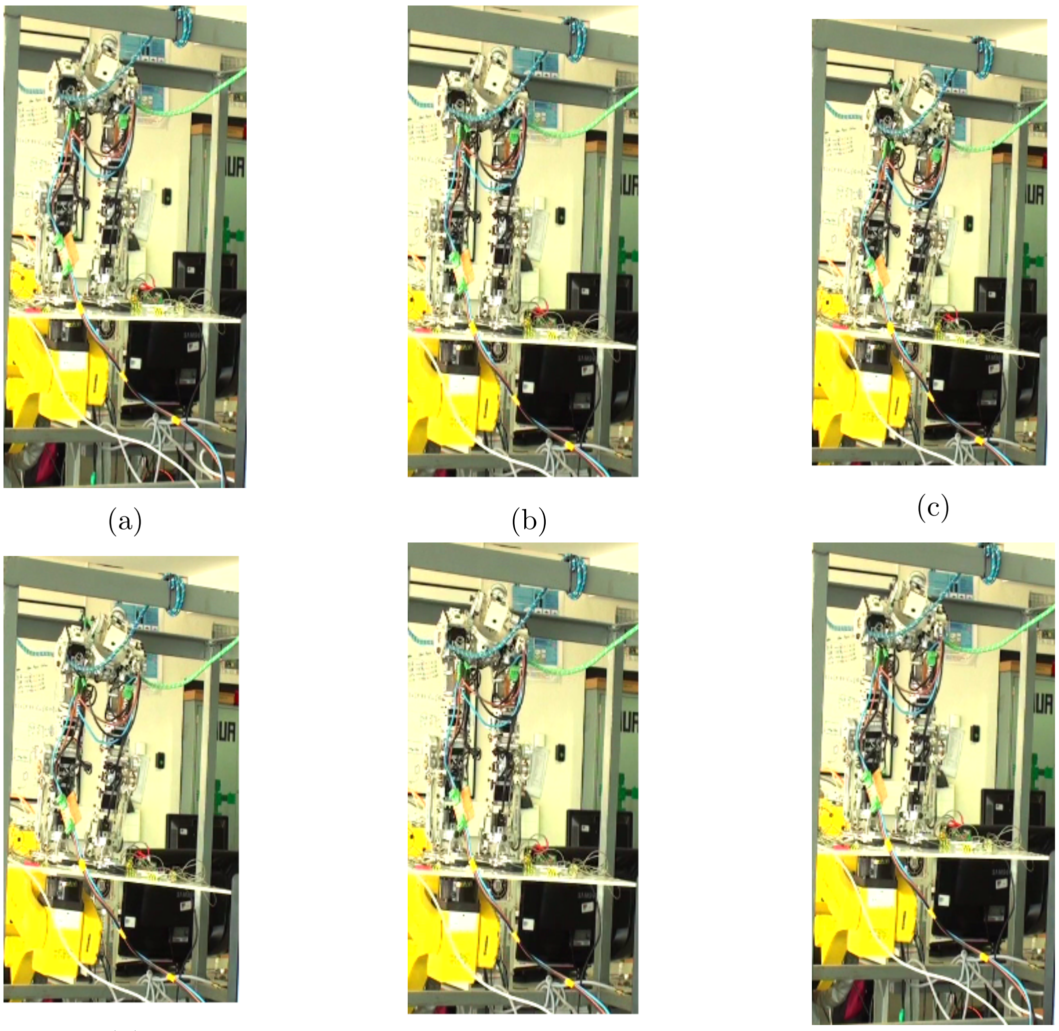

(a) (b) (c) (d) (e) (f)

Figure 5.3: Snapshots of the robot's balance in the lateral plane.

Only Visual Feedback

In this case, the human operator uses only its visual perception of unbalance to control the robot. The spatial variation of the COP coordinates and the respective time courses are represented in Figure 5.4 and Figure 5.5, respectively.

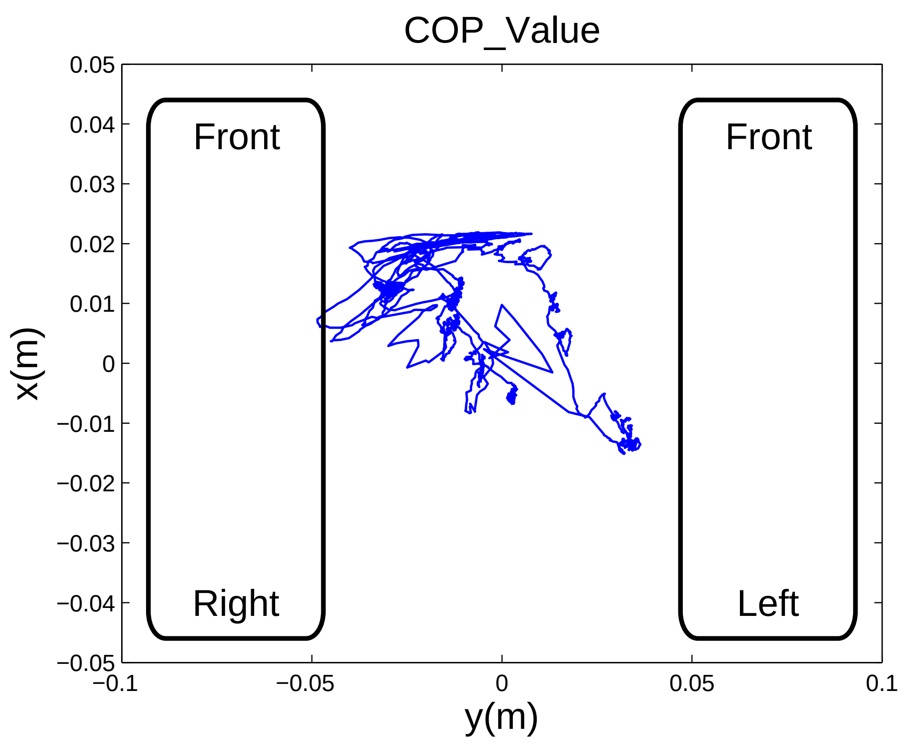


Figure 5.4: Spatial location of the COP under the feet without Force Feedback.

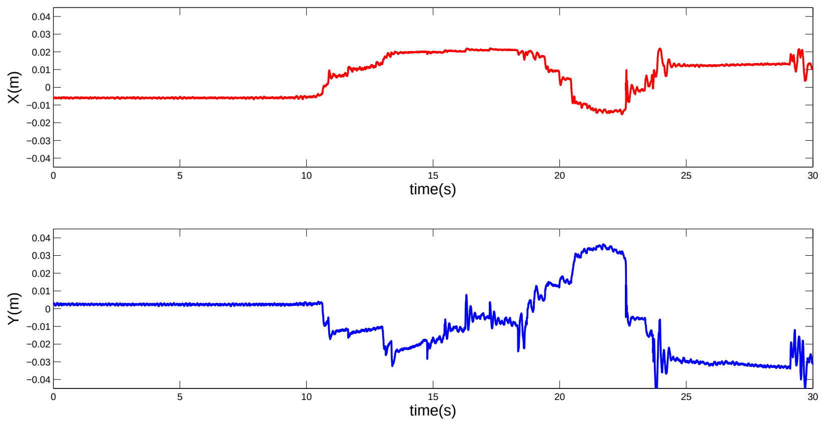


Figure 5.5: Trajectories of the COP in the x- and y-axis with without force feedback.

In the two figures above, it is possible to observe that the resultant COP coordinates show a significant variation in its two components. Further, the corrections produced by the human during the system's tele-operation can explain the more evident oscillations in the COP along the y-axis.

Table 5.1 summarizes the performance evaluation according to the previous measures.

	M_1 (mm)	M_2 (mm)	M_G (mm)	Area (mm^2)
x-axis	1.0700	1.2622	2.9948	0.0031
y-axis	1.8103	1.7186		

Table 5.1: COP metrics without using force feedback.

Visual and Haptic Feedback

In this experiment the human operator uses simultaneously visual and haptic feedback to perform the same task: react to the unbalance produced by the unexpected movement of the support plate. Figure 5.6 and 5.7 illustrates the COP generated in these circumstances.

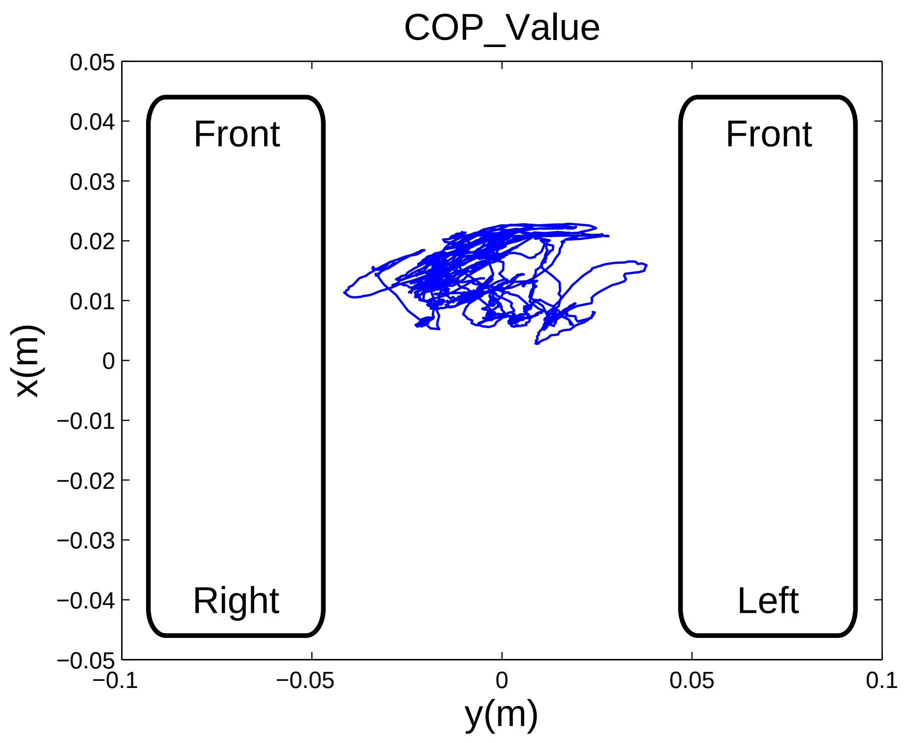


Figure 5.6: Spatial location of the COP under the feet with force feedback.

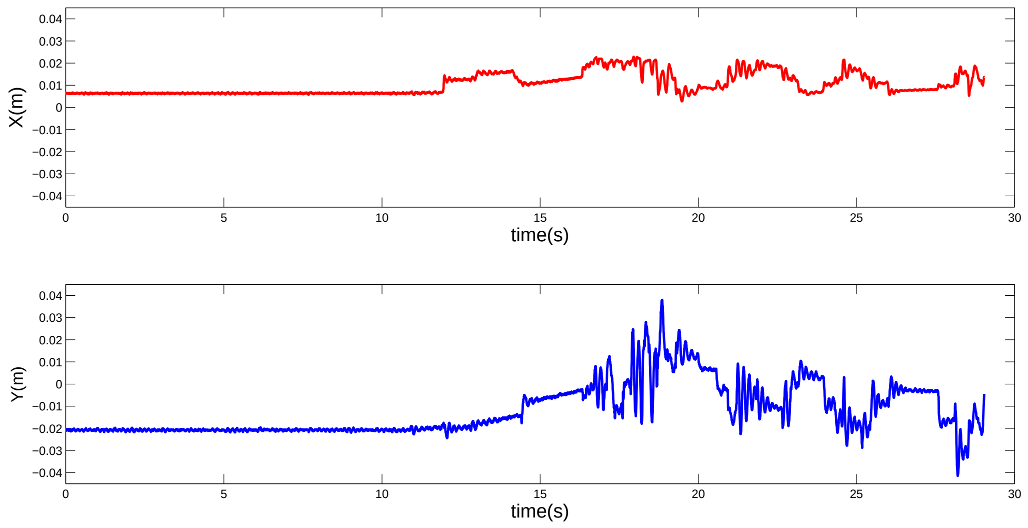


Figure 5.7: Trajectories of the COP in the x- and y-axis with force feedback.

As expected in the event of a lateral disturbance, the most significant changes occur now in the y component. At the same time, the bilateral control seems to make the operator feel more connected to the remote system and the improvements can be observed in the bottom of the Figure 5.7 In the same line of thought, Table 5.2 presents the performance measures used to assess the improvements achieved.

	M_1 (mm)	M_2 (mm)	M_G (mm)	Area (mm^2)
x-*axis*	0.4773	0.39687	1.5473	0.0016
y-*axis*	1.1209	0.86738		

Table 5.2: COP metrics using force feedback.

The control loop involves the output motion from the user and the input force from the humanoid robot. Figures 5.8 and 5.9 show the time course of the force generated on the user's hand and the joystick commands from the user, respectively. For comparative purpose, Figure 5.9 superimposes on the same graph the joystick tip position when only visual feedback (green) and when combining both visual and haptic feedback (blue). On the one hand, the force generated consists of both peaks and periods in which the values remain almost constant. On the other hand, the raw- and filtered-data (for a slope resolution of $\frac{\pi}{4}$) from the joystick indicate how the human operator tries to compensate the robot's unbalance in the different phases of the movement (the dashed line represents the angle of inclination of the support surface).

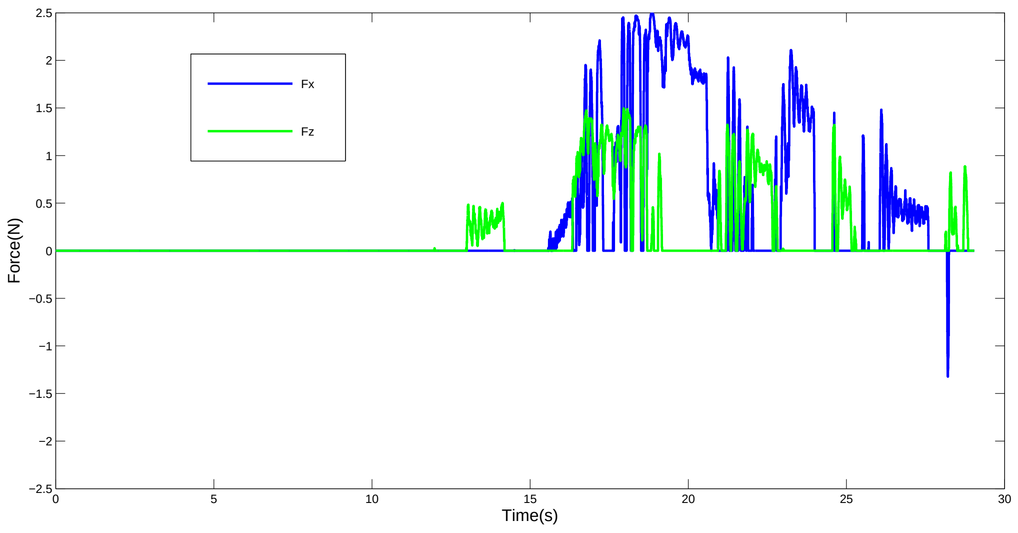


Figure 5.8: Force generated using visual and force feedback.

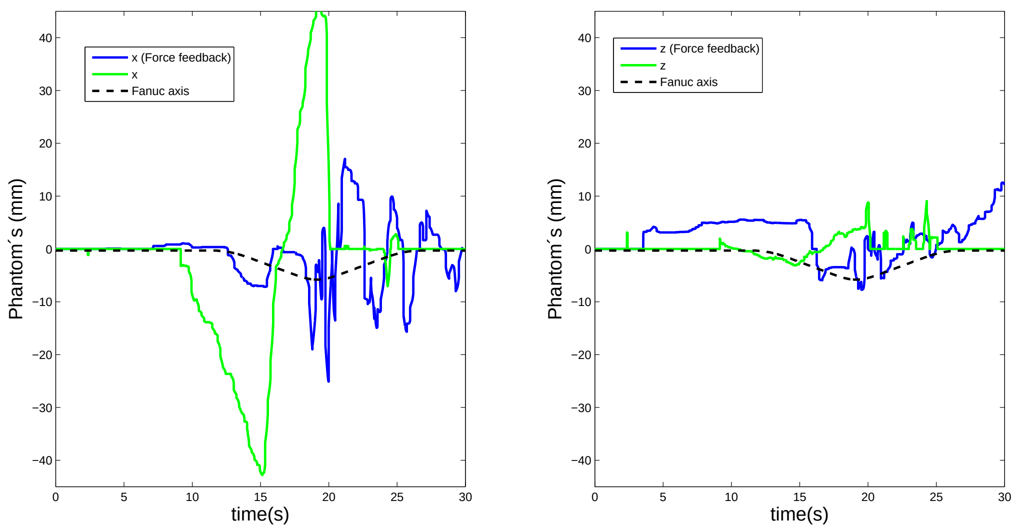


Figure 5.9: Joystick variation during the movement in the lateral plane.

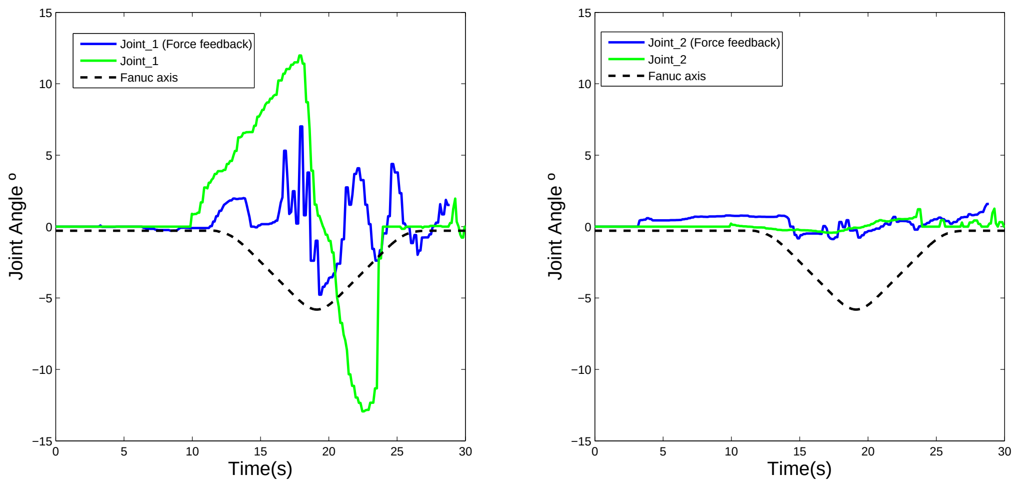


Figure 5.10: Ankle variation during the movement in the lateral plane.

To complete the picture, Figure 5.10 shows the time evolution of the robot ankle joints. Once again, the curves relating to visual and haptic feedback (blue) are superimposed. Despite improvements, Figure 5.9 and Figure 5.10 illustrate the major challenge in force feedback: the difficult trade-off between stability and performance. The constant variation of the angles shows that the operator is trying to reverse the unbalance phase. There is a relationship between the force felt by the operator, its control of the haptic device and, consequently, the humanoid robot actions. Observing the humanoid's motion, it is possible to view that when the slope reaches a maximum value the same happens to the joint angles of the robot.

5.2.2 Balance in the Sagittal Plane

In the second part of this experiment, the perturbation applied by the FANUC to the support plate affects the sagittal plane (see Figure 5.11). Here, force generation is based on the sigmoid function with the same parameter values.

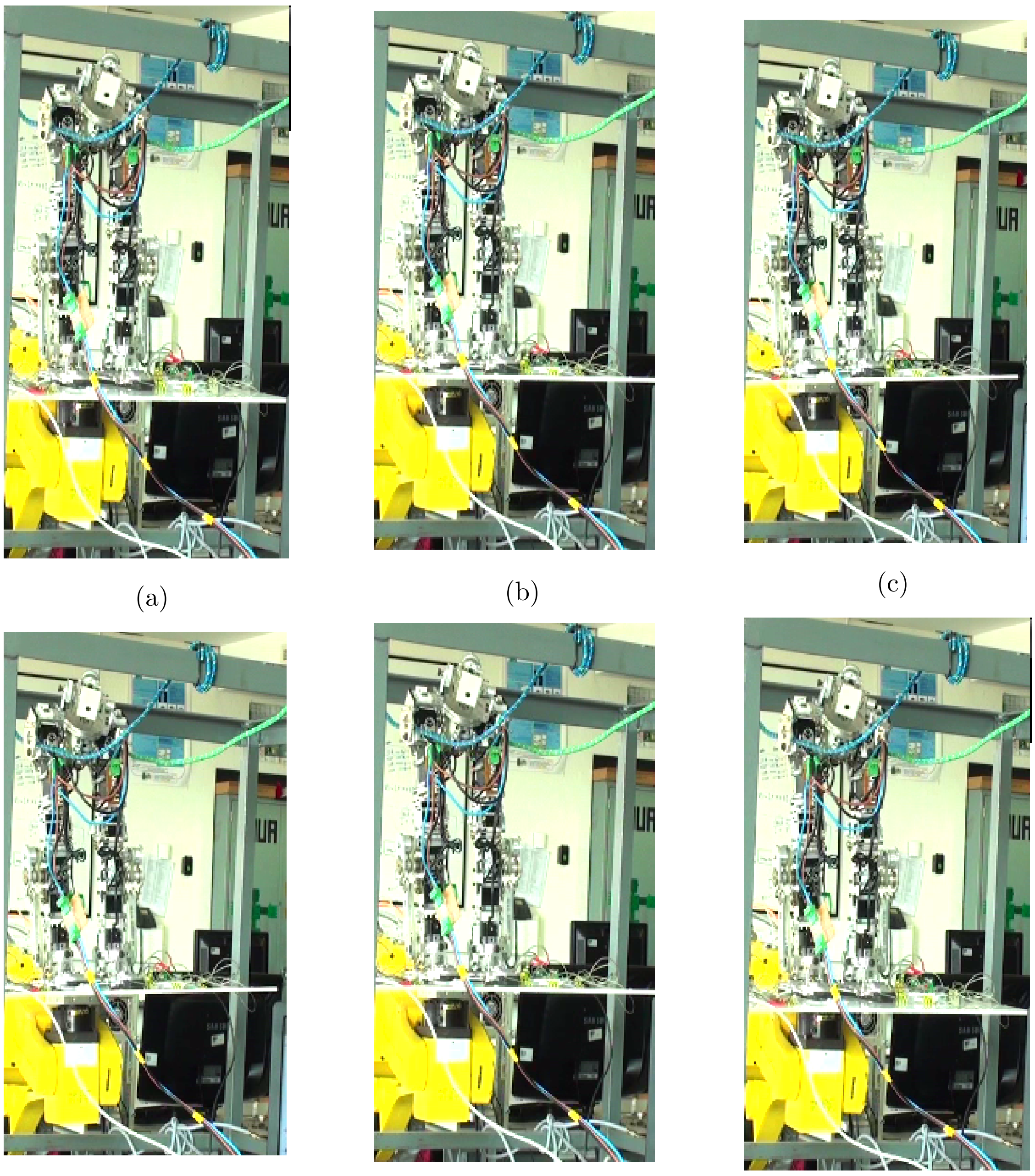

(a) (b) (c)
(d) (e) (f)

Figure 5.11: Snapshots of the robot's balance in the sagittal plane.

Only Visual Feedback

Following a procedure similar to the previous sub-section, the operator is instructed to guide the robot's behaviour using only the vision sense. The spatial variation of the COP coordinates and the respective time courses are represented in Figure 5.12 and Figure 5.13, respectively.

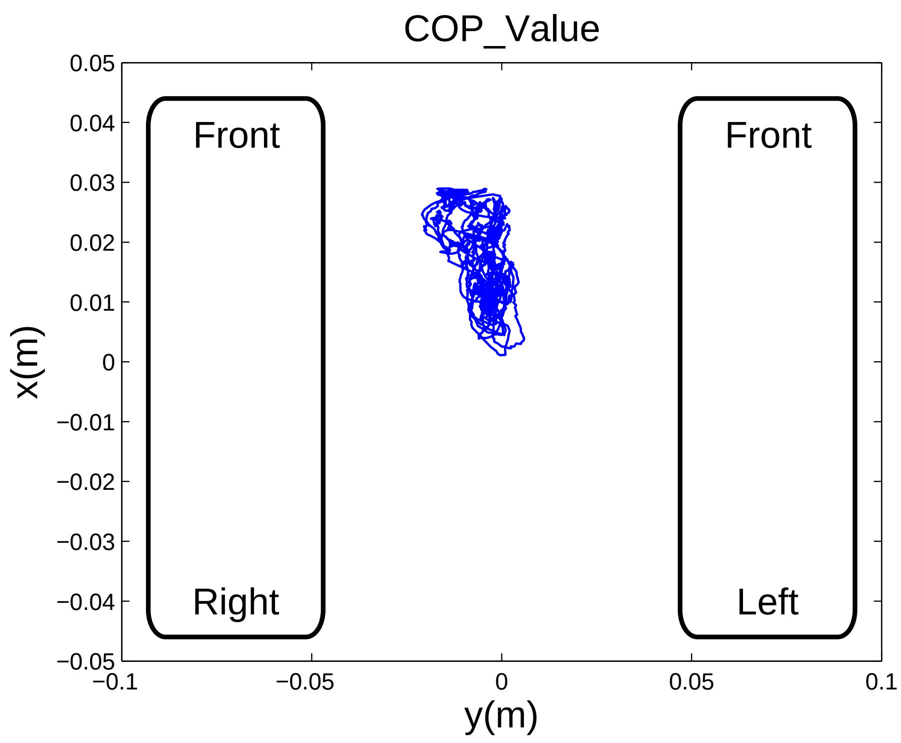


Figure 5.12: Spatial location of the COP under the feet with force feedback.

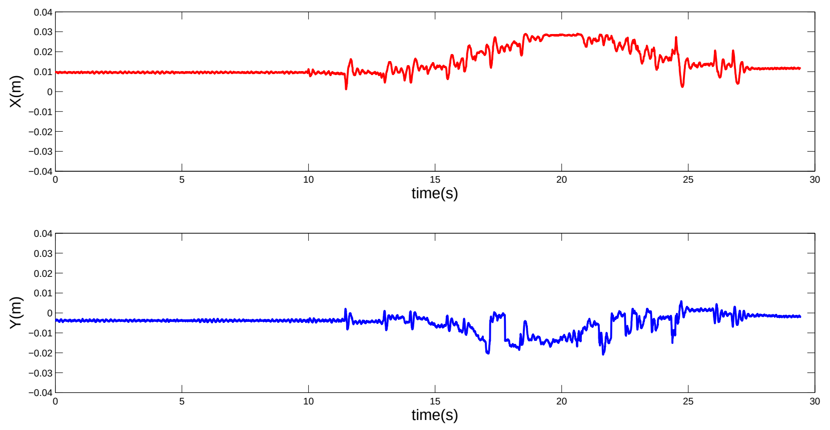


Figure 5.13: Trajectories of the COP in the x- and y-axis without force feedback.

In this experiment, the variation of the COP's coordinates is not as pronounced as before. The same result is evidenced by the values shown in Table 5.3.

	M_1 (mm)	M_2 (mm)	M_G (mm)	Area (mm^2)
x-axis	0.6309	0.45415	0.9206	0.0009
y-axis	0.4144	0.26745		

Table 5.3: COP metrics without force feedback.

Visual and Haptic Feedback

The repetition of the experiment, combining the visual and haptic modalities, led to the results presented below. First, the COP coordinates are more concentrated around the area of interest, mainly the x component (see Figure 5.14 and Figure 5.15).

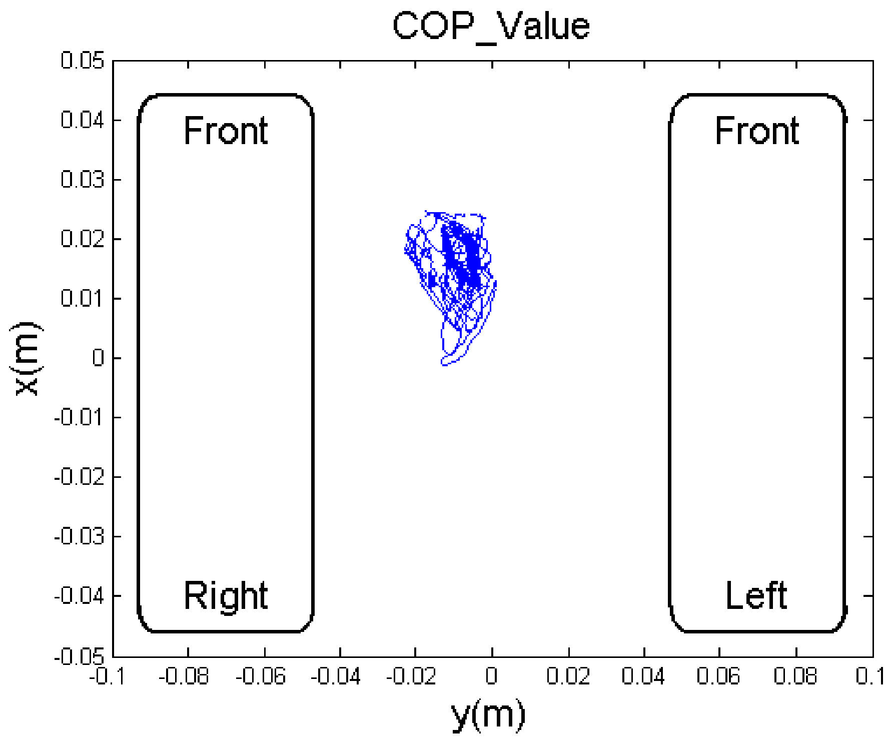


Figure 5.14: COP using Force Feedback.

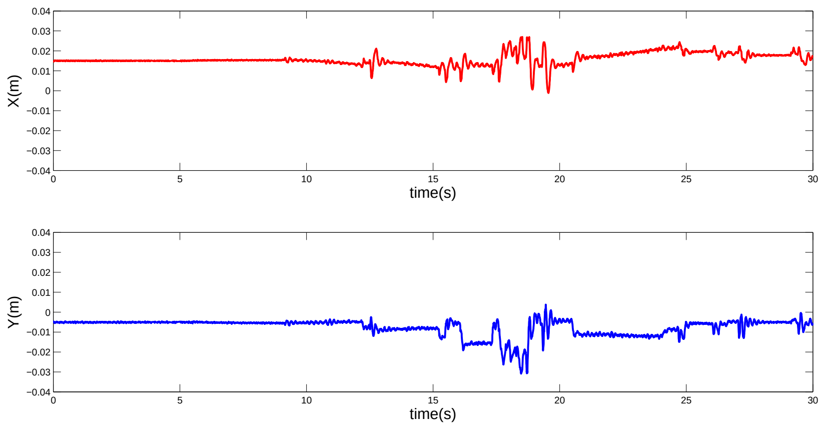


Figure 5.15: COPx and COPy using force feedback.

Table 5.4 shows the improvements achieved, except in what concerns the performance measure M2 calculated on the y-axis. Contrary to the lateral perturbation, the force generated is now dominated by peaks (see Figure 5.16) meaning that the human operator receives a sequence of contradictory signals that must handle to accomplish the task objective.

	M_1 (mm)	M_2 (mm)	M_G (mm)	Area (mm^2)
x-axis	0.3107	0.22157	0.6102	0.0009
y-axis	0.3773	0.29041		

Table 5.4: COP metrics using force feedback.

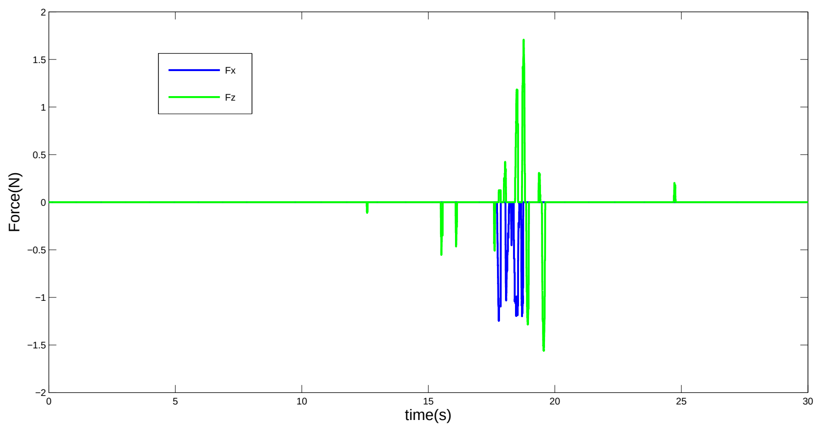


Figure 5.16: Force generated using visual and force feedback.

The commands from the user are depicted in Figure 5.17 (for a slope resolution of $\frac{\pi}{4}$). The raw- and the filtered-data shows a tendency and the time curves a smother evolution.

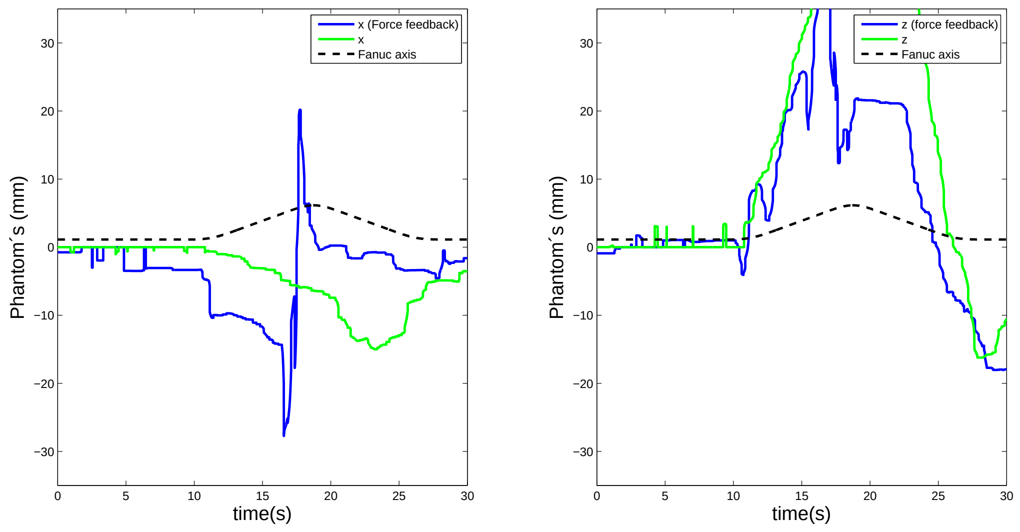


Figure 5.17: Joystick variation during the movement in the saggital plane.

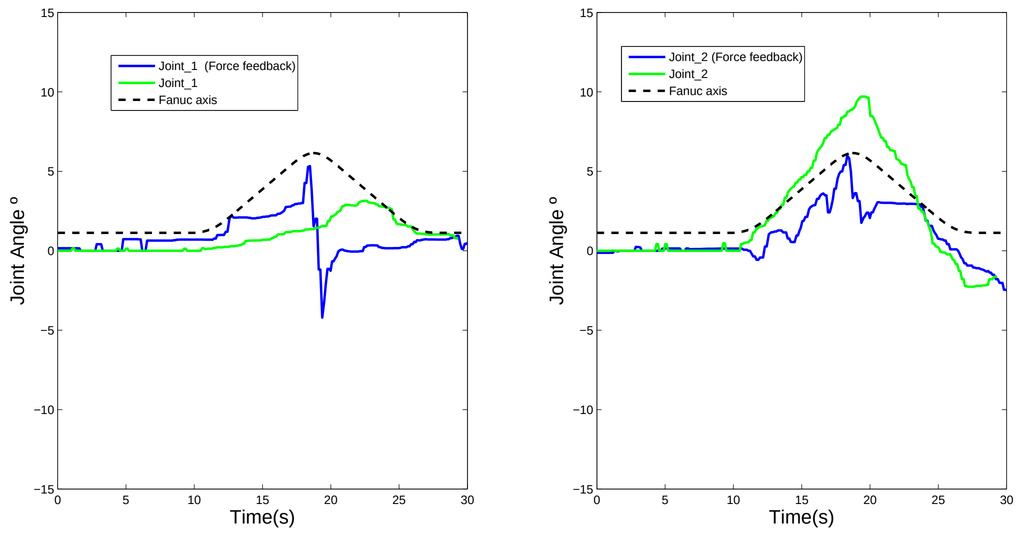


Figure 5.18: Ankle variation during the movement in the saggital plane.

The variation of the joint angles and the phantom's axis show that the movement done by the robot is not constant, *i.e.,* the operator is constantly counteracting the movement. Another curious aspect is the small variation at the beginning of the movement, although there is no force actuating in the first 10 seconds, the operator may not have the joystick completely stopped. Summarizing the results obtained in this experiment, it can be stated that the role of force feedback is important, but needs to be explored in a gradual way in order to become a useful tool for the purpose of teaching maneuvers like balancing.

5.3 Down/Up Motion of the Hip

In the second experimental scenario, a down/up motion of the robot's hip section with both feet on the ground (crouching task) is performed with human supervision for semi-autonomous humanoid operation. Here, the role of the human subject is to achieve an optimal trajectory by minimizing the variation of the COP during the execution of the predefined vertical movement. The option for a semi-supervisory control is possible (and desirable) because prior knowledge about the particular task is available. More specifically, movement along the vertical axis is pre-planned in the Cartesian space using 5th-order polynomial functions, while the other two coordinates are left free for the human operator aiming to minimize online the COP variation with respect to the initial value. The motion performed with human supervision is compared with a perfect vertical motion planned autonomously by the humanoid robot (*i.e.,* without any human intervention). After specifying the goal, the planner is invoked to compute a vertical trajectory connecting the robot's initial configuration to this target configuration. The pre-defined movement is divided into three phases: First, the hip section moves down to a predefined target height in about 1.5 seconds. Afterwards, the robot remains in that configuration for a short period of time. Finally, the robot performs the inverse movement to reach the initial height in the same 1.5 seconds. The snapshots of this experience are shown in Figure 5.19

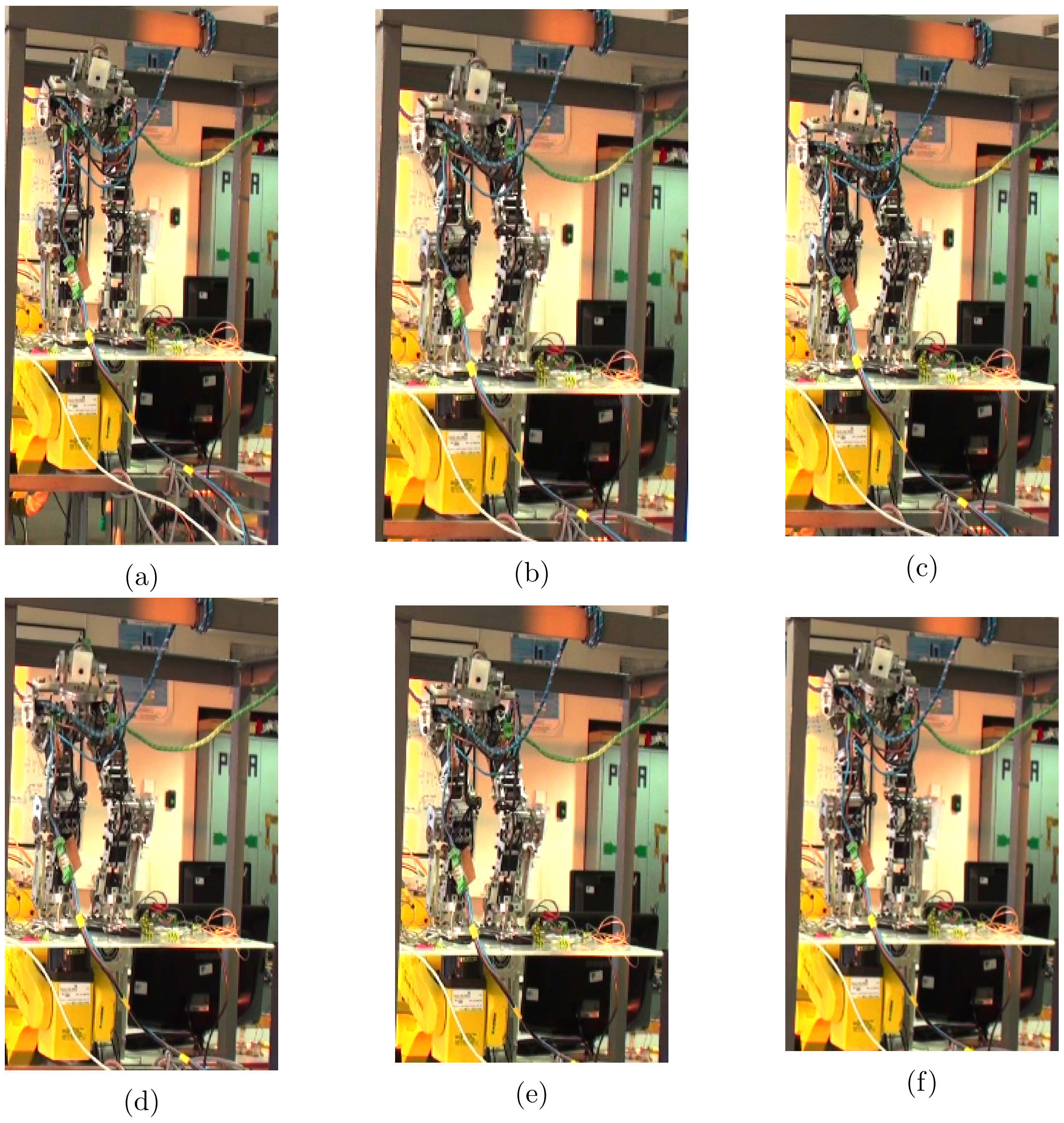


Figure 5.19: Snapshots of the down/up motion of the hip section.

Figures 5.20 and 5.21 show the spatial location of the COP under the feet using the two control modes in comparison. In the case of supervised control, the COP variation in the x component is higher than expected meaning the human operator was unable to find an optimal path. The differences can be better analysed from the temporal evolutions shown in Figures 5.22 and 5.23. For example, repeating the experiment with the same robot's initial configuration gave rise to very different initial COP coordinates (x-axis).

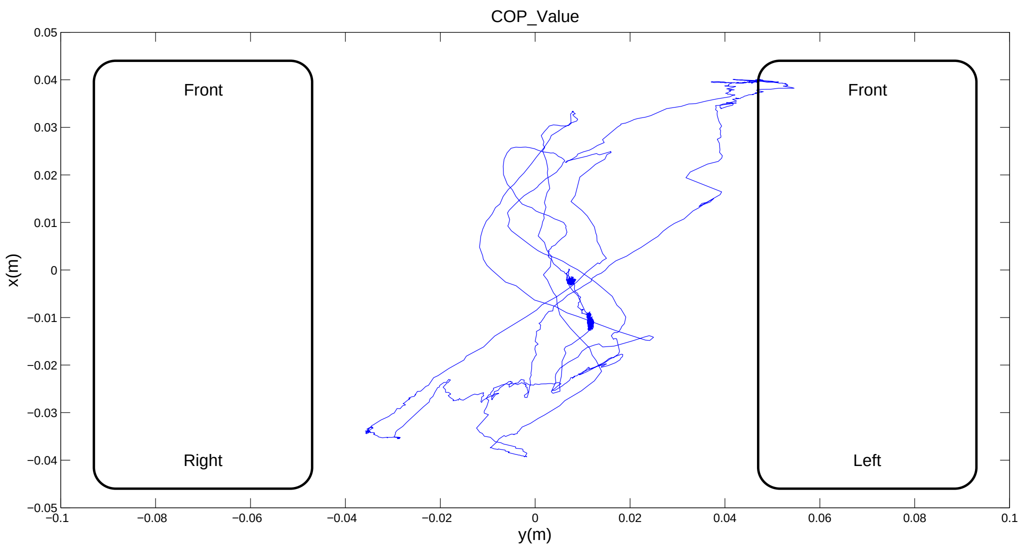


Figure 5.20: Spatial location of the COP under the feet with autonomous control.

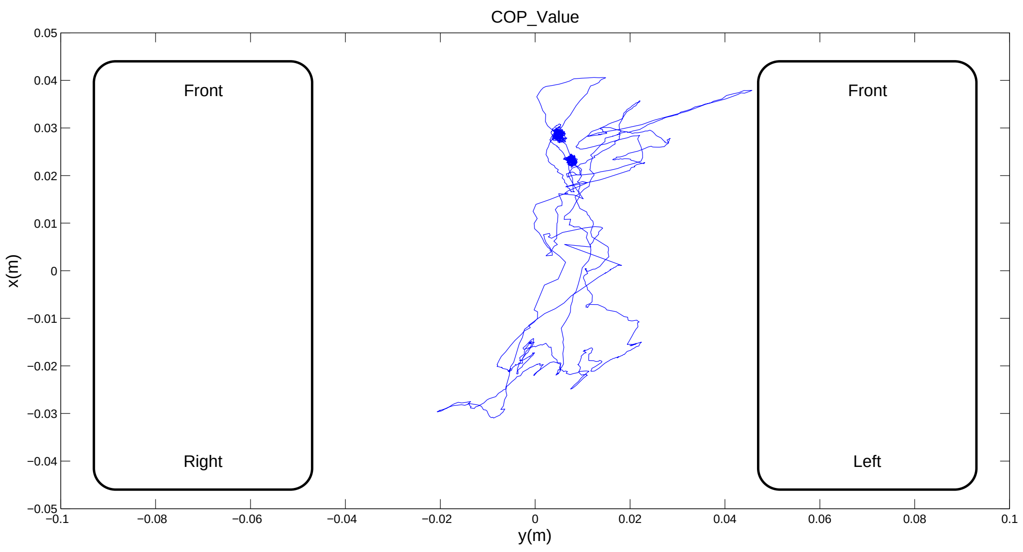


Figure 5.21: Spatial location of the COP under the feet with semi-supervisory control.

Using the haptic joystick to find a better path in balance, the resultant COP is shown in Figure 5.22 and 5.23.

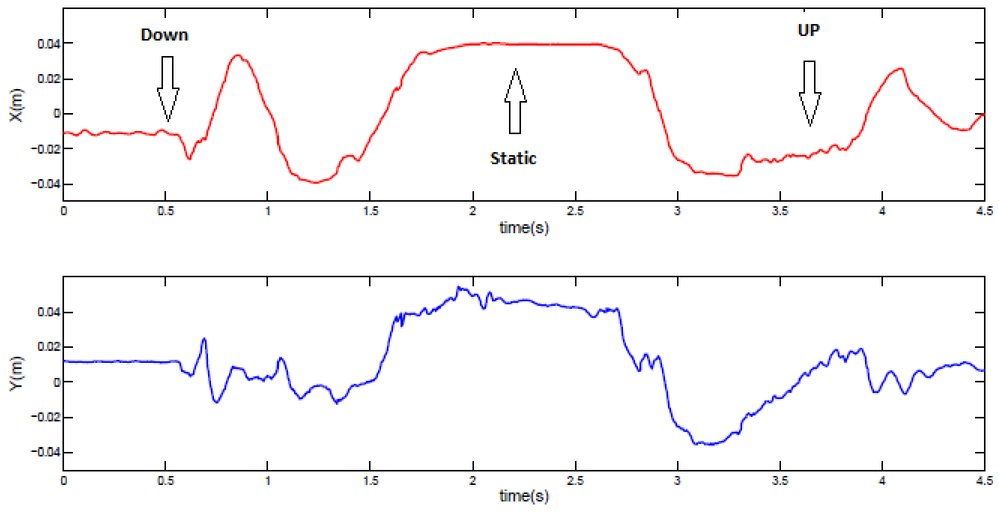


Figure 5.22: Trajectories of the COP in the x- and y-axis with autonomous.

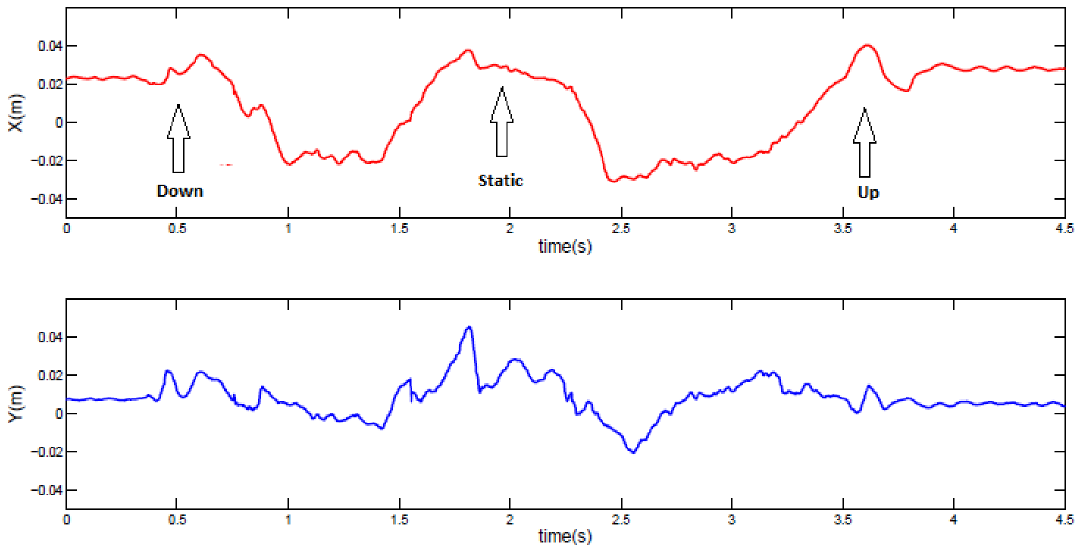


Figure 5.23: Trajectories of the COP in the x- and y-axis with semi-supervisory control.

The reason why the differences found are not very significant may be due to the complex dynamics of the humanoid robot dominated by non-linear phenomena (*e.g.*, backlash, friction, etc) that defeat the human intervention. These results indicate that one possible strategy to explore may be to combine the ground reaction forces with inertial and vision sensors to obtain more consistent information for synthetizing the forces relayed to the human operator.

In Table 5.5 the metric results are shown.

		M_1 (mm)	M_2 (mm)	M_G (mm)	Area (mm^2)
full-autonomous robot	*x-axis*	1.139	0.99219	1.8062	0.0072
	y-axis	0.69769	0.7028		
supervised control	*x-axis*	1.2156	0.95426	1.6428	0.0047
	y-axis	0.3805	0.40699		

Table 5.5: COP metrics.

The metrics clearly demonstrate that the COP of the humanoid robot suffers a large displacement. The results using haptic feedback are slightly better, but, as mentioned before, the difference is not considerable enough to allow definitive conclusions.

5.4 Discussion of Results

Globally, the results obtained show that the use of haptic joysticks with force feedback is an important aspect to consider, while leveraging the skills and capabilities of human users. However, further study should be conducted in order to fully understand their potential and implications in tele-operation of humanoid robots. Although the proposed approach is not expected to require specific prior knowledge on robot tele-operation from the user, practicing tele-operation skills seems essential in order to perform repeatable and coherent experiments. In this work, it was not possible to extend the training phase to allow the human operator to better understand what force rendering algorithm is most effective, what sensorial information reflects the real-word dynamics and how the robot reacts to the human intentions. Thus, further study will be required to understand the implications of these important factors on the overall system's performance. Nevertheless, the experiments presented in the later sections illustrate one of the basic challenges in force feedback: the difficult trade-off between stability and performance. Stability analysis is difficult task because any model of the system depends on the user and the environment itself (*i.e.,* the humanoid robot). As an alternative, performance metrics to help the human teacher understand the level of improvement achieved when repeating the same task over time is a fundamental factor to consider in future.

Chapter 6

Conclusions and Future Work

In this final chapter, the conclusions of this work and suggestions to keep improving this work will be explained in the next two sections, respectively.

6.1 Conclusions

This work was composed by three main objectives. The first objective was to developed an haptic interface for the right tele-operation of the humanoid robot. This objective was accomplish with success. Using one of the most important characteristic of the ROS environment, its modular characteristics, a new firmware interface was created to control the humanoid robot in different tasks. This new interface can be generalized and with small alterations applied to other tasks.

The second objective was to study and treat the different conditions that can interfere with the control of the humanoid robot by the human operator. This objective was divided into two parts: Study and treatment of the data responsible for controlling the humanoid robot and the force rendering problem. This objective was done successfully, although some improvements can be added to achieve better results (see Future Work section). The filter applied to the data before the transformation into angle values allows that the desired action is actually executed and in cases of unexpected noise, the robot is protected. To solve the force rendering problem, one force function was created, this new force show capabilities and can be an improvement to the human operator.

The last objective of this work was to validate and prove that the use of force feedback can be an improvement for tele-operation. The two experiments, using the fanuc, prove that with force feedback the path done by the robot is more constant and stable than the one done using only vision feedback.

Using haptic devices with force feedback to control humanoid robots presents in this way a good solution to retrieve valid data to be used in the next major step, Learning by Demonstration.

6.2 Future Work

This work can be considered as a preparatory step to achieve the ultimate goal of an autonomous humanoid robot with learning abilities. Before this is possible several other steps and improvements are required, such as:

- **Systematic and repeated experiments:** The preliminary results achieved with a real humanoid robot are promising and relevant, even if the methodology has much to be explored. In particular, systematic and repeated experiments with different subjects are needed since the success of the haptic teleoperation system is highly related with the user's experience.

- **Force generation:** Currently, the force feedback used in all experiments is based on the COP calculated from the load cells in the feet. Since the humanoid robot has available a set of inertial sensors, it would be interesting to include this sensorial information (complementary) when synthetizing the forces relayed to the human operator.

- **New scenarios and tasks:** In this work, only simple tasks involving balance were tested. Exploring new scenarios towards more challenging tasks (**e.g.,** biped locomotion with two Phantom Omni joysticks) are within the next goals.

- **Learning algorithm:** The next logic step is to use the recorded datasets to apply a learning algorithm that can help in developing sensorimotor strategies for humanoid behaviour without a *priori* rigorous models.

References

[1] Petar Kormushev, Dragomir N. Nenchev, Sylvain Calinon, and Darwin G. Caldwell. Upper-body kinesthetic teaching of a free-standing humanoid robot. In *2011 IEEE International Conference on Robotics and Automation*, pages 3970–3975. IEEE, May 2011.

[2] Baris Akgun and Kaushik Subramaman. Robot Learning from Demonstration: Kinesthetic Teaching vs. Teleoperation, 2011.

[3] Humanoid Robots Lab. `http://hrl.informatik.uni-freiburg.de`.

[4] da Vinci Surgery - Minimally Invasive Robotic Surgery with the da Vinci Surgical System.

[5] Manfred Hiller Daniel Germann, Tobias Bruckmann. Joystick Force Feedback based on Proximity to the Linearised Workspace of the Four-legged Robot ALDURO. 2011.

[6] Pedro Cruz. Haptic interface data acquisition system. Master's thesis, 2012.

[7] Rémi Sabino. Estrutura Híbrida de Locomoção para um Robô Humanóide. Master's thesis, 2009.

[8] Programmer's Guide. *OpenHaptics Toolkit version 3.0.* SensAble Technologies, Boston, USA, 2008.

[9] Emilio Estrelinha. Tele-operation of a humanoid robot using Haptics and Load Sensors. Master's thesis, 2013.

[10] Sensable Datasheet. Specifications for the PHANTOM Omni haptic device.

[11] Manuela M. Veloso Brenna D. Argall Brett Browning Manuela M. Veloso Brenna D. Argall Brett Browning Manuela M. Veloso Brenna D. Argall, Brett Browning. Learning Mobile Robot Motion Control from Demonstration and Corrective Feedback. 2009.

[12] Pedro Cruz, Vítor Santos, and Filipe Silva. Tele-Kinesthetic Teaching of a Humanoid Robot with Haptic Data Acquisition. 2012.

[13] Carlos A. Acosta Calderon and Rajesh E. Mohan. Teaching new tricks to a robot learning to solve a task by imitation. In *2010 IEEE Conference on Robotics, Automation and Mechatronics*, pages 256–262. IEEE, June 2010.

[14] Zeus; Robotic Surgical System.

[15] L. Adhami and E. Coste-Maniere. Positioning tele-operated surgical robots for collision-free optimal operation. In *Proceedings IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 3, pages 2962–2967. IEEE, 2002.

[16] Hu Haiying, Li Jiawei, Xie Zongwu, and Wang Bin. A robot arm/hand teleoperation system with telepresence and shared control. *Proceedings, 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics.*, pages 1312–1317, 2005.

[17] Andrew Y Ng Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler. ROS: an open-source Robot Operating System, 2009.

[18] ROS Nodes. `http://wiki.ros.org/Nodes`.

[19] ROS topic. `http://wiki.ros.org/Topics`.

[20] ROS Rviz. `http://wiki.ros.org/rviz`.

[21] ROS Bags. `http://wiki.ros.org/Bags`.

[22] J Kenneth Salisbury Thomas H Massie. The phantom haptic interface: A device for probing virtual objects. *Proceedings of the ASME winter annual meeting, symposium on haptic interfaces for virtual environment and teleoperator systems*, 55(1):295–300, 2011.

[23] Haptic Devices Phantom. *API Reference Manual.*

[24] Miguel Ribeiro. Desenvolvimento dos Sistemas Sensorial e Motor Para Um Robô Humanóide. Master's thesis, 2010.

Appendix A

Layers of the amplification shield

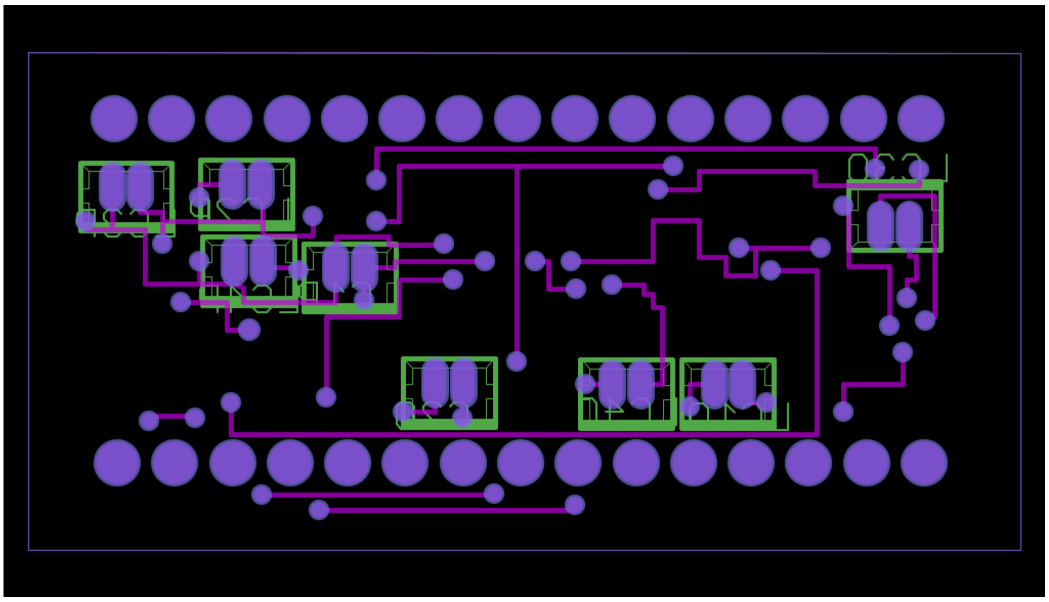


Figure A.1: Bottom layer of the amplification shield

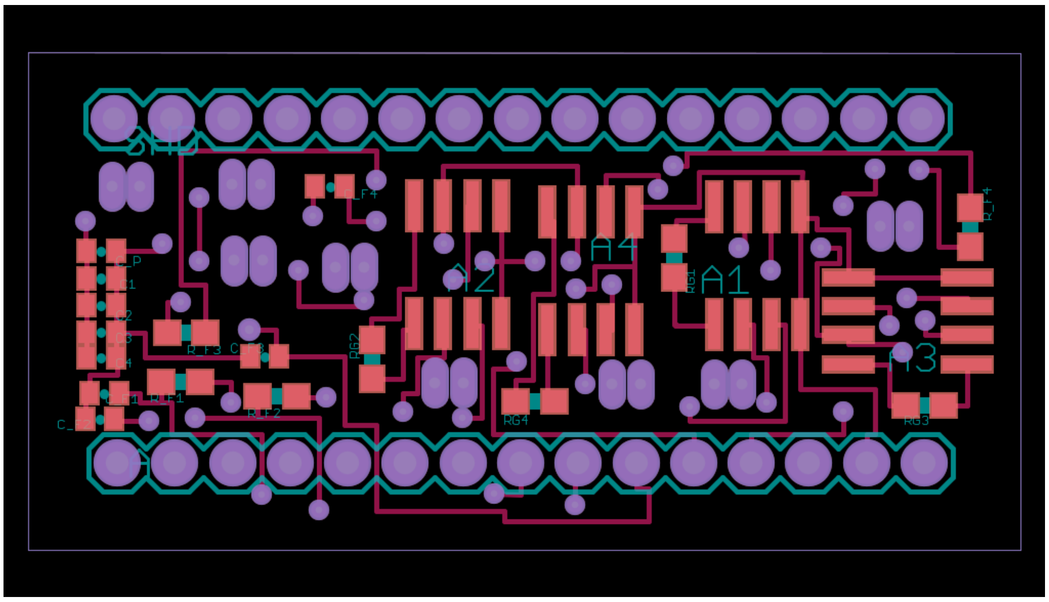


Figure A.2: Top layer of the amplification layer

Appendix B

Arduino Nano's Schematic

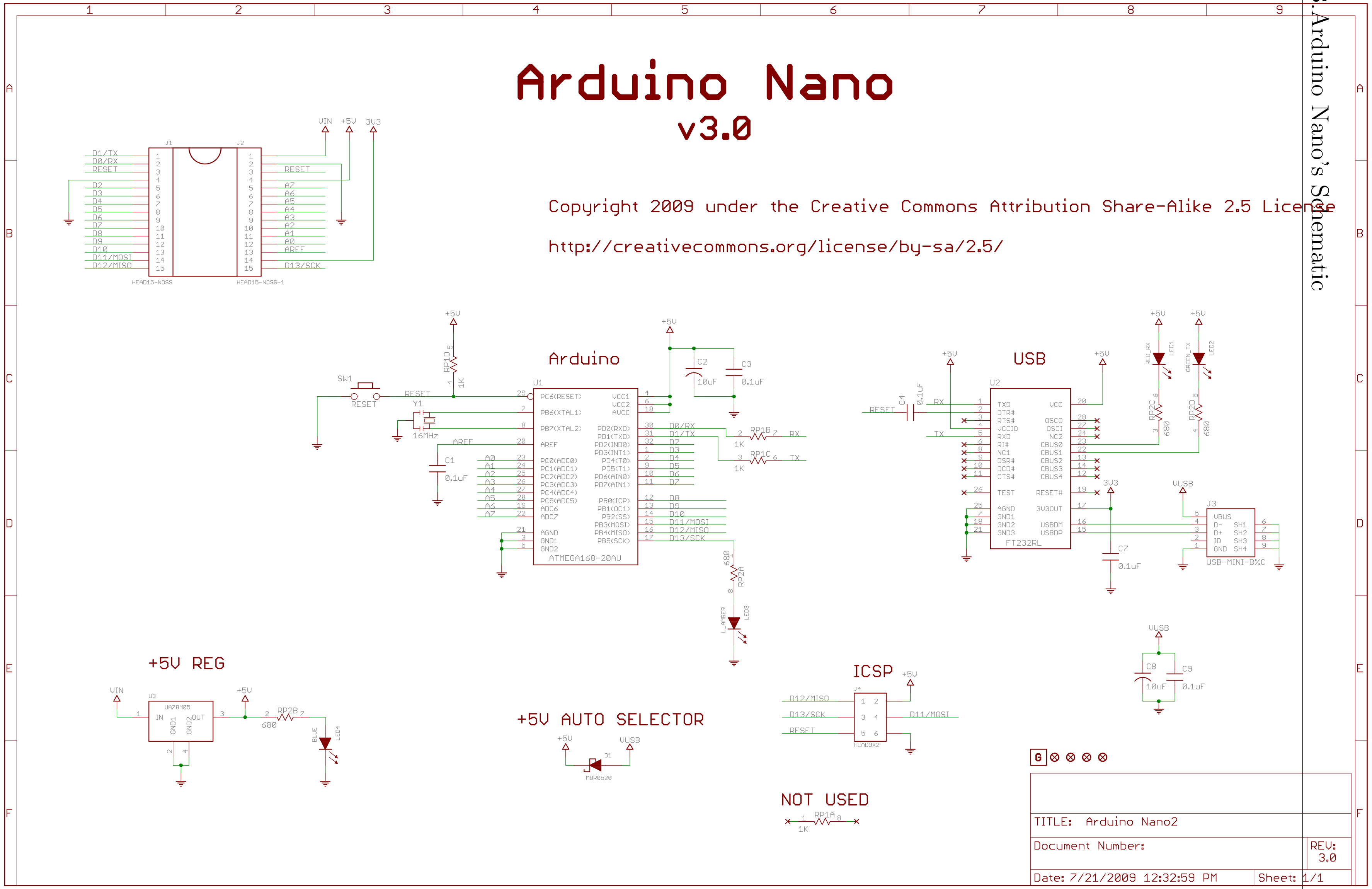


Figure B.1: Arduino Nano Schematic