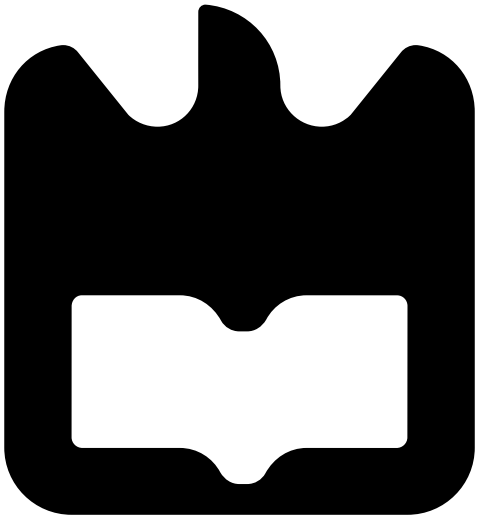
Universidade de Aveiro Departamento de Engenharia Mecânica
2015

Daniel Filipe da Ponte Marques

Desenvolvimento de Ferramentas de Treino Para Teleoperação Háptica de Um Robô Humanóide.

Development of Training Tools for Haptic Teleoperation of a Humanoid Robot.



Universidade de Aveiro Departamento de Engenharia Mecânica
2015

Daniel Filipe da Ponte Marques

Desenvolvimento de Ferramentas de Treino Para Teleoperação Háptica de Um Robô Humanóide.

Development of Training Tools for Haptic Teleoperation of a Humanoid Robot.

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

O júri / The jury

Presidente / President	**Prof. Doutor José Paulo Oliveira Santos** Professor Auxiliar da Universidade de Aveiro
Vogais / Committee	**Prof. Doutor Vítor Manuel Ferreira dos Santos** Professor Associado da Universidade de Aveiro (orientador)
	Prof. Doutor João Paulo Morais Ferreira Professor Adjunto do Instituto Superior de Engenharia de Coimbra

Agradecimentos / Acknowledgements

Em primeiro lugar gostaria de agradecer ao Professor Vitor Santos e ou Professor Filipe Silva por me terem fornecido a oportunidade de trabalhar neste projeto, como também por todo o apoio e incentivo ao longo do percurso que me incentivou perante todas as dificuldades encontradas. Um especial agradecimento aos meus colegas e amigos do Laboratório de Automação e Robótica pela constante ajuda na dissertação, tanto na resolução de problemas como na disponibilidade em serem cobaias dos mecanismos desenvolvidos. Não posso de deixar de agradecer aos restantes amigos e colegas presentes na minha vida, sempre me forneceram o apoio necessário e desnecessário ao longo deste longo curso. Presentes nos bons e maus momentos sempre ao meu lado a empurrar para horizontes maiores. Um muito obrigado! Por fim mas não menos importante, um grande agradecimento à minha família, sem ela simplesmente não estaria onde me encontro hoje.

Palavras-chave

Haptics; interface haptica; orientação haptica; orientação visual; memória cinestésica; robôs humanóides; interfaces de treino, force feedback, teleoperação cooperativa

Resumo

Em robótica, a teleoperação de robôs bípede humanóides é um dos tópicos mais emocionante. Tem a possibilidade de contornar modelos dinâmicos rígidos, com algoritmos de aprendizagem por demonstração utilizando interação humana. Para este procedimento, o Projeto Humanóide da Universidade de Aveiro - PHUA, empanha-se na produção de uma plataforma humanóide de corpo inteiro teleoperado com dispositivos hapticos. O estado presente do projeto apresenta um robô humanóide com 27 graus de liberdade. O projeto actual apresenta um modelo do robô importado para a *Virtual Robot Experimentation Platform*: V-REP. O uso do simulador permite vários exercícios com maior velocidade e tempos de preparação curtos, quando comparado com a teleoperação do robô real, além de proporcionar mais segurança para a plataforma e do operador durante os ensaios. Ao utilizar o simulador, o utilizador pode realizar testes à reprodução de movimento humano com a interacção de dois dispositivos de meios hápticos que fornecem força de retorno para o operador. As manobras realizadas têm os seus dados cinemáticos e dinâmicos armazenados para posterior aplicação na aprendizagem por algoritmos de demonstração. No entanto, a produção de movimentos mais complexos e detalhados requer grandes quantidades de habilidade motora do operador. Devido à mudança contínua de usuários no PHUA, um período de adaptação é necessário para os operadores recém-chegados a desenvolver uma afinidade com o complexo sistema de controlo. Este trabalho é focado no desenvolvimento de metodologias para diminuir o tempo necessário para o processo de formação dos utilizadores. Graças à versatilidade de personalização fornecidos pela V-REP, foi possível implementar interfaces que utilizaram orientação visual e haptica para melhorar as capacidades de aprendizagem do operador. Uma estação de trabalho, novas formulações e ferramentas de apoio que controlam a simulação foram desenvolvidos a fim de criar um controle mais intuitivo sobre a plataforma humanóide. Os operadores foram instruídos a reproduzir movimentos complexos em 3D sob diversas condições de treino (com feedback visual e haptico, apenas feedback haptico, apenas feedback visual, com ferramentas de orientação e sem orientação). O desempenho foi medido em termos de velocidade, a desvio de trajectória pretendida, a resposta à desvio e o tempo gasto para a criação do movimento. Os resultados deste estudo indicam que, com os mecanismos recém-implementadas, os operadores são capazes de ganhar o controlo sobre a plataforma humanóide dentro de um período relativamente curto de treino. Operadores submetidos a programas de orientação apresentam um período ainda mais curto de formação necessária, exibindo alto desempenho no sistema global. Estes fatos justificam o papel da orientação haptica em adquirir memória cinestésica em sistemas DOFs elevados.

Keywords Haptics; haptics interface; haptic guidance; visual guidance; kinesthetic memory; humanoid robots; trainer interfaces; force feedback; cooperative teleoperation.

Abstract In robotics, the teleoperation of biped humanoids is one of the most exciting topics. It has the possibility to bypass complex dynamic models with learning demonstration algorithms using human interaction. For this procedure, the Humanoid Project at the University of Aveiro - PHUA, ingrained in the production of a 27 degree-of-freedom full body humanoid platform teleoperated by means of haptic devices. The current project also comprises a robot model that has be imported into the Virtual Robot Experimentation Platform: V-REP. The usage of the simulator allows multiple exercises with greater speed and shorted setup times, when compared to the teleoperation of the real robot, besides providing more safety for the platform and the operator during the tests. By using the simulator, the user can perform tests and make achievements towards the reproduction of human movement with the interaction of two haptic devices providing force feedback to the operator. The performed maneuvers have their kinematic and dynamic data stored for later application in learning by demonstration algorithms. However, the production of more complex and detailed movements requires large amounts of motor skill from the operator. Due to the continuous change of users in the PHUA, an adaptation period is required for the newly arrived operators to develop an affinity with the complex control system. This work is focused on developing methodologies to lower the required time for the training process. Thanks to the versatility of customization provided by V-REP, it was possible to implement interfaces which utilized visual and haptic guidance to enhance the learning capabilities of the operator. A dedicate workstation, new formulations and support tools that control the simulation were developed in order to create a more intuitive control over the humanoid platform. Operators were instructed to reproduce complex 3D movements under several training conditions (with visual and haptic feedback, only haptic feedback, only visual feedback, with guidance tools and without guidance). Performance was measured in terms of speed, drift from intended trajectory, response to the drift and amplitude of the movement. Findings of this study indicate that, with the newly implemented mechanisms, operators are able to gain control over the humanoid platform within a relatively short period of training. Operators subjected to the guidance programs present an even shorter period of training needed, exhibiting high performance in the overall system. These facts support the role of haptic guidance in acquiring kinesthetic memory in high DOFs systems.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Context and Motivation

One of the exciting topics in the field of robotics is the development of humanoid robots that are able to produce high degrees of motion with the versatility and response of the human's motor capacity. This topic was already addressed in many works in the past several decades, where there was already achieved several breakthroughs, but still, none was able to achieve a full potential humanoid platform.

One of the greatest challenges on a humanoid platform is the creation of a dynamic and responsive biped locomotion able to autonomously surpass obstacles on its way. The great difficulty of this issue is consequence of the diversified number of the large number of degrees of freedom (joints) present on each limb, and non-linearities of its kinematic chains. The refine control of both legs isn't the only problem, to produce a human like locomotion the whole body demands an refine control of all body joints for the required balance and stability for unforeseen external and internal forces.

To solve those issues it is proposed the application of robot learning by demonstration. With this tool the complex manual programming and complex dynamic models are automated by demonstrations via teleoperation, or by vision and motion detection of the robot state and environment conditions. These demonstrations provide accurate data models retrieved by expert human controllers using high-degree-freedom haptic interface.

This work is centered on the University of Aveiro Humanoid Project (PHUA). PHUA consists on a small scale full-body humanoid platform with 27 degrees-of-freedom that is controlled by PHANToM OMNI haptic devices. The project has some work done on the concept of robot learning from demonstration, where the operator teleoporates the robot with haptic devices in order to produce motion and balance maneuvers receiving feedback regarding the robot's state.

To forward the library of maneuvers produced by PHUA a more capable operator is gradually needed, so in junction with further developing the control and feedback of the platform, this work also produced haptic guidance tools that would be able to transfer the skill of the expert operators to the new operators so the project doesn't suffer due to the high value of time needed to achieve the same control previously obtained.

1.2 Objectives

The main approach of PHUA is to study, develop and implement mobility in humanoid robots, where the user provides physical demonstrations interacting with a system of haptics device. This methodology should provide a natural interaction for education and tele-kinesthetic sensitivity where the user provides guidance and functional fixes at the same time having a sense (that is able to "feel") the system dynamics, their physical abilities and/or restrictions . One of the approaches is to use the simulated body of the humanoid robot, while it is remote-controlled by a human using a haptic device, collect and record all the sensory information and control commands. However this approach suffers the restriction of the human remote-controlling the model has to have vast knowledge and control over the kinematic mapping between the DOFs haptic and humanoid to perform complex movements such as the human locomotion. This restriction prevents the continuous growth of the platform due to regular exchanges of users in the project, thus producing a requirement of a vast adaptation period to attain the knowledge and control necessary to reproduce the movements already achieved in latter works.

The development of an application that can successfully transfer the knowledge and control abilities from previous users to the new ones is a necessary tool to maintain the continuity in the development of the platform. This application should not only provide skill transfer but, also, provide immersion in the human/haptic interaction for an intuitive control. The datasets acquired for the demonstration algorithms would only be as useful as the performance of the operator retrieving them.

Acknowledgment of the vast time necessary to reach the same level of control in the platform, as in the previous work [1], inspires the study of the human/haptic interaction instead of the traditional haptic/robot interaction.

The main objectives of the work are listed below:

- Teleoperation of the V-REP model with one haptic device;
- Analysis of the fundamentals of the reproduction of maneuvers;
- Development, implementation and evaluation of a trainer interface;
- Extend the previous objectives to a two haptic devices configuration;
- Teleoperation of the V-REP in one leg balance task.

1.3 Thesis Guide

This work is divided in to six main chapters, their organization is as follows:

- Chapter 1 introduces a briefly explanation to the context and objectives of the current work.
- Chapter 2 exposes the state of the art review on haptic interaction, teleoperation, kinesthetic teaching, haptic guidance, as also the work performed in the PHUA until date.
- Chapter 3 presents the experimental setup involved in this work, introducing the haptic device, its software libraries, the dual configuration for the haptic devices implemented and the ROS framework revolving it. The V-REP main features are also explored in this chapter.
- Chapter 4 presents some of the existing methodologies supporting teaching of operators with haptics devices. Then it is presented the approach to the IK control configuration, the IK User Trainer Interface. It explores the visual guidance tool developed and the customized interface that is implemented to the platform. The new formulations to its force feedback is also explore in this chapter.
- Chapter 5 presents the approach made for the more complex, joint-by-joint control configuration, the Torque User Trainer Interface. It presents the dedicated workstation developed for a more intuitive handling of the haptic devices, as well the haptic guidance mechanism developed and implemented in the new interface. Auxiliary tools are introduced in the platform to facilitate the control over the model.
- Chapter 6 presents the experiments undertaken to test and validate the methodologies described in this work. Relevant data gather during the experiments are presented and discussed.
- Chapter 7 discuss the main conclusions of this dissertation work, and presents some proposals for future works.

Chapter 2

State of the Art

2.1 Principles of Haptic Interaction

The word *haptic* has its origin from the Greek word *haptikos*, which means to be able to grasp or perceive, therefore haptic interaction refers to the manipulation and sensing with our sense of touch incorporated in the process [29]. It was in the 20th century that the term haptic started appearing in psychologist's studies referred to active touch of real objects. In the 70s and 80s the robotics community started significant research efforts on manipulation and perception by touch, which gave birth to devices with dexterity inspired by human abilities.

Kinesthesia, also known as proprioception, is included in haptics, it translates to the ability to perceive ones's body position, movement and height, thus becoming common to collectively designate the sensory motor components of haptics as the "haptic channel". This because in certain anatomical parts, in particular the hands, perceiving the world and acting upon it are activities that take place together. Summarizing, the haptic channel contains tactile and kinesthetic information that are combined to provide humans with means to perceive and act on their environment [7].

With the sudden improvement of technology in the 1990's it was possible to create a new type of *haptics*. The interaction of several emerging technologies made possible the creation of the *virtualized haptics*, also known as the *computer haptics*, the new breed utilized interfaces of the virtual environment that the user would perceive by the sense of touch, to stimulate the sense the user is applied with force, vibration and/or motion. This mechanical simulation may be used to assist in the creation of computed virtual objects, that can be physically palpated and controlled, also enhancing remote control of machines and devices.

This new sensory display modality presents information by exerting controlled forces on the human hand through a haptic interface, rather than, as in *computer graphics*, via light from a visual display device. Unlike *computer graphics*, where audio and visual channels features unidirectional information and energy flow (form the system to the user), haptics interactions is bidirectional, with energy and information flowing from and toward the user. This is one of the most important features of the haptic interaction modality [5].

Figure 2.1 shows the subsystems and information flow underlying to the interactions between human users and force-reflecting haptic interfaces. Haptic interaction occurs between two dynamical systems, the haptic interface with a computer and the human

user with a central nervous system [6].

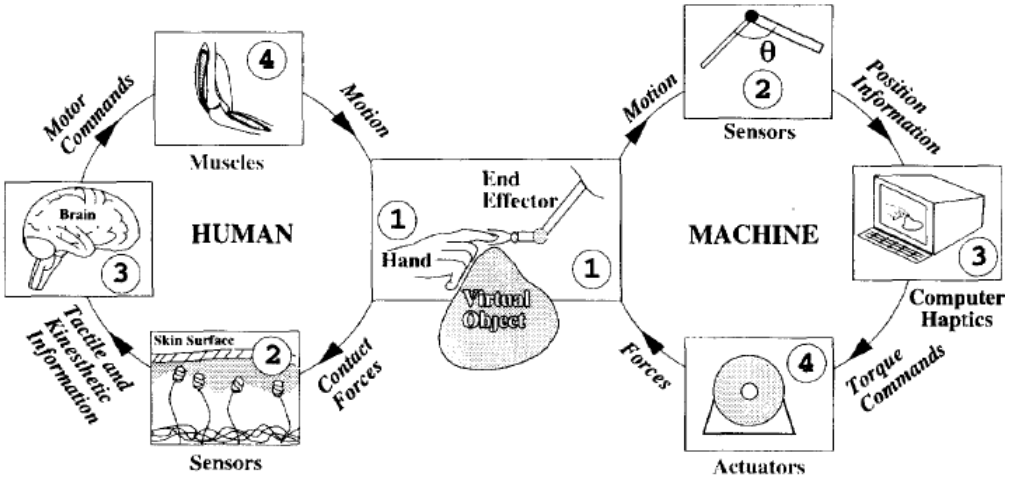


Figure 2.1: Haptic interaction between humans and machines [6].

- *Human sensorimeter loop*: when a human user interacts with a real or virtual object, forces are imposed on the skin. The associated sensory information is conveyed to the brain and leads to a perception of the object. After interpreting the environment, the brain motor commands activate the muscles, which results in hand and arm motion.
- *Machine sensorimeter loop*: when the human user manipulates the end-effector of the haptic interface device, the position sensors on the device convey its tip position to the computer. The models of objects in the computer calculate in real time the torque commands to the actuators on the haptic interface, so that appropriate reaction forces are applied on the user, leading to tactual perception of virtual.

The majority of haptics interaction are made in a virtual reality, which is a technology that allows the user to interact with computer simulated objects, or virtual objects, in real time, and under real or imaginary conditions. In addition to the haptic feedback, the virtual reality application also incorporates visual and audio feedback. This technology is presented in various applications such as 3-D simulators, medical and CAD development.

The application's main elements include:

- *simulation engine*, responsible for computing the virtual environment's behavior over time.
- *visual, auditory*, and *rendering algorithms*, which compute the virtual environment's graphics, sound, and force responses toward the user.
- *transducers*, which convert visual, audio, and force signals from the computer into a form the operator can perceive.
- *human operator*, who typically interacts (bidirectional) with the haptic interface device and perceives audio and visual feedback, from computer, headphones, visual displays, etc.

The force that is perceived by the user is synthesized through computer algorithms, in a process know as haptic rendering. This process allows the correction of forces between the interaction of the haptic interface representation inside the virtual environment and the objects populating in the environment, allowing the haptic device to correctly render such forces on the human operator. Figure 2.2 illustrates the main components of a haptic-rendering algorithm [5].

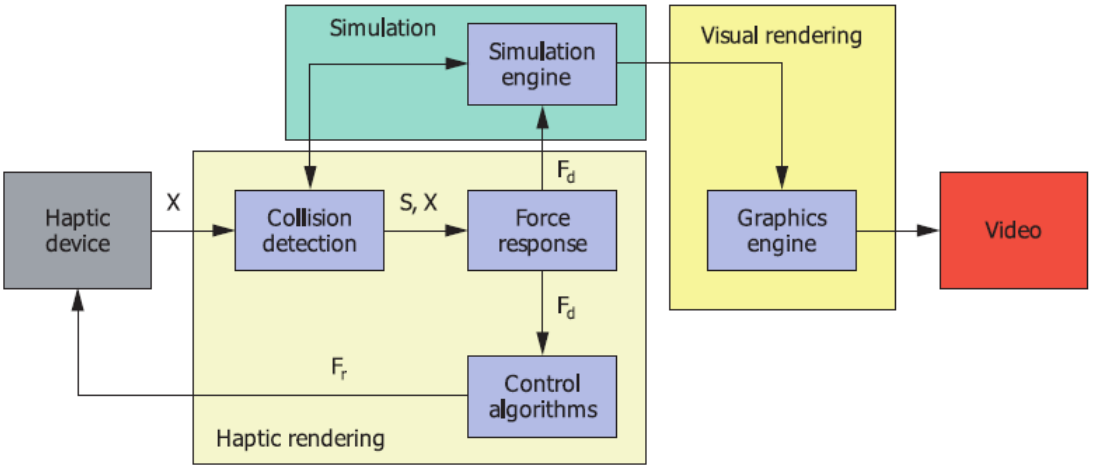


Figure 2.2: Hapitc rendering architecture split onto three manin blocks. *Collision-detection* algorithms provide information about contacts S occurring between an avatar at position X and objects in the virtual environment. *Force-response* algorithms return the ideal interaction force F_d between avatar and virtual objects, while *Control* algorithms return a force F_r to the user [5].

The *Collision-detection* algorithms are responsible for the detection of the collisions between objects and avatars in the virtual environment and for the processed information about the nature of those same collisions. The *Force-response* algorithms compute the interaction force between avatar and virtual objects involved in the collision, approximating this force as closely as possible to the contact forces that would normally exist during real contact. The force-response algorithm feeds the control algorithm with the desired force and torque vectors which it returns the actual force and torque vectors. These are then feed to the the inverse dynamics algorithm that is responsible for the proper conversion into device joint's torque.

For a smooth and natural haptic rendering, the algorithms should be present in a loop of at least 1kHz. Although this isn't a tabled value, it is very common and acceptable servo rate for a normal simulated textures. Higher servo rates can provide crisper contact textures sensation [8].

Nowadays, haptics can be found in a wide variety of applications, some example are the computer aided design /engineering/manufacturing(CAD/CAE/CAM), video games, surgical simulation and medical training [8].

2.2 Teleoperation, Kinesthetic Teaching and Learning from Demonstration

It is common to find teleoperation of mobile robots, including humanoid robots, with a haptic interface. The addition of the force feedback enabled from the haptic interface to the remote images and navigation sensor data provided by the usual application highly enhances the performance of the operator. For that cause there is, the study of modern telepresence systems, the synchronization of teleoperations/autonomous control reaction, among others. Small mobile robot have a broad set of sensors used for navigation and for characterization of the working environment. More specifically, the front of the car allocates a force sensor which measures the force between the car and an object and with that measurement the teleoperator's haptic joystick generates proportional forces while also commanding the vehicle [9] .

A teleoperated mobile robot with a haptic interface is shown on Figure 2.3. This example has the haptic device's joystick has the main input device for the operator commanding teleoperated robot. The device is connected by a PCI interface board to client's computer, enabling high data rate between the device and the program. To provide a highly reliable communication, an interface in object-oriented programming language C++ is implemented. To support the configuration the communication between the client and server computers is based on sockets, using the Internet user datagram protocol(UDP). Unlike the Internet standard communication protocol (TCP/IP), UDP presents a better real-time performance, by continuously send and receive the latest information possible, avoiding delays associated with the attempt of recovering lost information, greatly improving the synchronism of the telematic system [9].

The focus of learning from demonstration is to develop an intuitive and interactive method based on user demonstrations to produce human-like and autonomous behavior for a robot. By capturing the sophisticated operation of a human expert using high degree-of-freedom haptic interfaces, a user's skill can be stored, analyzed, and transferred to a robot or even other humans. At first, the human teleoperates the robot to perform a certain task. After the robot executes the commands from the human operator, learning algorithms, such as artificial neural networks (ANN) and support vector machines (SVM), are used to create learning modules over the temporal sequences of the trajectory. Afterwards, the learned modules control the robotic system, which will try to autonomously perform the trained task [12]. Skills transfer mechanism are rather common in robotics. The previous referred paper [12], had the goal to successfully recreate the operator's handwriting, and then transfer haptic forces to and untrained user, through a haptic device. The following Figure 2.4 illustrates an example of the letter "b" demonstrated twice for the robot.

Another application of skill transfer mechanism can be visualized in Figure 2.5. A user teleoperates a humanoid robot to demonstrate how to perform lifting tasks, collaboratively with another user [39]. In a first stage the user uses the PHANToM Desktop haptic device to control the robot, and afterward through observation. This process allows the robot to reproduce the overall dynamics of the task, more specifically the force patterns for the lifting of the object as the adaptation to the human user's hand motion.

Controlling a full-body humanoid is an extremely difficult task, especially if the robot is standing free on its legs. Physical human-robot interaction with humanoid has been studied in the context of assisted balancing or walking. The development of full potential

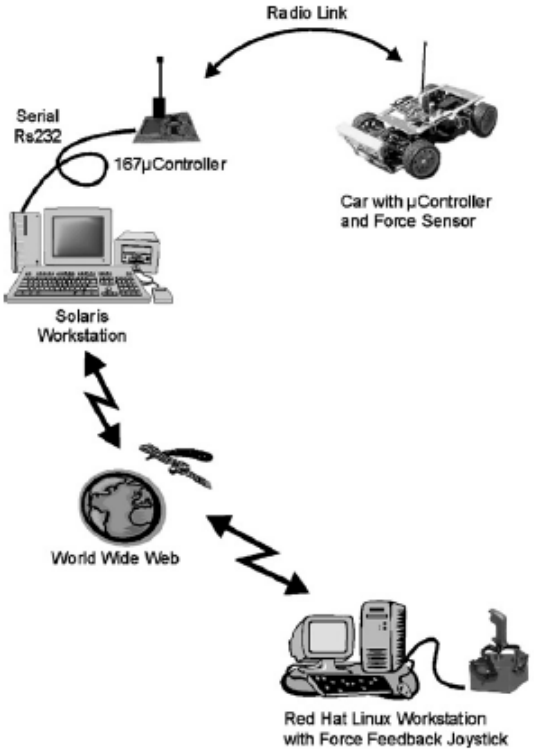


Figure 2.3: The telematic system of a teleoperated mobile robot [9].

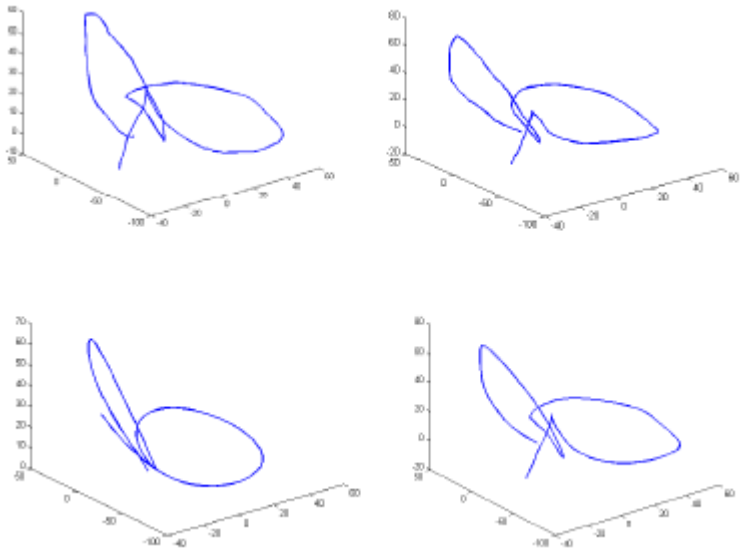


Figure 2.4: Letter "b": human writing patterns(top) and robot's patterns(bottom). Bottom left: result with NN learner, bottom right: results with SVR learner [12].

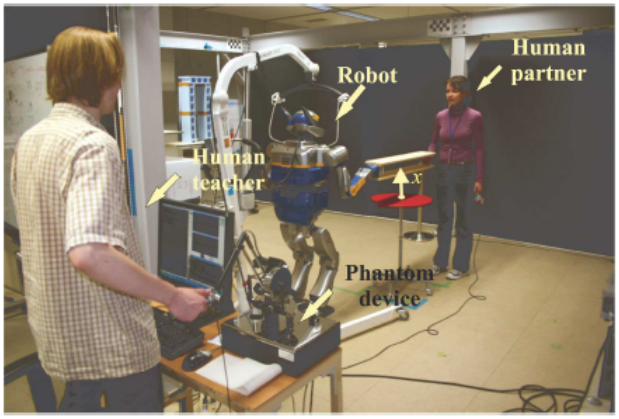


Figure 2.5: A user (on the left) teleoperates a humanoid robot to demonstrate how to perform a lifting task collaboratively with another user (on the right) [39].

humanoid platform depends largely on their ability to learn new tasks by themselves or to imitate human demonstrations of tasks. One experiment leads a free-standing humanoid robot to acquire new motor skills by kinesthetic teaching. Imitation leaning is used for training the upper body of the humanoid via kinesthetic teaching, while at the same time Reaction Null Space method is used for keeping the robot balanced [40]. Figure 2.6 illustrates the full experience, where in the first two frames a force/torque sensor records the exerted forces created during the demonstrations, while in the last frame of the figure the acquired forces values are utilized to produce a controller that applies the learned trajectories in terms of position and force to the end-effector.

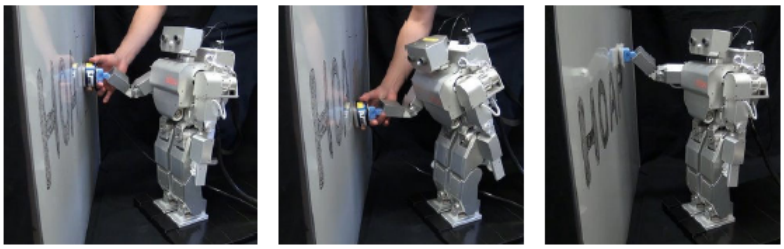


Figure 2.6: Upper-body kinesthectic teaching of a free-standing robot [40].

2.3 Methods and Applications of Haptic Guidance

The term *haptic guidance* refers to the application of haptics in skills training. In the haptic guidance paradigm, the suject is physically guided through the ideal motion by the haptic interface, thus giving the subject a kinesthetic understanding of what is required. Kinesthetic is crucial in haptic training, because it is one of the most powerful ways of perceiving incoming stimuli. Furthermore, kinesthetic memory, or the ability to remember limb position and velocity has proved to be very important in the haptic training. Humans have a remarkable ability to remember positions of their limbs quite

accurately and for long periods [11].

Previous studies submitted subjects to haptic guidance, in order to learn a complex 3-D motion under three training conditions (haptic, visual, haptic and visual combined), and afterward required to manually reproduce the movement under two recall conditions (with vision, without vision). They performance was measured in terms of position, shape, timing, and drift. The study indicated that haptic guidance is effective in training. While visual training was better for teaching the trajectory shape, temporal aspects of the task were more effectively learned from haptic guidance [10].

The new device developed for the Paralympics is a prime example of haptic guidance over visual feedback, when training for the highest levels of a sport requires fine-tuning of muscle memory, but this is often a visual game, with athletes watching high-speed video of themselves and their competitors to perform the right moves. So impaired vision can be a major obstacle. This device gives physical feedback instead - and it could even be used to help athletes emulate the movements of star athletes.

Figure 2.7: Ghost trainer armband [23].

The vibrating arm band, called Ghost, detects a user's arm motion and buzzes to give feedback. It would work by pre-loading the wristband's software with the right pattern of muscle movements, or even a certain athlete's exact swing. The pattern includes certain way points, which would indicate where your swing or stroke is going. Then the device would guide the athlete through the motions. Sensors can detect the joints' flexing and twisting, and vibrations and sounds indicate whether the user is doing it wrong or right. With each try, the athlete's muscle memory gets better, but there's no coach demonstrating how to do it and no hours of video to sit through. It could help vision-impaired and fully sighted athletes [23].

Haptic training system are present in the teaching of refined hand skills, such as calligraphy, using demonstration from and expert user to train the trainee. On the Expert's phase a pencil like haptic devices are use to record the expert's motion and force. The Trainee's phrase follows a combination of the following two methods [28]:

- *Opposite presentation of expert's force along trajectory*, the haptic device generates the force that has the same amplitude of the expert, but in opposite direction. The trainee tries to cancel the force and consequently, necessary internal force is "proactively" generated.

- *The utilization of Virtual Fictures orthogonal to the trajectory*, Information of the expert's motion is given (Figure 2.8) , and the pathway is used as ruler by using Virtual Fixtures technique.

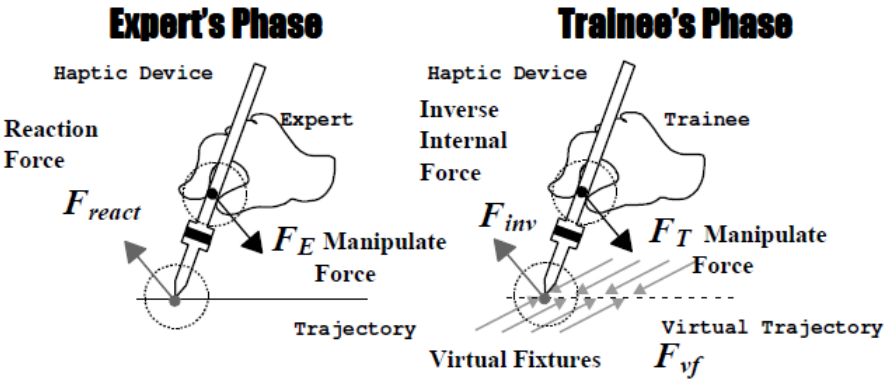


Figure 2.8: Proposed method [28]

2.3.1 Existing Methodologies for haptic training

Learning to perform new movements is usually achieved by following visual demonstrations. Haptic guidance by a force feedback device is a recent and original technology which provides additional proprioceptive cues during visuo-motor learning tasks [32]. This segment presents the several researches explored for the development of bout User Trainer Interfaces.

Haptic Guidance in Position(HGP)

Haptic guidance in position (HGP) mostly uses a proportional derivative controller i.e. following point-per-point the visual representation of the target trajectory. Based on this technology, Solis *et al.* [33] had developed a Japanese calligraphy system using reactive robot technology. Unfortunately, this study mainly focused on the technical aspects. In the same vein, Henmi *et al.* [34] also designed a Japanese calligraphy system using a "record and playback" strategy: The authors recorded positions and forces applied by a human teacher and displayed them to the students. However, in these two studies, no behavioral data was reported. Gillespie [35] developed a virtual teacher based on a proportional derivative position controller to help students to properly move a simulated crane. This pilot study with 24 participants showed that their implementation of the virtual teacher concept did not significantly improve the learning of oscillating curves. More recently, Palluel-Germain *et al.* [36], in a pilot study, analyzed the effects of using HGP to train the fluency of writing cursive letters in kindergarten children. Fluency of handwriting (analyzed by kinematics parameters such as average velocity, number of velocity peaks, and number of breaks during the production) was tested before and after the training sessions (either visuo-haptic or control). Letters were computer generated to control the dynamics by changing the distance between successive points of a discrete trajectory. Results showed that the fluency of handwriting for all letters was higher after

the visuo-haptic training session than after the control training session: The movements of the hand were faster, exhibited less velocity peaks and the children lifted the pen less frequently during handwriting. Finally, other studies in adults confirmed the positive effect of visuo-haptic training sessions using proportional derivative position controller but most of them mainly describe the technical aspects. In these studies, the analysis of kinematics criteria remained rather unexplored [32]. Figure 2.9 represents an example of an application of HGP in a stylus.

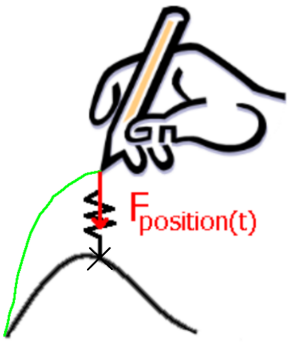


Figure 2.9: Haptic guidance in position (HGP); the force felt by the user at time t is proportional to displacement between the current user position and the theoretical position on the model trajectory (adapted) [32].

Haptic Guidance in Force(HGF)

Haptic guidance in force (HGF) is an alternative control method, which is congruent with two well-known psychophysical principles: The homothety principle which states that the trajectory keeps its shape characteristics whatever its size, and the isochrony principle which states that the velocity for tracing increases as function of its size. Hemni *et al.* [34] compared the effects of HGF and HGP on a handwriting task in addition to visual cues. Preliminary results showed both techniques to be equally effective. Srimathveeravalli *et al.* [37] introduced a new paradigm providing the closest possible replication of an expert's skill. The authors proposed that if the nature of forces generated by the teacher and by the student were the same, then their trajectories would be similar. Force profiles of the teacher were then used to guide the motion of the student. Demonstration of its efficiency was shown by comparing this method with other classical haptic training methods in terms of shape matching with an unfamiliar Tamil (Indian) letter. Results confirmed the authors' hypothesis and showed that a "record-and-playback" training method with force information was more efficient than training method with only position control. Unfortunately, this study mainly focused on a shape matching score and did not examine kinematics criteria. Recently, Morris *et al* [38] explored the use of haptic feedback for teaching a sequence of forces. Results showed that adults are able to learn sensory-motor skills via visuo-haptic training. This result would allow us to better understand the positive effects of HGF during training session of handwriting observed by Srimathveeravalli [37] [32]. Figure 2.10 represents an example of an application of HGF in a stylus.

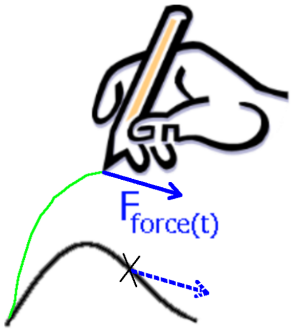


Figure 2.10: Haptic guidance in force (HGF); the force felt by the user at time t is the same as the force existing for the theoretical trajectory at the same time (adapted) [32].

Remote Haptic Collaboration

A lumbar puncture (LP) is a diagnostic and at times therapeutic procedure that is performed to collect a sample of cerebrospinal fluid for biochemical, microbiological, and cytological analysis. Lumbar punctures may also be done to inject medications into the cerebrospinal fluid, particularly for spinal anesthesia or chemotherapy. While inserting the lumbar needle into the epidural space between the vertebral bones of the lower back and into the fluid filled spinal canal, the physician must be greatly careful so as to not perforate the dura matter, a tough fibrous layer that covers and protects the spinal cord and its nerves, as this would result in complications such as spinal headache and the cerebrospinal fluid (CSF) leak that can lead to life threatening infections on rare occasions. Currently, in medical school, the students gain experience by performing the procedure on real patients, while a trained teacher stands by and instructs medical students as they perform the procedure. Although successful lumbar puncture skill can be learned through that method, the disadvantages are obvious especially for the patient, because unnecessary pain, tissue damage, or injury would be caused by the student's incorrect operation.

Using a PHANToM device produced by SensAble Technologies, which is capable of six input degrees of freedom and three output degrees of freedom. This device allows the user to touch and manipulate virtual objects. Therefore the so-called "loss of resistance" effect can be well simulated. The force feedback is calculated on a local PC, with which the haptic device is connected via the parallel port and the output occurs immediately to the device. When the medical teacher manipulated their local PHANTOM Desktop, the force was transmitted through the network to a remote PC terminal. Then, the training student could feel the "loss of resistance" effect via their local PHANTOM Desktop. In the same way, the incorrect operation of the training student can be immediately corrected by the remote teacher. Thus, users at both ends of the network could experience remote haptic interaction and collaboration.

The architecture of our system is shown in Figure 2.11. The physical simulator is to provide a physics-based simulation of the interaction between the needle and virtual patient. The results of physical simulation are displayed through graphic rendering and haptic rendering. For the purpose of realistic haptic feeling, the network controller is designed to acquire stable output to mitigate or eliminate the adverse effects of unpre-

dictable network delays, packet loss and out-of-sequence packets [27].

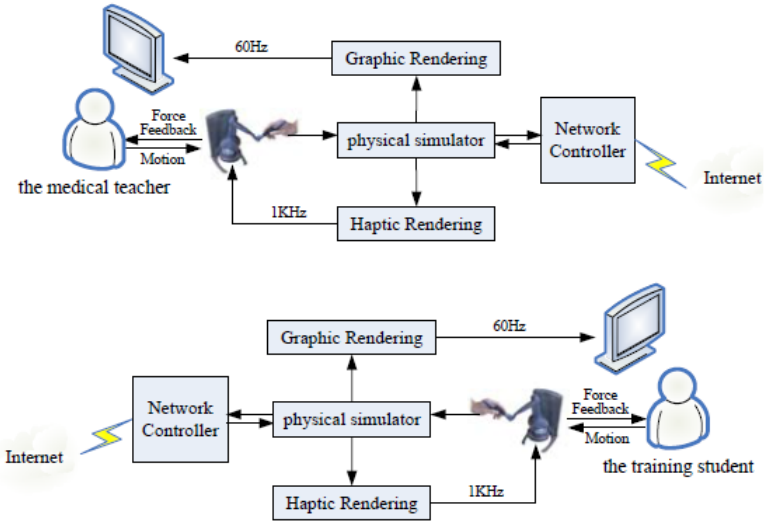


Figure 2.11: The system architecture [27].

2.4 University of Aveiro Humanoid Project(PHUA)

Started in the lecture year of 2003/2004 in the University of Aveiro, the PHUA is a collaboration between the Department of Mechanical Engineering (DEM) and the Department of Electronics, Telecommunication and Informatics (DETI), to create a full body humanoid robot. The PHUA project has the main objective to development and integration of hardware and software components in a functional low-budget platform, to execute studies in balance and locomotion tasks. The up to date robot's physical body is shown in Figure 2.14 .

The platform obtain its first finished model in 2008, with several thesis and published papers towards it. The current model has suffered alterations from the initial model, in 2009 it was development a new small-size platform with hybrid actuation has it is seen in Figure 2.14, with actuators and sensors installed as new mechanical components.

The robot replicates a full body humanoid platform with a total of 27 DOF, where 25 of those are active DOF and the remaining 2 are passive. The actuation on the joints is made by HITEC analog and digital servomotors, combined with elastic elements, belts and pulleys are used on the legs and torso for adjustable transmission system providing torque to the joints with higher requirements. Apart from the motor components, the robot is equipped with a set of 8 force sensors, distributed 4 for each foot, which are responsible for collecting data for balance demonstrations [2].

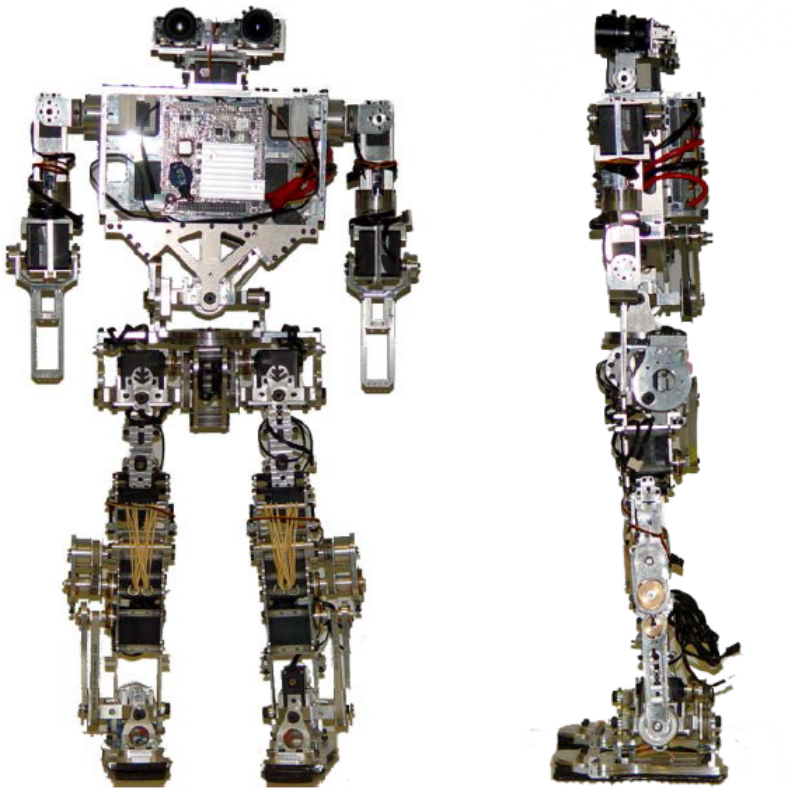


Figure 2.12: Front and side views of the PHUA platform's full body [13].

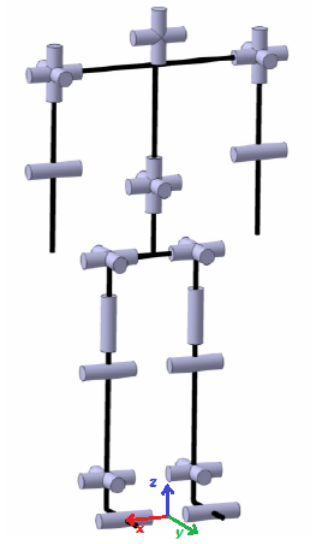


Figure 2.13: PHUA kinematic chains [2]

Joint	Joint's DOF
Toe	1(×2) passive
Ankle	2(×2) active
Knee	1(×2) active
Hip	3(×2) active
Trunk	3 active
Shoulder	3(×2) active
Elbow	1(×2) active
Neck	2 active
Total	25 active 2 passive

Table 2.1: PHUA's DOF

This platform is capable of identifying and following objects with its head and perform several exercises with its legs or arms, while presenting anthropometric proportions similar to that of an human body. These features allow for the robot's foot kinematics

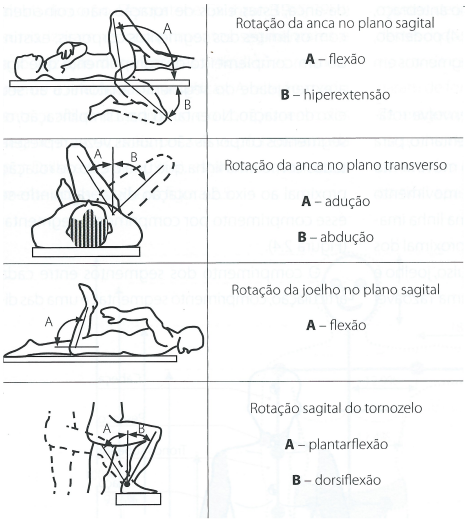


Figure 2.14: Representation of the movements about the legs [41].

to come closer of the natural gait of the human. The Table 2.2 details the robot's joints anthropometric ranges with the respective total angular course, the data is measured from its home position (in degrees), this position is defined with the limbs at full stretch downwards, with the torso upright and a zero tilting in the head [2] [3].

Table 2.2: PHUA's DOF range limitations

DOF	Minimum	Maximum	Total Range
Head Tilt	0°	10°	10°
Shoulder Flexion/Extension	−45°	135°	180°
Shoulder Abduction/Adduction	0°	180°	180°
Elbow Flexion/Extension	0°	120°	120°
Torso Rotation	−90°	90°	180°
Torso Flexion/Extension	−15°	90°	105°
Torso Lateral Flexion/Extension	−90°	90°	180°
Hip Extension/Flexion	−30°	120°	150°
Hip Adduction/Abduction	−40°	45°	85°
Knee Extension/Flexion	0°	130°	130°
Ankle Inversion/Eversion	−45°	30°	75°
Ankle Extension/Flexion	−20°	40°	60°

The later works on the project introduced the concept of telekinesthetic teaching by haptic teleoperation, utilizing learning from demonstrations algorithms in junction with haptic control. The data gathered from the demonstrations, through the haptic interaction, can be later used in the robot's learning ability through demonstration algorithms, producing the possibility to create complex and diverse maneuvers without the need of complex and stiff dynamics models. The haptic interface allows the operator to perceive the perturbations of the robot state by feeling it in the force feedback, this setup allows a dynamic and fluid response to the several robot's states.

The control methodologies and data gathering possibilities are currently being tested

for later use in learning procedures. The last work on the projected introduced a simulation of the robot in the software Virtual Robot Experimentation Platform (V-REP) that contemplated a full body rendering of the robot with the respective mass and joint placement. Figure 2.15 illustrates the full model of the robot in the simulator.

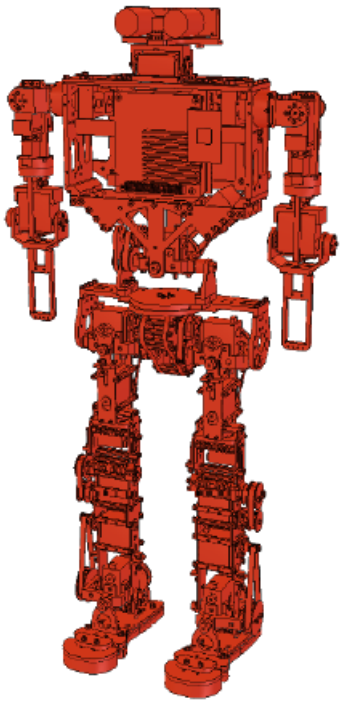


Figure 2.15: Visual appearance of the V-REP model [1].

With the new simulated PHUA it was possible to test some new control methodologies for the leg system. The previous work [1] implemented two distinct method, the Inverse Kinematics Mode (IK mode), illustrated on Figure 2.16 and the Torque mode, illustrated on Figure 2.17. With the IK mode the operator controls the position of the waist with the end-effector of the PHANToM Omni haptic device, moving the rest of the legs system by inverse kinematics, this methodology implied that the feet would be stuck in the same position. The correspondence between the global position of the device and V-REP is as follow:

$$\begin{bmatrix} x_{vrep} \\ y_{vrep} \\ z_{vrep} \end{bmatrix} = \begin{bmatrix} -x_{phantom} \\ y_{phantom} \\ z_{phantom} \end{bmatrix}, \tag{2.1}$$

with the inverse kinematics calculations being entirely executed by V-REP.

With the Torque Mode the operator controls each individual leg through a configured kinematic mapping between the device's and the humanoid's joints. each joint would assume the configuration of the respective joint of the PHANToM Omni device as is shown in the Figure 2.17. Due to the complexity of controlling six DOFs simultaneously the feet's DOFs were configured to assume a position relatively to the other joints, instead

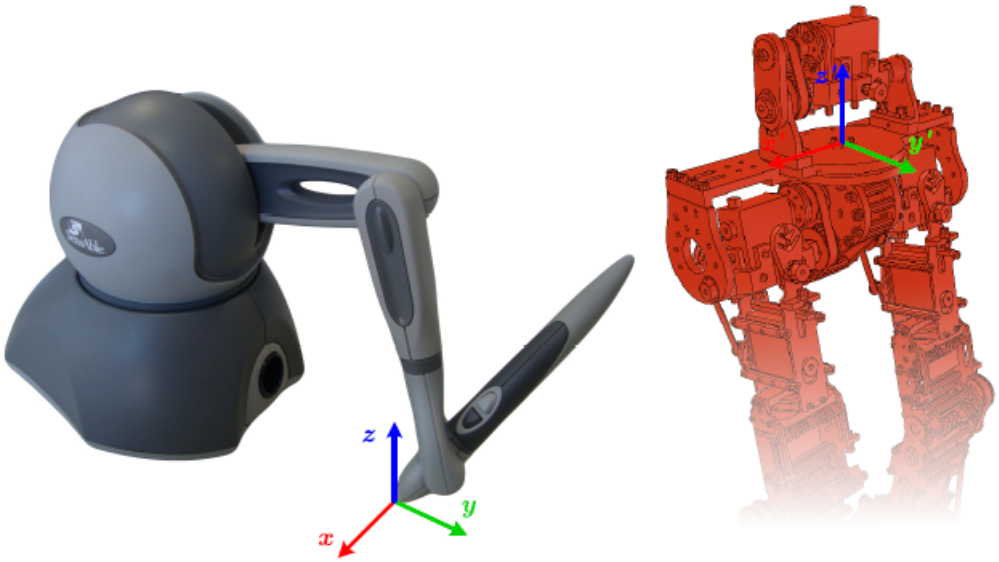


Figure 2.16: Inverse kinematics correspondence, for model teleoperation [1].

of a direct control from the user. The leg's yaw rotation was fixed since the real robot presents the DOF blocked as well.

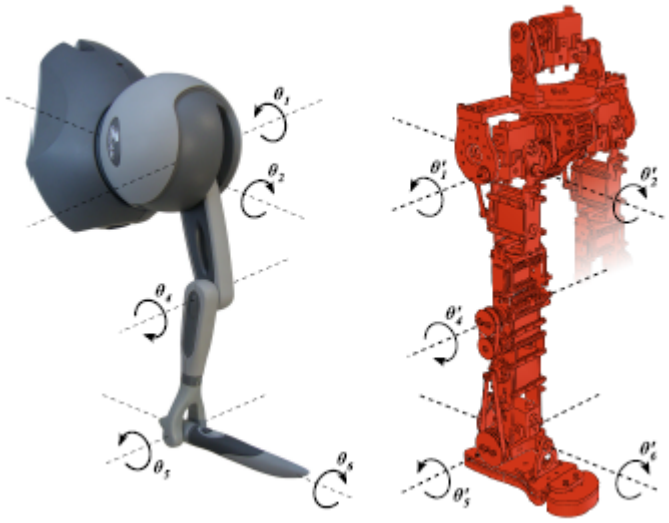


Figure 2.17: Torque Mode joint correspondence [1].

The kinematic mapping between the two models are as follow:

$$\begin{bmatrix} \theta_{1,vrep} \\ \theta_{2,vrep} \\ \theta_{4,vrep} \\ \theta_{5,vrep} \\ \theta_{6,vrep} \end{bmatrix} = \begin{bmatrix} \theta_{1,phantom} \\ \theta_{2,phantom} \\ \theta_{4,phantom} \\ \theta_{4,phantom} - \theta_{1,phantom} \\ -\theta_{2,phantom} \end{bmatrix} \tag{2.2}$$

The default joint configuration of the Omni device didn't coincide with adopted convention, so the device's joint space suffered a transformation to implement the represented convention represented in Figure 2.17. Equation 2.3 describes the transformation applied to the device's joints [1]:

$$\begin{bmatrix} \theta_{1,phantom} \\ \theta_{2,phantom} \\ \theta_{4,phantom} \end{bmatrix} = \begin{bmatrix} \theta_{2,default} \\ \theta_{1,default} \\ -\pi/2 + \theta_{2,default} - \theta_{3,default} \end{bmatrix} (rad) \tag{2.3}$$

In the most recent work [1] on the platform it was implemented a formulation for a force feedback that could transmit the unstableness of the robot state. This formulation utilized the position of the CoP relative to the robot's stability point (origin point of the feet coordinates system), and its distance to the robot's support base boundary. This formulation created a bi-dimensional vector force that is implemented on the end-effector of PHANToM Omni [1].

Chapter 3

Experimental Setup

In this chapter, the main software and hardware solution are presented. Initially there is a description of the hardware chosen for the haptic device as also the respective library, referencing their limitations and applications, following with a introduction of the simulator used to replicate the robot model. At last, the programming platforms used to develop the application are explained with a briefing of the current setup.

3.1 PHANToM OMNI Haptic Device

In the present, the world of haptic devices grows vast, but within that world the former SensAble's hapitc devices, now known as Geomagic's hapitc devices, are wide popular among the researchers of the haptic community. The devices can be found in application such as robotic control, medical simulations and skills assessment [16].

The haptic device used in this project was the SensAble PHANToM Omni,although the current model is designed as Geomagic Touch but since its presents differences with the used model it will be refer as its former label. Figure 3.1 shows the model used in the project and current model on the market.

Figure 3.1: SensAble PHANToM Omni(Left), Geomagic Touch(Right) [17] [30].

Constructed of durable metal and injection-molded plastics, PHANToM Omni offers six-degree-of-freedom positional sensing. Additionally, it allows a high degree of flexibility

Table 3.1: PHANToM Omni device specifications [17].

Worksapce	160 W x 120 H x 70 D mm
Range of motion at wrist	Hand movement pivoting
Nominal position resolution	> 450 dpi 0.055 mm
Maximum exertable force and torque at nominal position (orthogonal arms)	0.75 lbf/3.3
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
Force feedback (6 Degrees of Freedom	x, y, z
Interface	IEEE 1394 FireWire® port: 6-pin to 6-pin

with a portable design, compact footprint, removable stylus and two integrated buttons. The Omni model connects to the computer station via IEEE 1394 FireWire port .

Although the Omni model possesses six DOF, only the first three joint are actuated, meaning that its only possible to generate torque in the joints responsible for the translational component of the device, leaving the last three joints, responsible for the orientation, unable to render torque on them. This inability is due to its construction, the device is built with a capstan drive actuating two out of the three base DOF, allowing for its compact form and good force feedback level on its actuated workspace, coupled with high precision optical encoders. The rendered force feedback on the axis x,y and z relies on the measurement of instantaneous position and velocity acquired by the joints encoders. As for the last three DOF, the joints are passive, equipped with potentiometers to track the device's orientation [17]. The following Table 3.1 details the specifications of the Omni model.

All of the six revolute joints have their own mechanical limitations, when reacted, a lock will block the joint preventing forward rotation. The workspace on the Omni model is relative small, so when on its limitations proceed with caution, any transpassing of the same would cause damage to the device.

As mentioned before the first three joints of the Omni model give translational movements for the end-effector while the remain three provide the rotational movements or Euler angles. Due to this configuration the model resembles to a 6 DOF manipulator, more accurately a 3-bar Revolute-Revolute-Revolute (RRR) spatial manipulator. The kinematic diagram is shown in Figure 3.2.

To allow knowing the human operator performance and his representation in the virtual simulation, its required the forward and inverse kinematic models of the PHANToM Omni device. However, for the completion of this work, the kinematics and dynamic equations of motion for the device weren't needed so they wont be presented in the document. For an approach to explicit mathematical models and alternative dynamic considerations, refer to [16; 20].

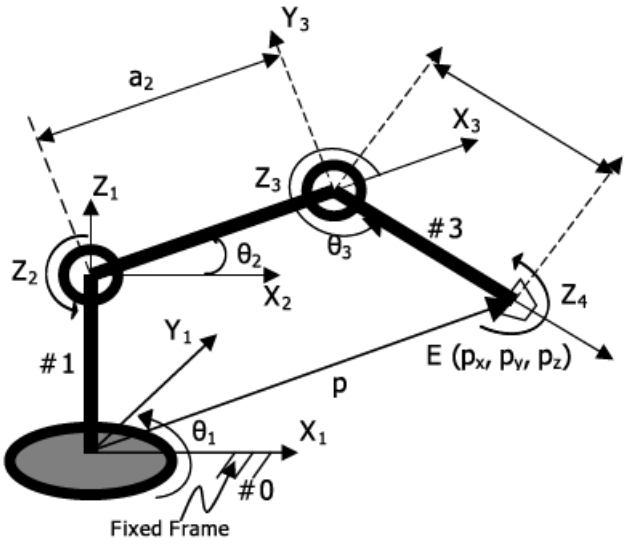


Figure 3.2: Kinematic diagram of PHANToM Omni [16].

The software provided by SensAble, described in the next segment, is well structured and provides all the means necessary to implement the device's control.

3.2 OpenHaptics Toolkit

Offered by SensAble, OpenHaptics Toolkit is a proprietary programming solution developed in C/C++ to build haptically enabled applications. While its available for Windows, Linux and MacOS X, the lasted version, OpenHaptics 3.2.2, only came out for the Windows machines, leaving Linux and MacOS with the former version of the software, OpenHaptics 3.0.

The OpenHaptics Toolkit includes [18] [19]:

- QuickHaptics micro API;
- Haptic Device API (HDAPI);
- Haptic Library API (HLAPI);
- PHANToM Device Drivers (PDD);
- Utilities;
- Source code examples;
- Documentation.

This haptic toolkit handles complex calculations, provides low-level device control and supports different polygonal objects, material properties and force effects. Integration with OpenGL application is facilitated. The typical structure under OpenHaptics is depicted in Figure 3.3. Its usual implementation incorporates object visualization with haptic interaction on a virtual environment.

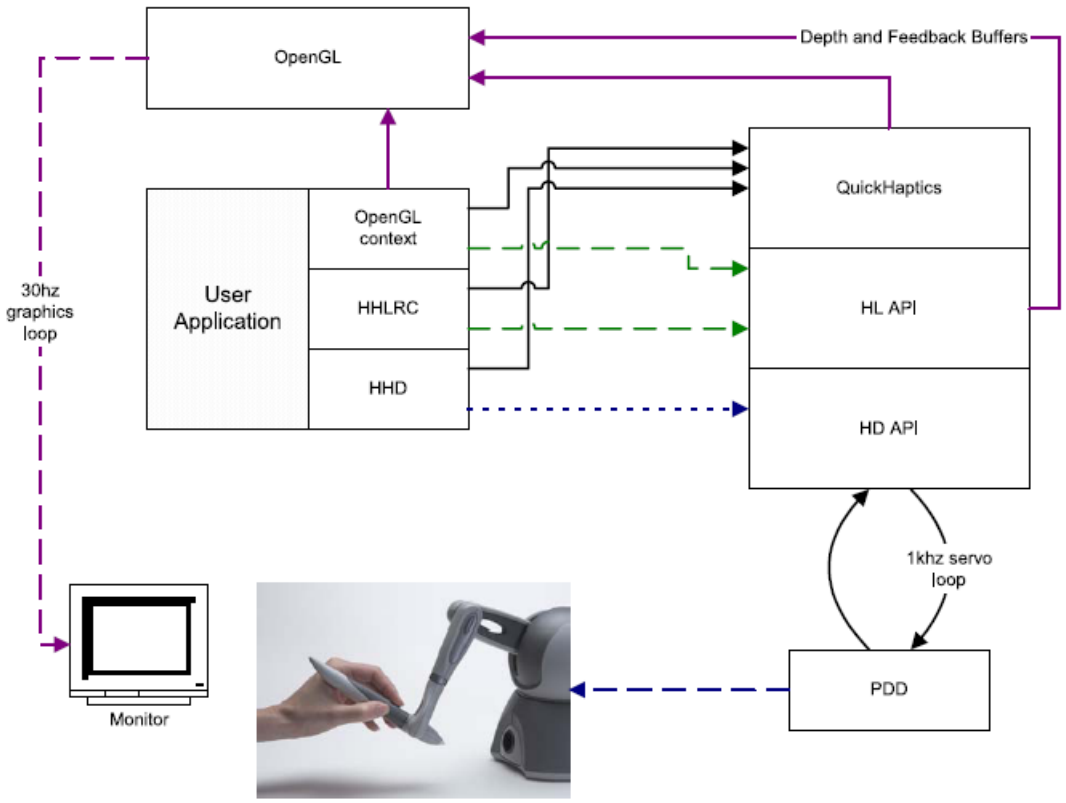


Figure 3.3: A typical OpenHaptics application structure [19].

QuickHaptics/HDAPI/HLAPI

OpenHaptics 3.0 allows to ease programming by containing the basic steps common to all haptics/graphics applications in the C++ classes of the QuickHaptics micro API, which makes use of the Standard Template Library (STL).

The following diagram illustrates the relationship between the QuickHaptics micro API and the existing HD and HL layers of OpenHaptics.

The primary functional classes defined by QuickHaptics API are: *DeviceSpace*, the workspace through which the haptic device can move; *QHRenderer*, on-screen window that renders shapes from a camera viewpoint and lets the user feel those shapes with haptic device; *Shape*, which is a base class for one or more geometric objects that can be rendered both graphically and haptically and *Cursor*, the graphical representation of the end point of the second link on the PHANToM device. These are the core elements of a haptic interaction.

The HDAPI s the foundational layer for haptics, and is is best suited for developers who are already familiar with haptic paradigms and sending forces directly. It provides low-level access to the haptic device, enables haptics programmers to render forces directly, offers control over configuring the runtime behavior of the drivers, and provides convenient utility features and debugging aids.

The HLAPI is designed for high-level haptics scene rendering. It is mainly targeted for advanced OpenGL developers, who are however less familiar with haptics programming,

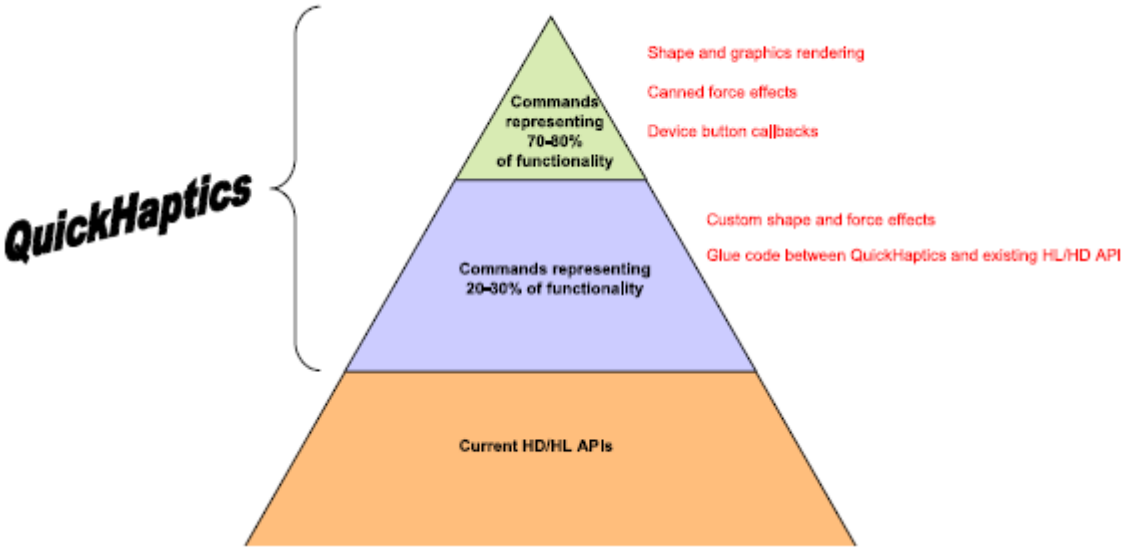


Figure 3.4: The OpenHaptics structure diagram [19].

but desire to easily incorporate haptics to existing graphics applications. Synchronization of the haptics and graphics threads is greatly simplified.

HDAPI requires the developer to manager direct force rendering for the haptic device, whereas HLAPI handles the computations of haptic rendering based on geometric primitives, transforms, and material properties. Thereby, HLAPI programmers will not have to concern themselves with such lower levels issues as designing force equations, handling thread safety, and implementing highly efficient data structure for haptic rendering. HLAPI also avoids the developer from managing the synchronization of servo loop, unlike HDAPI users. Direct force rendering with HDAPI requires efficient force rendering/collision detection algorithms and data structures. This is due to the high frequency of force refreshes, required for a stable closed-loop control of the haptic device. The servo loop refers to the tight control loop used to calculate forces to send to the haptic device. In order to render stable haptic feedback, this loop must be executed at a consistent 1 kHz rate or higher as mention previously. In order to maintain such a high update rate, the servo loop is generally executed in a separate, high priority thread. This thread is referred as the servo loop thread.

For the context of this work, the main API used is the HDAPI. Since the aim of the project is to use the PHANToM Omni haptic device, to generate a force feedback for the user, in response to a particular state of the robot and to read raw encoder and motor joint torque values. The collision detection algorithms and physics/dynamics calculations will be carried out by a third-party software, the V-REP simulator, in the simulation context.

PHANToM Device Drivers (PDD)

To able a proper communication with the computer, a proper software driver form the OpenHaptics PDD must be install. Due to its long and tricky installation procedures, the relevant steps are describe in detail in Appendix C, for future reference.

In context of this work, a dual PHANToM Omni configuration is implemented, to co-teleoperate the PHUA model, on this initial stage, and then the robot itself.

According to the proprietary user's guide [17], paired configuration can be set up to make possible a pair of Omni devices to work in tandem with each other. Although, with the previous work on the project [1] , it was concluded that the paired configuration isn't possible on Linux. This is due to the old and inappropriate Linux drivers for PHANToM Omni device.

3.3 Virtual Robot Experimentation Platform, V-REP

The V-REP, Virtual Robot Experimentation Platform, is a relatively new robot simulator with integrated development environment, containing incredible features, lots of function and elaborate APIs. The robot simulator V-REP is based on a distributed control architecture, which means each object/model can be individually controlled via an embedded script, a plugin, as ROS node, a remote API client, or a custom solution. These programming approaches are also mutually compatible and can even work hand-in-hand, what makes the software very versatile and ideal for multi-robot, and multi-function applications. Controllers can be written in C/C++, Python, Java, LUA, Matlab, Octave or Urbi [14].

V-REP possesses and integrated script interpreter and elaborate API functions, allowing the user to customize almost every aspect of the simulation. According to the user manual [15], the software main structure can be composed in three primary elements: *simulation controllers*, *scene object* and *calculation modules*.

3.3.1 Simulation controllers

Building complex simulation scenarios involves, almost certainly, a distributed control framework. It simplifies the task by partitioning control entities, speed-up simulation by distributing the CPU load over several cores or several machines and it allows a simulation model by controlled by native code execution.

The execution of the control code of a simulation or a simulation model is usually handled using three techniques [15]:

- **control code is executed on another machine**, executing the control code on another machine means a distinct machine or a robot is connected to the simulator machine via a specific network (e.g socket, serial port, etc). The main advantages of the approach are the originality of the controller (the control code can be native and running on the original hardware) and the reduced computing load on the simulation machine. However, this imposes serious limitations on the synchronization with the simulation loop low-level controllers, such as real-time motion level controllers, require synchronization with the simulation loop(i.e executed at the same moment at each simulation pass).

- **control code is executed on the same machine but in another process**, if the control code is executed on the same, but in another process(or another thread), there is the benefit of a balanced load on the CPU cores, but it also comes with a alack of synchronization with the simulation loop.

- **control code is executed on the same machine and thread**, the main advantages of executing the control code on the same machine and in the same thread as the simulation loop are the inherent synchronization with the simulation loop and the fact that any communication or thread switching lag or delay are virtually inexistent. On the other hand, there is an increasing load on the simulation loop CPU core.

The latter two control modes are often implemented via external executables or plugins loaded by the simulator.

Figure 3.5 presents a brief explanation of the control architecture of V-REP.

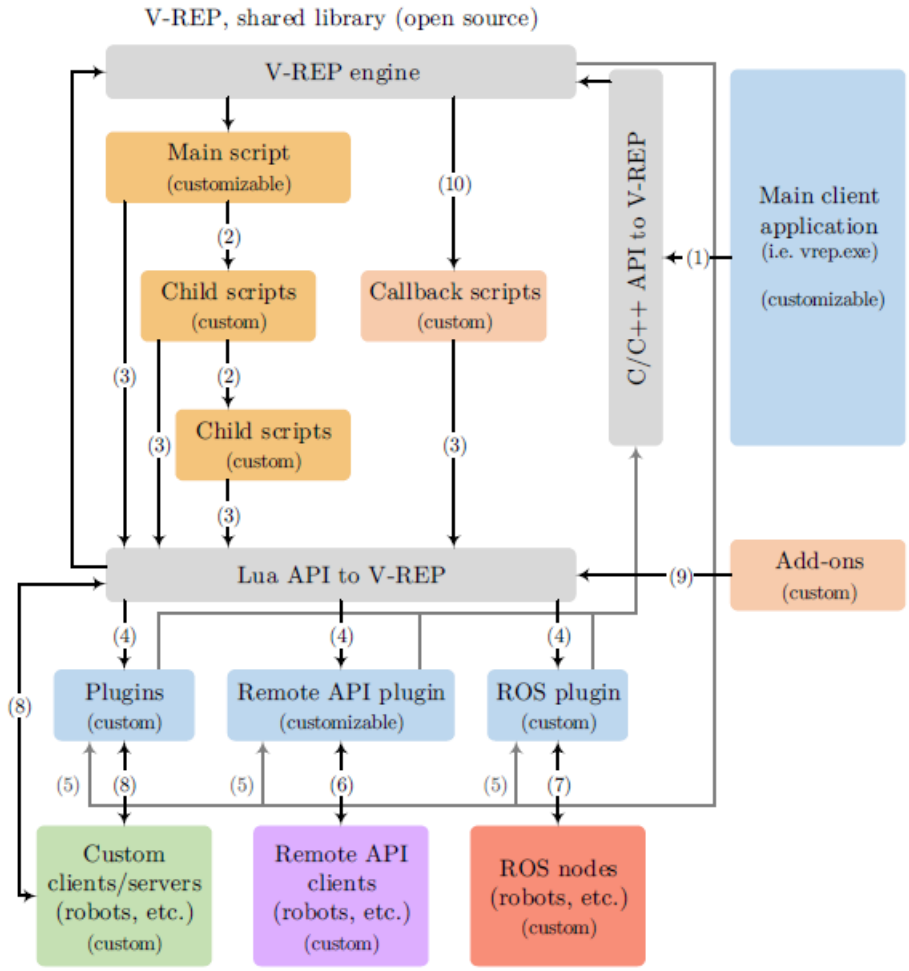


Figure 3.5: V-REP control architecture [15].

Embedded scripts

V-REP is a highly customizable simulator: almost every step of a simulation is user-defined. This flexibility is allowed through an integrated script interpreter. The scripting language is Lua, which is an extension programming language designed to support general procedural programming. By default, V-REP uses the official and original Lua,

but if you wish you can tell V-REP to use another flavor of Lua by setting variable *useExternalLuaLibrary* in *system/usrset.txt* to **true**.

An embedded script is a script that is embedded in a scene (or model), i.e. a script that is part of the scene and that will be saved and loaded together with the rest of the scene (or model). There are different types of embedded scripts that are supported.

Two major types of embedded scripts are supported [15]:

- **Simulation scripts**: simulation scripts are scripts that are executed only during simulation, and that are used to customize a simulation or a simulation model. The main simulation loop is handled via the main script, models/robots are controlled via *child scripts*, low-level dynamic motor controllers are written via joint control callback scripts, and dynamic collisions can be handled via a contact callback script. General callback scripts are multi-purpose callback scripts.

- **Customization scripts**: customization scripts are scripts that are executed all the time, and that are used to customize a simulation scene or the simulator itself.

Although the main script can be customized, it is advisable that only experienced programmer in explicit handling of the simulation modules do it, thus avoiding the risk of the simulation not operating properly. In contrast, the customizing of the scenes and objects through child script is a very important tool, since they can load/unload plugins, register ROS publishers/subscribers, open and handle communication lines by socket or serial port, launch executables or start remote API server services. Child scripts can be executed in a thread or a non-thread fashion.

Non-threaded child scripts are pass-through scripts. This means that every time they are called, they should perform some task and then return control. If control is not returned, then the whole simulation halts. Non-threaded child scripts operate as functions, and are called by the main script twice per simulation step: from the main script's actuation phase, and from the main script's sensing phase. This type of child script should always be chosen over threaded child scripts whenever possible.

Non-threaded child scripts are executed in a cascaded way: child scripts are executed starting with root objects (or parentless objects), and ending with leaf objects (or childless objects).

A non-threaded child script should be segmented in four parts [15]:

- **the initialization part**: this part will be executed just one time (the first time the child script is called). This can be at the beginning of a simulation, but also in the middle of a simulation: remember that objects associated with child scripts can be copy/pasted into a scene at any time, also when a simulation is running. Usually the user should put some initialization code as well as handle retrieval in this part.

- **the actuation part**: this part will be executed in each simulation step, during the actuation phase of a simulation step. Refer to the main script default code for more details about the actuation phase, but typically, the user should do some actuation in this part (no sensing).

- **the sensing part**: this part will be executed in each simulation step, during the sensing phase of a simulation step. Refer to the main script default code for more

details about the sensing phase, but typically, the user should do only do sensing in this part (no actuation).

- **the restoration part**: this part will be executed one time just before a simulation ends, or before the script is destroyed.

Threaded child scripts are scripts that will launch in a thread. The launch of threaded child scripts is handled by the default main script code, in a cascaded way. When a threaded child script's execution is still underway, it will not be launched a second time. When a threaded child script ended, it can be relaunched only if the execute once item in the script properties is unchecked.

Threaded child scripts have several weaknesses compared to non-threaded child scripts if not programmed appropriately: they are more resource-intensive, they can waste some processing time, and they can be a little bit less responsive to a simulation stop command.

Threaded child scripts should be segmented in 3 parts [15]:

- **the initialization part**: this part will be executed when the thread starts. This can be at the beginning of a simulation, but also in the middle of a simulation: remember that objects associated with child scripts can be copy/pasted into a scene at any time, also when a simulation is running. Usually the user would put some initialization code as well as handle retrieval in this part. If the child script was not flagged as execute once and exits the regular part before simulation end, then this part might be called several times (but each time with a Lua state that was reset).

- **the regular part**: this part can be executed in a loop. The code in the loop is in charge of handling a specific part of a simulation . The loop can waste precious computation time and runs asynchronously with the main simulation loop. Thus, it necessitates specialized API for a synchronous loop.

- **the restoration part**: this part will be executed one time just before a simulation ends, or before the thread ends.

Plugins, Add-ons, remote API and ROS

A **plugin** is a shared library (e.g. a dll) that is automatically loaded by V-REP's main client application at program start-up . It allows V-REP's functionality to be extended by user-written functions (in a similar way as with add-ons). The language can be any language able to generate a shared library and able to call exported C-functions (e.g. in the case of Java, refer to GCJ and IKVM). A plugin can also be used as a wrapper for running code written in other languages or even written for other microcontrollers (e.g. a plugin was written that handles and executes code for Atmel microcontrollers).

Plugins are usually used to customize the simulator and/or a particular simulation. Often, plugins are only used to provide a simulation with custom Lua commands, and so are used in conjunction with scripts. Other times, plugins are used to provide V-REP with a special functionality requiring either fast calculation capability (scripts are most of the times slower than compiled languages) or an interface to a hardware device (e.g. a real robot).

An **add-on** in V-REP is quite similar to a plugin: it is automatically loaded at program start-up, and allows V-REP's functionality to be extended by user-written functionality or functions.

Add-on scripts are written in Lua. Two types of add-ons are supported [15]:

- **Add-on functions**: add-on functions appear first in the add-on menu. They can be seen as functions that will be executed once, when selected by the user. Importers and exporters can conveniently be implemented with them.
- **Add-on scripts**: add-on scripts are executed constantly while the simulator is running, effectively running in the background. They should only execute minimalistic code everytime they are called, since the whole application would otherwise slow down. Add-on scripts are called differently depending on the simulation state:
  - When simulation is not running (i.e. stopped or paused): add-on scripts are called just before visualization of the scene
  - When simulation running: add-on scripts are called just after the main script was called

The **remote API** is part of the V-REP API framework. V-REP offers a remote API allowing to control a simulation (or the simulator itself) from an external application or a remote hardware (e.g. real robot, remote computer, etc.). The V-REP remote API is composed by approximately one hundred functions that can be called from a C/C++ application, a Python script, a Java application, a Matlab/Octave program, an Urbi script, or a Lua script. The remote API functions are interacting with V-REP via socket communication in a way that reduces lag and network load to a great extent. All this happens in a hidden fashion to the user. The remote API can let one or several external applications interact with V-REP in a synchronous or asynchronous way (asynchronous by default), and even remote control of the simulator is supported (e.g. remotely loading a scene, starting, pausing or stopping a simulation for instance).

ROS is a distributed pseudo operating system allowing for easy management and communication between multiple computers connected in a network.

V-REP can act as a ROS node that other nodes can communicate with in following 3 ways:

- **The V-REP node offers ROS services**. V-REP ROS services are available as soon as V-REP is launched (given that roscore is running and the ROS plugin to V-REP was correctly loaded).
- **V-REP can be enabled to advertise topics, and publish data to them**. V-REP ROS publishers can only be enabled and are only operational while a simulation is running. One exception to this is the info topic: data to that topic will be streamed as long as V-REP is running.
- **V-REP can be enabled to subscribe to topics, read data from them, and apply the data to specific objects/items inside V-REP**. V-REP ROS subscribers can only be enabled and are only operational while a simulation is running.

The ROS functionality in V-REP is enabled via a plugin. The code to the plugin is open source and is located in the *programming/ros_ stacks* folder. The plugin can easily be adapted to the user own needs. The plugin is loaded when V-REP is launched, but the load operation will only be successful if roscore was running at that time. Make sure to inspect V-REP's console window or terminal for details on plugin load operations.

Following diagram illustrated in the Figure 3.6 how ROS messages are handled on the server side (i.e. on the V-REP ROS plugin side):

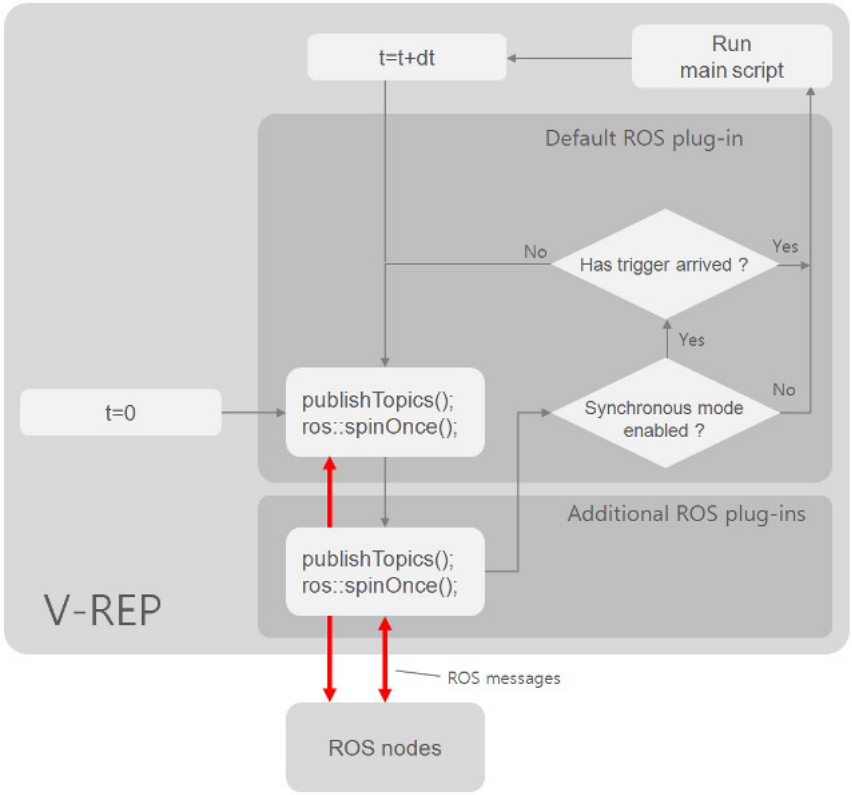


Figure 3.6: ROS message handling, server side [15].

A thorough description of ROS framework is presented in the following segment of this chapter.

3.3.2 Scene Objects

The main elements in V-REP that are used for building a simulation scene are scene objects(objects in short). This objects are attached or assembled to each other in tree-like hierarchy to build a simulation model [15].

The following list gives a brief functional description of each object type:

- **Shapes**: a shape is a rigid mesh that is composed of triangular faces.
- **Joints**: a joint object is a joint or actuator. Four types are supported: revolute joints, prismatic joints, screws and spherical joints.
- **Graphs**: a graph is used to record and visualize simulation data.

- **Dummies**: a dummy is a point with orientation. Dummies are multipurpose objects that can have many different applications.

- **Proximity sensors**: a proximity sensor detects objects in a geometrically exact fashion within its detection volume. V-REP supports pyramid, cylinder, disk, cone-and ray type proximity sensors.

- **Vision sensors**: a vision sensor is a camera-type sensor, reacting to light, colors and images.

- **Force sensors**: a force sensor is an object able to measure forces and torques that are applied to it. It also has the ability to break if a given threashold is overshot.

- **Mills**: a mill is a convex volume that can be used to perform cutting operations on shape objects.

- **Cameras**: a camera is an object that allows seeing the simulation scene from various view points.

- **Lights**: a light is an object that allows illuminating the simulation scene.

- **Paths**: a path is an object that defines a path or trajectory in space. It can be used for various purposes, also as a customized joint or actuator.

- **Mirrors**: a mirror can reflect images/light, but can also operate as an auxiliary clipping plane.

Some of above objects can have special properties allowing other objects or calculation modules to interact with them. Objects can be: ***collidable*** objects can be tested for collision against other collidable objects; ***measurable*** objects can have the minimum distance between them and other measurable objects calculated; ***detectable*** objects can be detected by proximity sensors; cuttable objects can be cut by mills; ***renderable*** objects can be seen or *detected* by vision sensors; ***viewable*** objects can be *looked through, looked at*, or their image content can be visualized in views.

Each object has a position and orientation within the simulation scene. An object's position and orientation its refer as configuration of the object. Objects can be attached to other objects (or built on top of each other). If object A is built on top of object B, then object B is the parent and object A is the child. Objects configurations weren't changed (both objects kept their respective configuration), however if the user moves object B, object A will automatically follow, since object A is attached to object B. This daisy chain connection porvides a very useful tool to build simulated models when robotics arms, or other serial manipulator, are intended to be simulated [15].

Every object has an absolute configuration (or cumulative configuration) that is relative to the world's reference frame, and a local configuration (or relative configuration) that is relative to the parent object's reference frame. In above example, when object A became child of object B, object A's absolute configuration didn't change, but its local configuration was modified.

3.3.3 Calculation Modules

V-REP offers powerful calculation functionalities, or calculation modules, that are not directly encapsulated in objects (like proximity sensors or vision sensors for instance), but that rather operates on one or several objects . Calcultion modules include:

- **the collision detection** module;
- **the minimum distance calculation** module;
- **the inverse kinematics calculation** module;
- **the geometric constraint solver** module;
- **the dynamics** module;
- **the path planning** module;
- **the motion planning** module.

With the exception of *dynamics* module, all other calculation modules allow registering *calculation objects* that are user defined. These object are different from scenes objects, but are indirectly linked to them by operating on them. Calculation objects can be defined as the interaction, with respect to a calculation module, that occurs between scene objects. For example, if *collision object* A is defined by *collidable object* B and *collidable object* C , it represents the collision detection performed between object B and object C [15].

The **collision detection** module allows registering collision objects which are collidable entity-pairs (collider entity and collidee entity). During simulation, the collision state of each registered collision object can then be visualized with a different coloring, or recorded in a graph object. The module with only detect collision, it will not react to them.

The **distance calculation** module allows registering distance objects which are measurable entity-pairs. During simulation, the minimum distance segment of each registered distance object can then be visualized, or recorded in a graph object. V-REP can measure the minimum distance between two measurable entities in a very flexible way. The calculation is an exact mesh-mesh (or collection of meshes) minimum distance calculation. The distance calculation module will only measure distances; it does however not directly react to them.

V-REP's **inverse kinematics** (IK) calculation module is very powerful and flexible. It allows handling virtually any type of mechanism in inverse kinematics mode (IK mode) or forward kinematics mode (FK mode). The problem of IK can be seen as the one of finding the joint values corresponding to some specific position and/or orientation of a given body element (generally the end effector). More generally, it is a transformation from the task space coordinates into the joint space coordinates. For a serial manipulator for instance, the problem would be to find the value of all joints in the manipulator given the position (and/or orientation) of the end effector. The inverse problem (finding the end effector position given the joint values) is referred to as FK problem and is often perceived as an easier task than IK. This is surely true when dealing with open kinematic

chains, but does not hold true for general type mechanical configurations. The module also support conditional, damped/undamped, and weighted resolution.

A more detail approach is used in this module since its a major parameter on this work.

V-REP uses IK groups and IK elements to solve inverse and forward kinematics tasks. It is important to understand how an IK task is solved in order to take full advantage of the functionality of the module. An IK group contains one or more IK elements. IK elements specify simple kinematic chains. One IK element represents one kinematic chain. A kinematic chain is a linkage containing at least one joint object [15] . Basically, an IK element is constructed by:

- a **base** (any type of object, even a joint. In that case however the joint is considered as rigid). It represents the start of the kinematic chain.
- several **links** (any type of object except joints). Joints which are not in IK mode are however also considered as links (in that case they behave as rigid joints (fixed value)).
- several **joints**. A joint which is programmed to behave in inverse kinematics mode (IK mode).
- a **tip**. The tip is always a dummy and is the last object in the considered kinematic chain (when going from the base to the tip). The tip dummy should be linked to a target dummy and the link should be an IK, tip-target link type.
- a **target**. The target is always a dummy and represents the position and/or orientation the tip should adopt (or follow) during simulation. The target dummy should be linked to a tip dummy and the link should be an IK, tip-target link type., just as previous mention.

As mention in the previous descriptions the *tip* implemented on the last object in the kinematic chain will always try to adopt the position and/or orientation of the target. Figure 3.7 illustrates an IK element and the respective IK element task for the target acquisition.

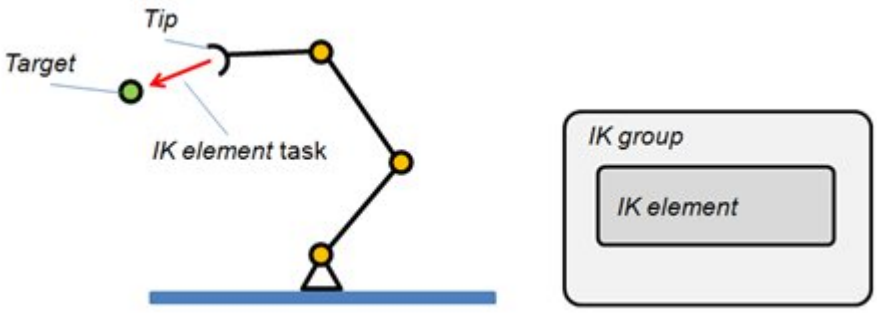


Figure 3.7: IK element and corresponding model of the IK solving task [15].

When the simulation is running and some additional parameters have been defined correctly , then the mechanism (the specified kinematic chain) should move towards the

target. This is the most basic case of IK task. Two separate kinematic chains are handled in an identical fashion, however this time, two IK groups are needed (and each one of them should contain one IK element for each kinematic chain). Solving order of the two IK groups is not important, because the calculation are sequential as its shown on the Figure 3.8.

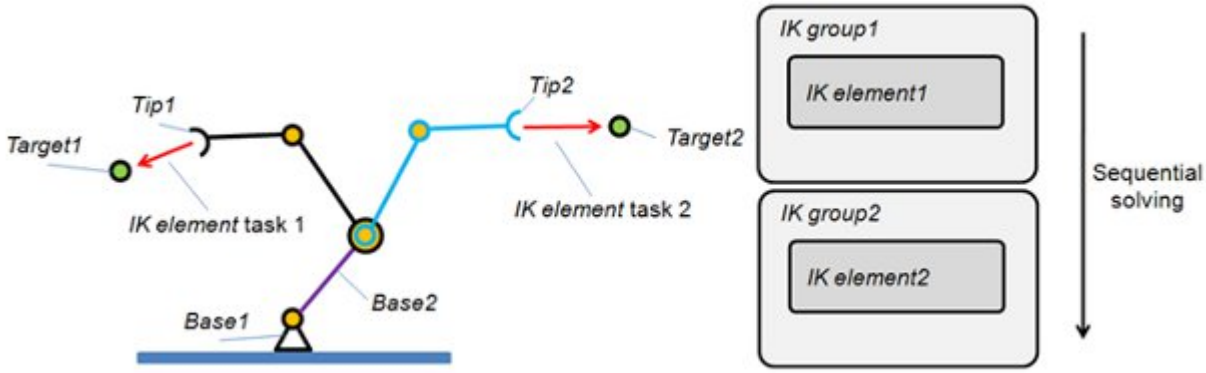


Figure 3.8: Two IK chains sharing one common link but no common joints and corresponding model of the IK solving tasks [15].

However, a more difficult case appears when two or more kinematic chains share common joints. In that case sequential solving doesn't work most of the time (in following example, the two IK elements tend to rotate the common joint into opposite directions) and a simultaneous solving method is needed. To simultaneously solve several IK elements, just group them into one common IK group. This case is illustrated in the following Figure 3.9.

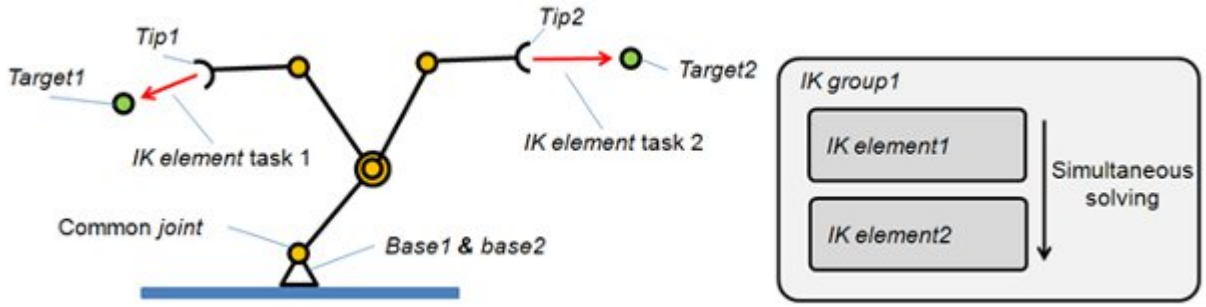


Figure 3.9: Two IK chains sharing one common joint and corresponding model of the IK solving task [15].

The **geometric constraint solver** is and alternative to fast and accurate IK calculation, IK calculation can be difficult to implement depending on the mechanism's complexity. The geometric constraint solver module is slower and less precise at solving kinematic problems problems, but might be easier and more intuitive to use. Moreover, it allows interacting with a mechanism in a more flexible way than the inverse kinematics calculation module.

V-REP's **dynamics** module currently supports four different physics engines: the *Bullet physics library*, the *Open Dynamics Engine*, the *Vortex Dynamics engine* and the *Newton Dynamics engine*. At any time, the user is free to quickly switch from one engine to the other according to his/her simulation needs. The reason for this diversity in physics engine support is that physics simulation is a complex task, that can be achieved with various degrees of precision and speed [15].

The dynamics module allows simulating interactions between objects that are near to real-world object interactions. It allows objects to fall, collide, bounce back, but it also allows a manipulator to grasp objects, a conveyor belt to drive parts forward, or a vehicle to roll in a realistic fashion over uneven terrain. In this work the engine of choice is the ODE.

The **path planning** module allows handling path planning tasks in 3D-space, and in 2D-space for vehicles with non-holonomic motion constraints. The path planning module does not include motion planning for kinematic chains, which is handled by the motion planning module. The module utilizes a Rapidly-exploring Random Tree (RRT) algorithm to calculate its approach.A path planning task usually takes several input values or parameters: a start position (or start configuration): this is the initial configuration of a device (e.g. robot); a goal position (or goal configuration): this is the desired configuration for the device or robot; obstacles: those are the objects that the device (or robot) shouldn't be colliding with, while following a path from the start to the goal configuration.

The **motion planning** module allows handling motion planning tasks for kinematic chains. It does not include path planning for loose objects in space, which is handled by the path planning module. A motion planning task allows to compute a trajectory (usually in the configuration space of the manipulator) from a start configuration to a goal configuration, by taking into account the manipulator kinematics, the manipulator joint limits, the manipulator self-collisions, the collisions between manipulator and obstacles (or the environment) [15].

The modules presented are integrated in the simulator and implement by V-REP, without making any assumptions on the simulation. This behavior improves the flexibility and portability of the models, distinguishing from a reliance on external libraries and recompiling or installing needs.

3.4 ROS Framework

3.4.1 Introduction and Concepts

The Robot Operating System (ROS) is the element that connects all the separate applications on this work. It defines the communication between the haptic devices with the V-REP simulation.

ROS is a framework that is widely used in robotics. the philosophy is to create functionalities that can be shared and used in others robots without much effort. ROS was originally developed in 2007 by the Stanford Artificial Intelligence Laboratory (SAIL). The platform is now maintained by the open-source community and Willow Garage, responsible for the Personal Robots Program(PR/PR2 robots). A lot of research institutions and companies have started to develop projects and adapt their products to be used in ROS. NAO robot is just an example of a fully supported platform. Sen-

sors and actuators used in robotics have also been adapted to be used with ROS. This pseudo-operating system provides standard operating system facilities such as hardware abstraction, low-level device control, implementation of commonly used functionalties, message passing between processes, and packages creation and management [21].

Designed to run under a Unix-like system, ROS eases the interaction of software and hardware, using a modular architecture. This architecture is based on a centralized topology where all the processes are connected in a network. Any node in the system can access this network [21]. ROS utilizes a peer-to-peer topology to connect a number of processes, potentially on a number of different hosts. the fundamental concepts of ROS implementation, regarding the peer-to-peer network, *nodes, Master, Parameter Server, messages, services, topics*, and *bags* [22].

Nodes are processes that perform computation. ROS is designed to be modular at a fine-grained scale; a robot control system usually comprises many nodes. For example, one node controls a laser range-finder, one node controls the wheel motors, one node performs localization, one node performs path planning, one Node provides a graphical view of the system, and so on. A ROS node is written with the use of ROS client library that fully support C++ and Python programming languages.

The ROS **Master** provides name registration and lookup to the rest of the peer-to-peer network. Without the Master, nodes would not be able to find each other, exchange messages, or invoke services. The **Parameter Server** allows data to be stored by key in a central location. It is currently part of the Master.

Nodes communicate with each other by passing **messages**. A message is simply a data structure, comprising typed fields. Standard primitive types (integer, floating point, boolean, etc.) are supported, as are arrays of primitive types. Messages can include arbitrarily nested structures and arrays (much like C structs).Messages are routed via a transport system with *publish / subscribe* semantics.

A node sends out a message by *publishing* it to a given **topic**. The topic is a name that is used to identify the content of the message. A node that is interested in a certain kind of data will *subscribe* to the appropriate topic. There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics. In general, publishers and subscribers are not aware of each others' existence. The idea is to decouple the production of information from its consumption. Logically, one can think of a topic as a strongly typed message bus. Each bus has a name, and anyone can connect to the bus to send or receive messages as long as they are the right type.

The publish / subscribe model is a very flexible communication paradigm, but its many-to-many, one-way transport is not appropriate for request / reply interactions, which are often required in a distributed system. Request / reply is done via **services**, which are defined by a pair of message structures: one for the request and one for the reply. A providing node offers a service under a name and a client uses the service by sending the request message and awaiting the reply. ROS client libraries generally present this interaction to the programmer as if it were a remote procedure call.

At last there is **bags**. Bags are a format for saving and playing back ROS message data. Bags are an important mechanism for storing data, such as sensor data, that can be difficult to collect but is necessary for developing and testing algorithms.

ROS is an asset of invaluable worth in the programming of robots. The large online community provides a well documented source and the it presents a distribution under

a BSD license (permissive free software license), which allows the development of both non-commercial and commercial projects.

ROS provides the PHUA platform and any other humanoid robot platform with hardware abstraction and low-level device control, also it enables several modules with a specific functional providing efficient control and monitoring of the simulation parameters.

It should be noted that ROS has several other features and utilities that were not, and will not be, fully described and exploited in the present work.

3.4.2 PHANToM Configuration

The configuration established in the previous work, is still used on the present work. It was established two configuration, the IK mode utilized a single PHANToM Omni haptic device configuration, while the Torque mode utilized a dual Omni configuration. This segment will provide a brief explanation on the dual configuration installation, for further details refer to [1] .

As previous mentioned the outdated Linux drivers for the Omni device wont allow for a paired configuration, thus each haptic device is connected to one of the two computer avaible for the purpose. At ROS level, they run perfectly independent of each other, on two different nodes. To implement the communication across machines and between the haptic devices and V-REP, running on an arbitrary machine, the following conditions must be met [22]:

- Only one ROS master is needed. Therefore, one of the two machines must be selected to run it.
- All nodes (device handles nodes and others) must be configured to use the same master.
- there must be complete, bi-directional connectivity between the machines, on all ports.
- Each machine must advertise it self by a name that all other machines can resolve.

The connection of the two computers is done indirectly through a central network fixture, rather than cable two computers directly. This method requires two network cables, one connecting each computer to the fixture, which can be an Ethernet hub, a USB hub, a switch or a router. This solution is also prepare to accommodate any extra devices needed for future works.

Even though connecting the two computers through a wireless connection is also possible, such as WI-FI, Bluetooth or Infrared connection, a wired Ethernet Local Area Network (LAN) is the most appropriate solution in the context of the project, wired connection also provides better reliability and performance.

Using a central infrastructure, e.g. a router, to connect the two computer allows not only the distribution of data and resources, but also an easy Internet connection as its shown in the Figure 3.10.

A router is a more sophisticated network device than a switch or a hub, offering features like firewall support and DHCP server capability.

Thus, to assemble the work station of this project is needed:

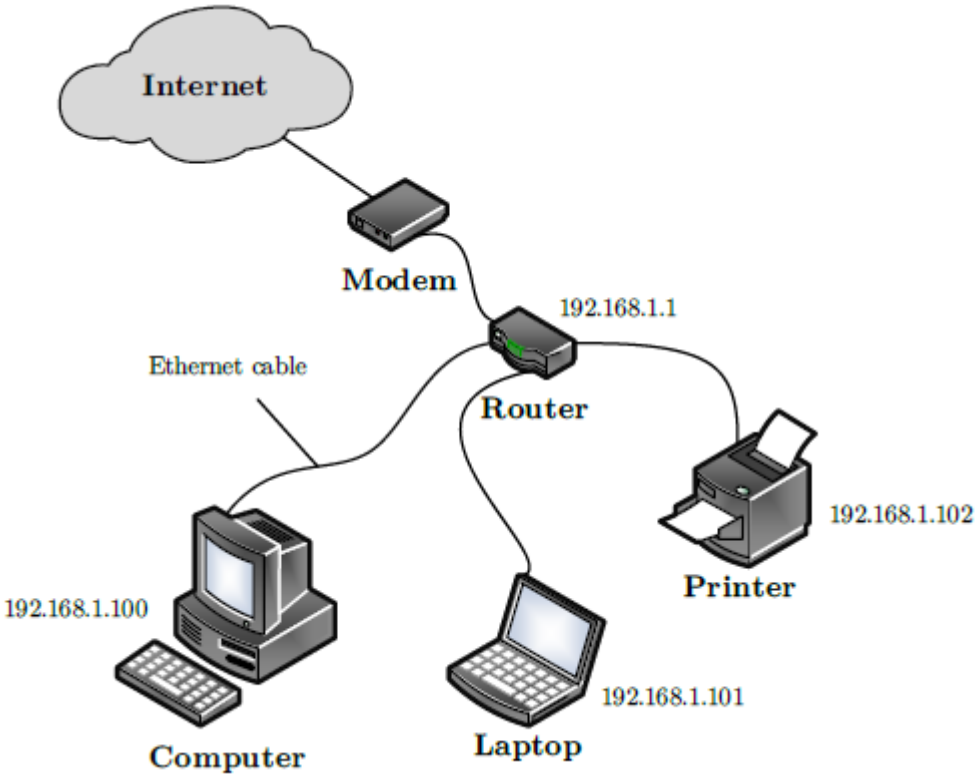


Figure 3.10: Example of a network using a router

- two desktop computers;
- two PHANToM Omni haptic devices;
- two Firewire PCI express cards;
- one network router.

While both machine have ROS modules running on them, only one of the computers can have the V-REP simulator running. Additionally, beside joining the local area network internally, the router will also join the LAN to the Wide-Area Network (WAN) of the Internet. Figure 3.11 illustrated the completed setup.

A Firewire PCI express card is mounted on each computer, the card serves the communication with the haptic devices. IEEE 1394a 6 pin cables are used to connect the devices to the computer. Appendix C describes how to successfully enable the connection and configure the haptic devices.

When a ROS node advertises a topic, it is provided a *hostname:port* combination for other nodes to contact when needing to subscribe to that topic. For a successful hostname address, its necessary to configure the host file in */etc/hosts*. Apart from the *localhost* name, each file must contain the IP address and the hostname of the other computer. With that the router associated the name each machine with the respective IP address.

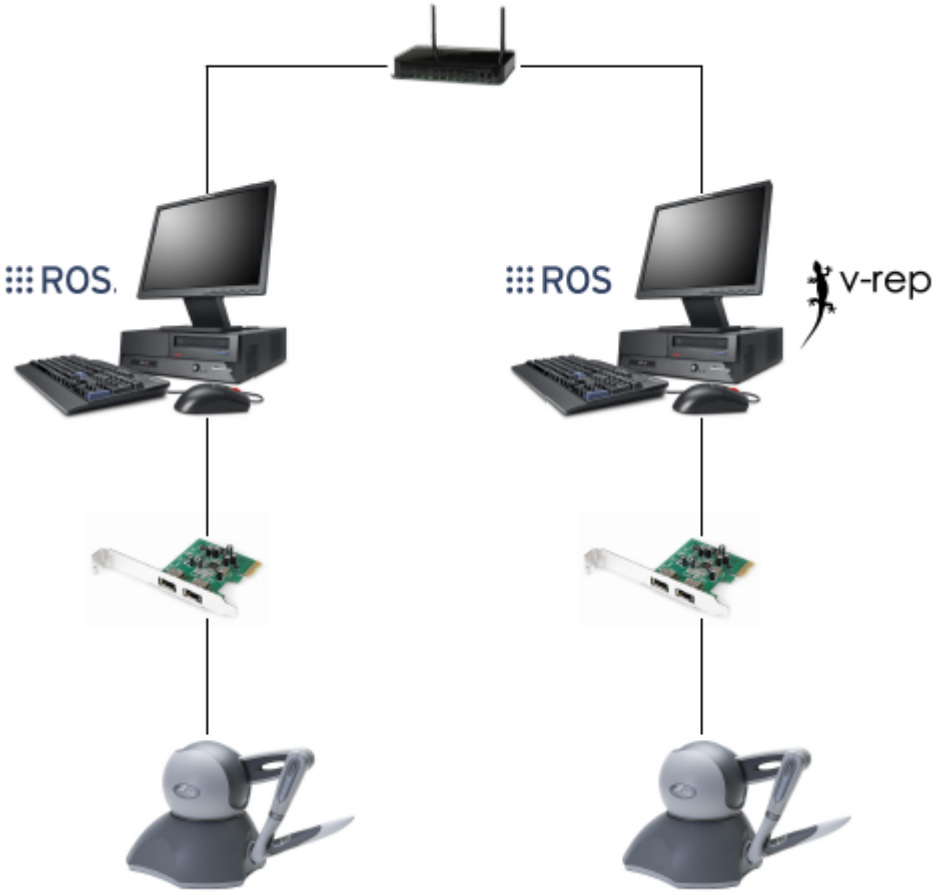


Figure 3.11: Experimental setup for cooperative haptics teleoperation [13].

V-REP enables ROS functionality via a plugin, which only is loaded if *roscore* is already running previous to the start of V-REP. For organization purpose the ROS master is running in the same machine as V-REP is, while the remaining machine will run the high computational resource modules, e.g. robot state and haptic feedback modules. While this configuration was set to deal with the hardware restriction, it allows for one computer to be fulling dedicated to the V-REP simulation, with the exception of the Omni control module, lifting the computational effort, thus providing a more responsible simulation.

3.4.3 Package *humanoid_simulation*

As mention before the platform has several modules apart from the V-REP simulator, these modules operate as ROS nodes and are part of the *humanoi_simulation* package. A *package* may contain ROS runtime processes (nodes), a ROS dependent library, datasets, configuration files, or anything else that is usefully organized together. The *humanoi_simulation* package was develop in the previous work [1], it contains modules designed to handle the communication between hardware and software, and to repro-

duce the control loop idealized for robot teleoperation. The modules presented in the *humanoi_simulation* package are written in C++ programming language, that define *classes* that can be ported to other platforms, or recycle into new applications or algorithms. This work functions with this package, creating new nodes or modifying existing ones. A *Doxygen*-based documentation was created to help the understanding of the code structure, the APIs used and the functions developed.

The previous work's package was composed of four modules, excluding V-REP: *phantom_control*, *robot_state*, *haptic_feedback*, and *recorded_data*. These nodes and the V-REP module had a linear interaction between them, the previous interaction is represented in the Figure 3.12.

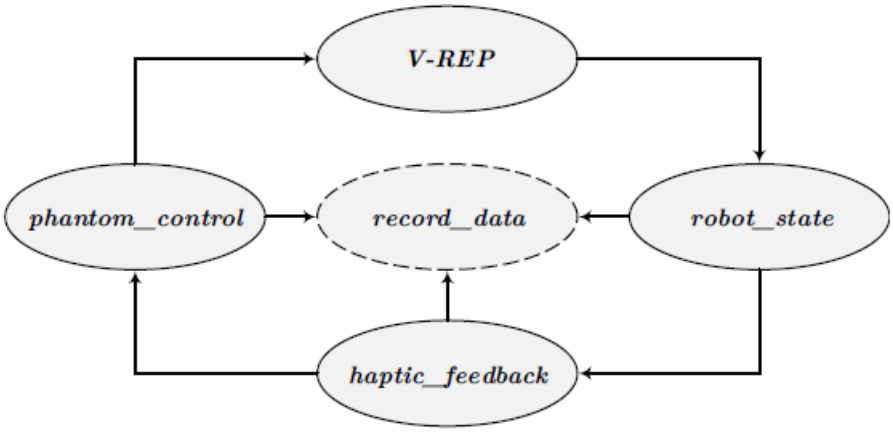


Figure 3.12: Interactions between the previous designed ROS modules [1].

In this work two nodes were added to the package, the *ghost_control* and the *archiver*. Also, due to the new customized V-REP interface, explained further in the work, the interaction between some nodes and V-REP became more mutual than linear. Figure 3.13 illustrates the current interaction between nodes of the humanoid platform.

The *robot_state* module is responsible for calculating the COP and defining the robot's ground base support. To complete those tasks it receives the force measure by the sensors on the simulation. It sends the calculated COP global position back to V-REP for a real-time display of it, and to the *haptic_feedback* module with the addition of the support polygon position. This module receives settings from the V-REP interface, that are describe the Chapter 4. According to the settings received the module utilizes the sensor information into force feedback according to the selected formulation method. The computed force feedback is transmitted to the *phantom_control* module, which in turn connects to the haptic device to reproduce them to the user. The forces implemented on the device are subjected to alterations in their direction base on settings received from the V-REP. V-REP interface also transmits settings that configure the output of the Omni device to the simulation such as the contribution of the transitiveness of the simulated model in contrast with the dislocation of the device. The *phantom_control* module connects the V-REP and the Omni device simultaneously, returning the devices status affected by the selected settings to the simulator, while constantly updating the forces produce by the feedback. The *record_data* and the *archiver* modules don't interact directly with the process of the real-time teleoperation. The *record_data* module

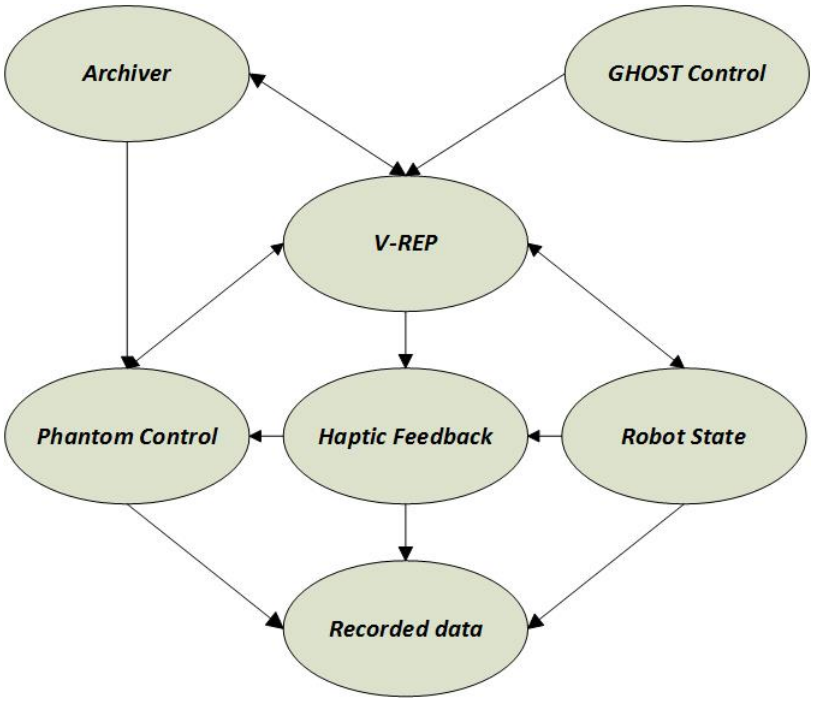


Figure 3.13: Interactions between the current designed ROS modules.

function is to record the data transmited between the other modules, to be used for futher visualization adn analyses. The *archiver* module has two distinct roles, similar to the *record_data* module, it records the settings outputs of the V-REP interface for each individual user, uploading them to the simulation at its initialization. The second function of it is to, when enabled, upload saved joints configurations, that represent certain maneuver through time, to the *phantom_control* module to function as the haptic guidance tool explain in Chapter 5. Similar to the second function of the *archiver* module the *ghost_control* module functions as support tool, controlling the visual guidance tool presented on Chapter 4.

The next segment, gives a brief discretion to the layouts of the modules.

Module *phantom_control*

The *phantom_control* module is responsible for the interaction between the haptic device and the simulation, this being the primary interface for the robot control. The application is able to setup a inverse kinematics or a joint-by-joint robot control. To successfully communicate to the haptic devices and the V-REP simulator, the application has to use both ROS and OpenHaptics protocols. The application depends on other modules to execute some of its functionality, however if those modules were to crash during a simulation the control over the teleoperation wont be affected.

The module's structure follows the typical organization of an HDAPI program, while also including the ROS functionality. The HDAPI consists if two main components: the *device* and the *scheduler*. The scheduler manages a high frequency thread for sending forces and retrieving state information from the device, also reference as the *servo loop*. This is a high priority thread, responsible for low-level interaction between the libraries

and the PHANToM hardware, thus securing the 1000Hz for the force update.

Prior to the device's loop start, a series of routines take place that verify calling arguments, hardware state and communications. The general pattern for the HDAPI is to initialize eh device, enable forces, create scheduler callbacks to define force effects and start the scheduler. When the application is closed, the device and scheduler are cleanup. With the device and scheduler are initialized the module also starts a parallel *main loop*, with the ROS node running on it. Figure 3.14 illustrates a simplified schematic of the *phantom control* threaded application.

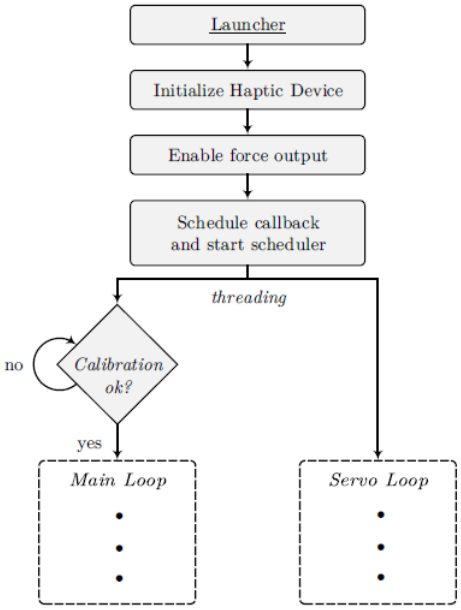


Figure 3.14: Simplified schematic diagram of phantom control module, which runs two separated thread, the main loop and the servo loop [1].

The **servo loop** operates within a haptic frame ensuring data consistency used by the device. At the start of each frame the devices state is updated and store for use in the same frame. on the end of the frame, new states such as forces or torques are sent to the device. Figure 3.16 illustrates a simplified schematic of the servo loop implemented in the *phantom control* module.

Thanks to implemented signals in each scene of the simulation, the module can assert which configuration is selected, thus being able to select between vector orientated force rendering or torque joint implemented force.

The parallel **main loop** started by the module, initializes the ROS node and configures the ROS messaging and servicing. Only the retrieval of the joints handles are made through the service protocol, with all methods of communication base on publishing/subscribing protocol. Figure 3.15 illustrates a simplified schematic of the servo loop implemented in the *phantom control* module.

The configuration of the devices buttons provide for useful support tool for the teleoperation. *Button 2* state configuration allows the user to choose between the states of control. As the module detects the active scene, the user selects which leg the devices controls.

- On inverse kinematics control;
- On torque control;
  - dual leg joint-by-joint configuration;
  - left leg joint-by-joint configuration;
  - right leg joint-by-joint configuration;
- On guidance control;
  - following left leg joint-by-joint configuration;
  - following right leg joint-by-joint configuration;

Due to linearity of the IK control, no states are affected, since there is only one state for the configuration. Although all the other configurations can be select online without compromising the simulation.

The IK mode is define by the control of a three-dimensional point, that represents the hip's target of the implemented IK element on V-REP, in function of the dislocation made by the end-effector of the device with the *Button 1* of the device pressed.

The torque mode defines a more complex control, it takes the position of the haptic device joints and sends it to the simulation, thus it assuming the values received. As stated before, with the *Button 2* the user can select which leg assume the sent configuration, with the dual configuration assuming a parallel configuration on bout legs. The data of the joints positions is only sent with the *Button 1* of the device pressed, when its state is false the module runs the support tool *Joint Configuration Finder*, which is explain in Chapter 5.

The guidance control mode is defined by the haptic guidance tool developed and explain in Chapter 5. Its functionality is similar to the torque mode, but instead of the device's joint position it sends vector with preload joints position.

In spit of the PI controllers used for the haptic guidance tool and the Joint Configuration Finder, calculate force feedback, their implementation is made on the *phantom_control* module, due to the necessity of accessing the device's joints position for the calculation of the error for the PI controllers described on the Chapter 5.

Module *robot_state*

The *robot_state* module purpose is to obtain information that if able to translate the current state of the robot, as well as to define important indicators of the system's kinematics and dymanics.

The structure of the module is based on a C++ class responsible to monitoring the paramenters published by V-REP involving the robot state. This parameters are stored and use to calculate important utensils for the platform.

The module main functions are to computarize the sensor data from the V-REP to calculate the simulated model's CoP, and to manifest the support polygon of the robot from dummy objects placed on the robot's feet.

This parameters are then published to the *haptic_feedback* module for its force formulation. Apart form the *haptic_feedback* module it also publishes the CoP to V-REP to be utilized in the Torque User Trainer interface, that is presented on Chapter

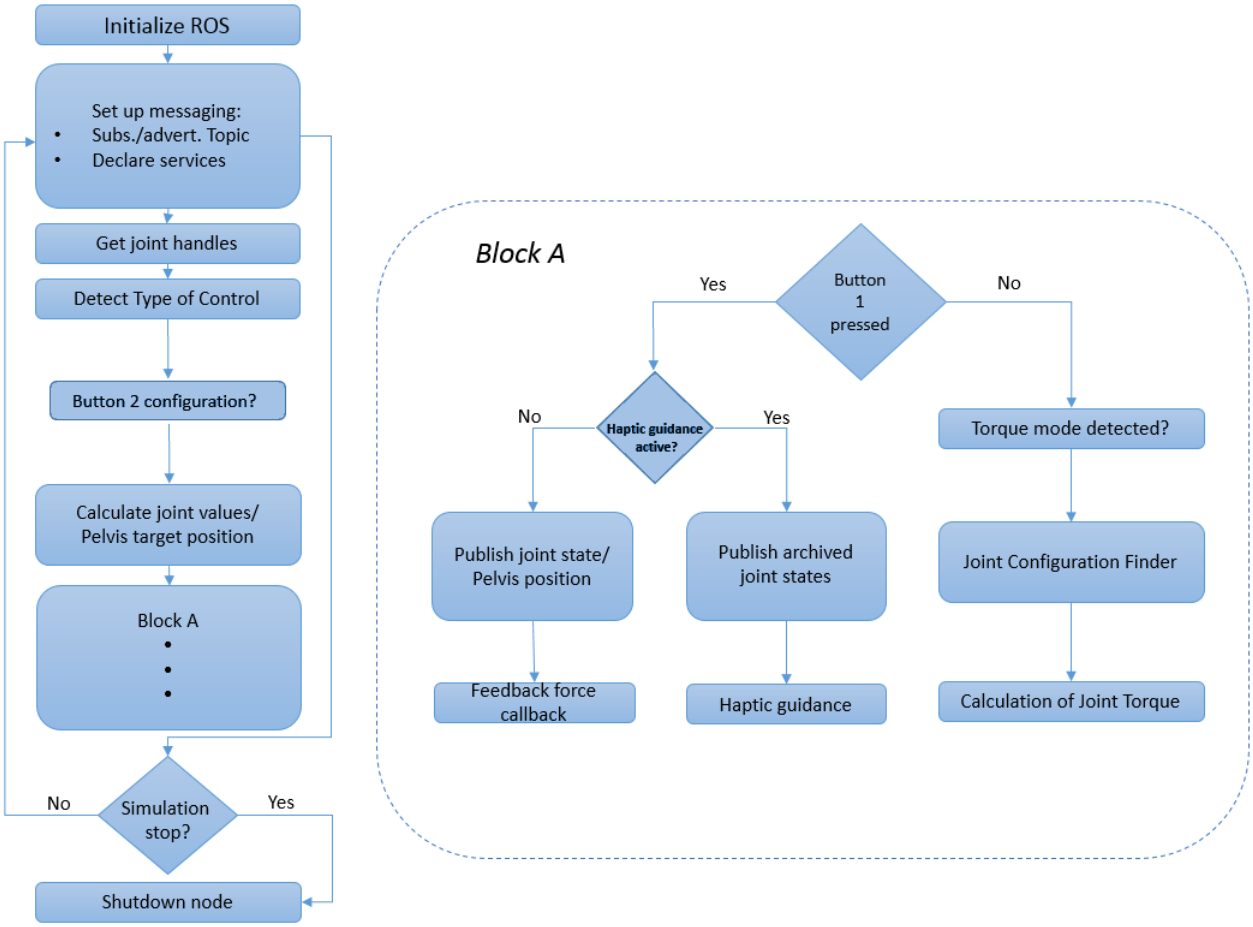


Figure 3.15: Simplified schematic diagram of the main loop in the *phantom control* module.

5. To facilitate the understanding of the module Figure 3.17 illustrates a simplified schematic of the *robot state* module.

Module *haptic_feedback*

Similar to the *robot_state* module the *haptic_feedback* module's struct is centered on a C++ class. The function this class is to utilize the utensils provided from the *robot_state* module and calculate the force feedback correspondent to the analyzed robot state.

It mainly evaluates the CoP position with the support polygon, when, if within it, applies an algorithm to the calculated minimum distance of the CoP to the support polygon edges, this procedure is applied only to the IK control mode. For the Torque control mode, it analyses the individual torque of each leg joint, where it applies an algorithm in function of the relation of the CoP position with the assigned foot position. The equations develop for the algorithms are approached in Chapter 4 & 5.

The IK mode algorithm has multiplied equations configurations, this configurations are selected through the V-REP interface, and published to the *haptic_feedback* module for correct selection, any alterations can be done with the simulation online, without the

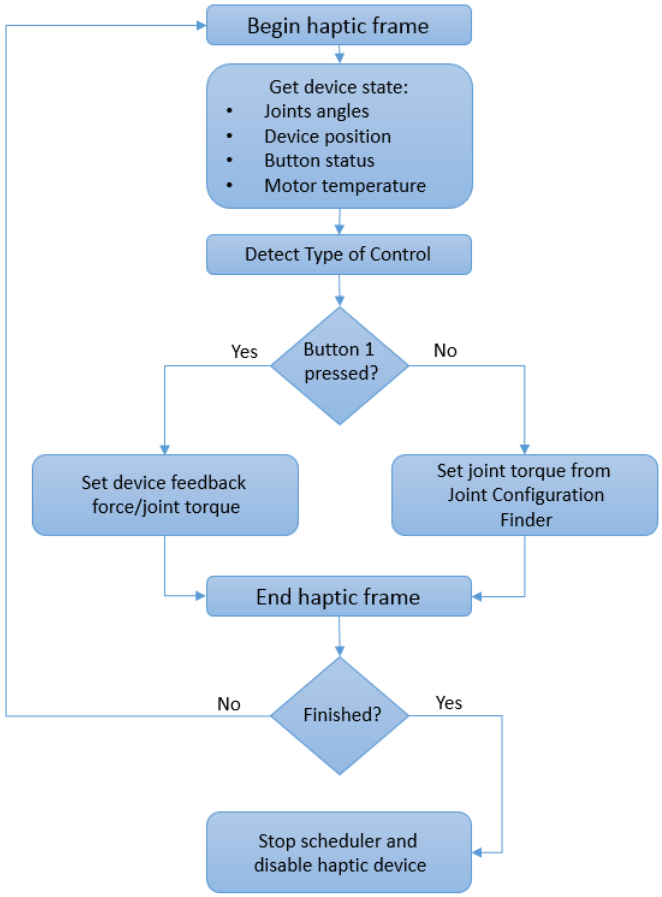


Figure 3.16: Simplified schematic diagram of the servo loop in the *phantom control* module.

need of extra compilation time.

Figure 3.18 illustrates a simplified schematic diagram of the structure of the *haptic feedback* module and Figure 3.19 represents a schematic diagram of the force formulation for each control configuration.

Module *ghost_control*

Ghost_control module is identical to the *phantom_control* module, it structured with OpenHaptics and ROS protocols. It is similar in every aspect to the *phantom_control* module IK configuration, differing in the target *target dummy* and the inability to reproduce forces on the user.

This module was developed to control the visual haptic tool presented in Chapter 4, with slight modification its workspace and references to adjusted to the different model assigned.

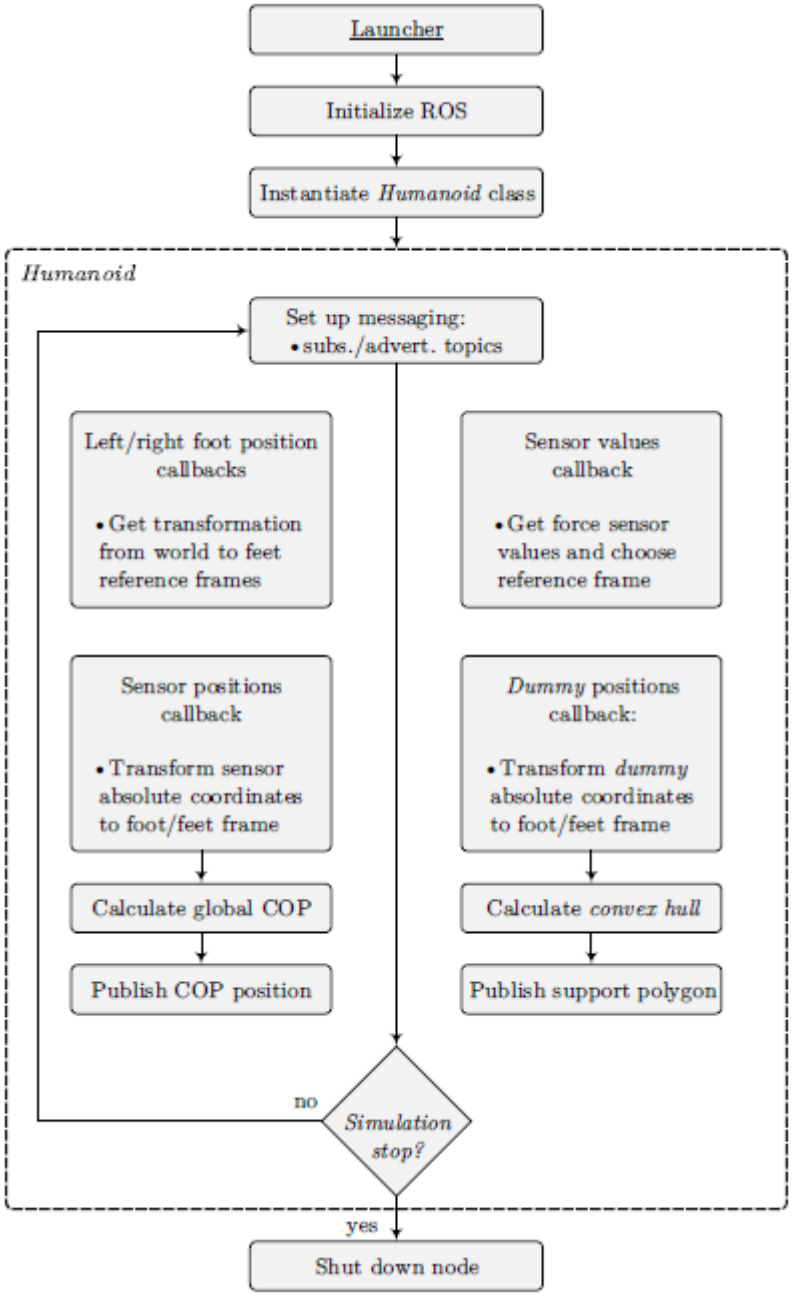


Figure 3.17: Simplified schematic diagram of the servo loop in the *robot state* module [1].

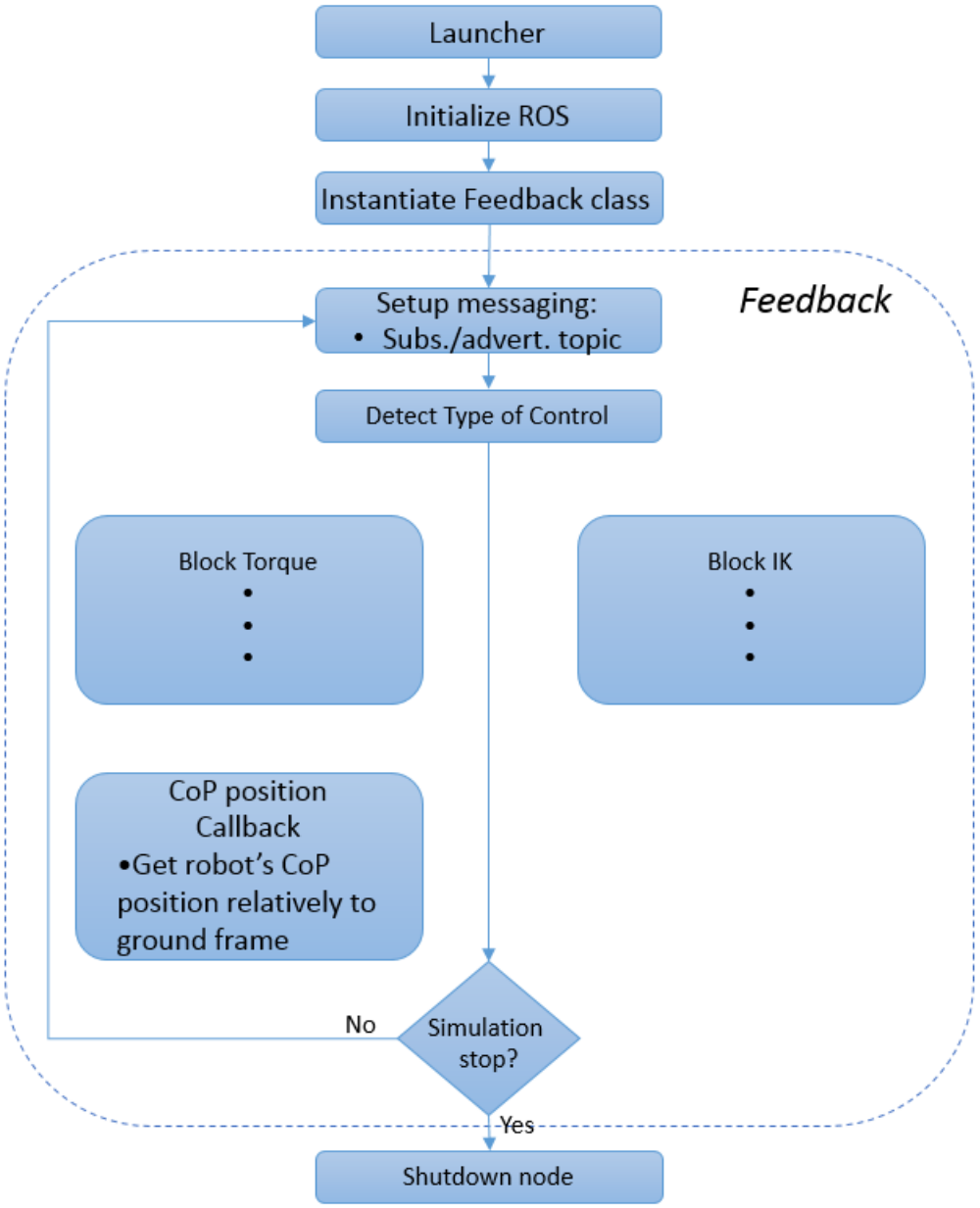


Figure 3.18: Simplified schematic diagram of the *haptic feedback* module.

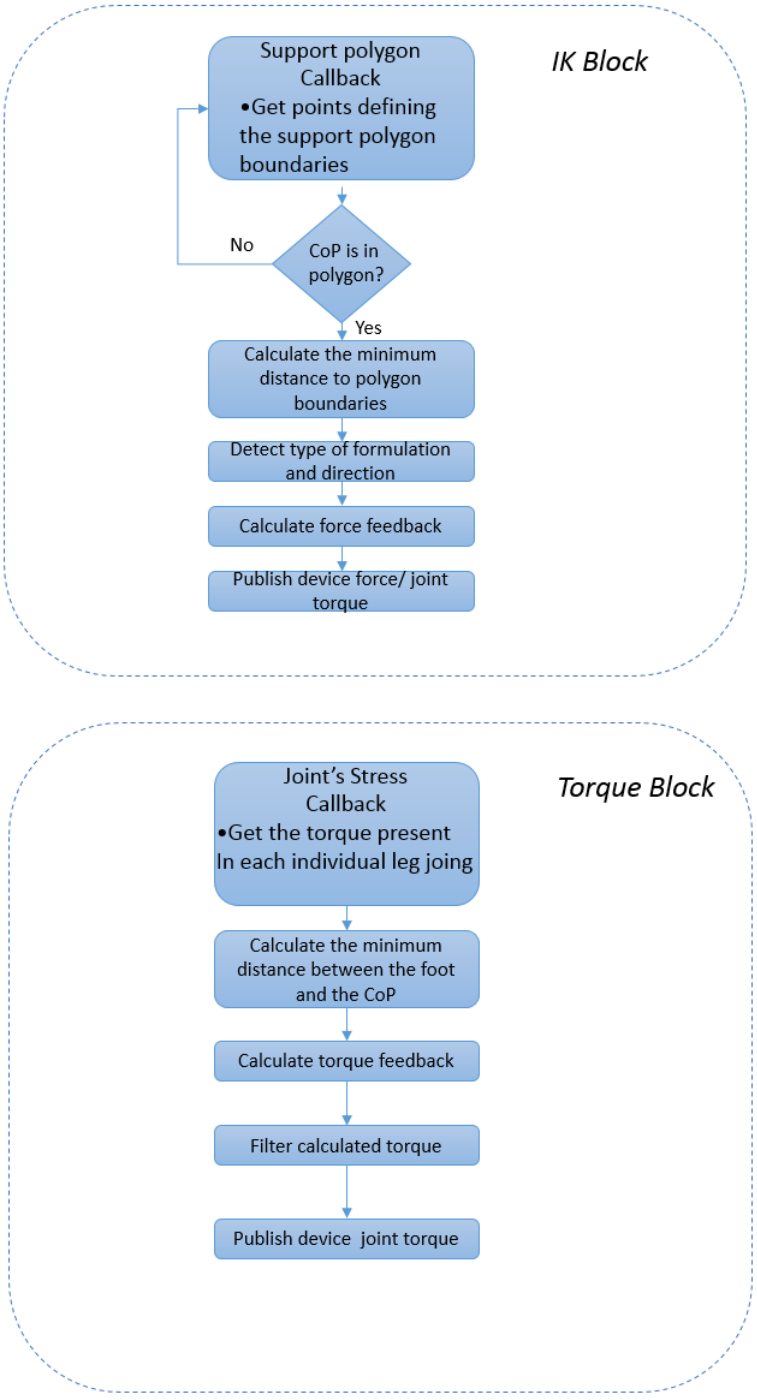


Figure 3.19: Simplified schematic diagrams of the responsible blocks for the calculation of the feedback in each control configuration.

Chapter 4

Haptic Training Interface for Inverse Kinematics Control

The simulation of PHUA robot on V-REP simulator, allows the user to refine his control before switching to the real robot. Unfortunately to gain affinity to the control inputs and outputs of the system based on instruction is too difficult, when a great part of the control is supported by motor skill and dynamics responses from the user, taking a few weeks for minimum control over the two control configuration without any support.

This work aimed to develop the User Trainer Interface, a solution to the issues mentioned before. Due to the distinctive method of control from the two modules the User Trainer Interface contemplates two approaches, one for the IK configuration and another for the Torque configuration. The new interface exhibits multi windows presentation, guidance tools and custom UIs. These features were implemented to improve the user performance with personalized settings while also training him to achieve better control over the platform more quickly.

This chapter presents the approach for the IK control configuration. The IK User Trainer Interface (IK UTI) features a visual guidance tool named GHOST to assist the user in the find of the boundaries of the robot, as well the force feedback generated during the movement. IK UTI also features several custom UIs to help configuring the device's inputs and outputs to the user individual preference, among other settings.

4.1 IK User Trainer Interface

This section introduces and explains the properties of the visual guidance tool "GHOST", how it was found its range of motion as also how it moves independently of the user control. The approach to the interface and all the custom UIs made for supporting the user and interact with the inputs and outputs of the simulation are described with a detail description of its control.

4.2 Concept for the UTI

With full body model of the robot the stability of the decreases, due to the great increase of mass in the model producing higher inertia. New user have to undergo with several trials in the simulation before understanding the new boundaries of the robot workspace.

To reduce the numbers of trials needed, a visual guidance tool is implemented to help the users understand the limits of the boundaries.

The increase of the mass also produce bigger shifts in the CoP position, thus making the force feedback oscillate its magnitude significantly. This oscillation often causes disturbing on the user hand will controlling the model, which in turn produces even more oscillations in the simulation. The shifts in the Cop position can be reduce if the user has a optimize control over the simulation. This optimization can be achieve trough a personalize configuration of the inputs and outputs of the simulation.

Finally all the components mentioned previously have to accessible by the user in simply and intuitive manner, thus presenting a user friendly interface.

4.2.1 GHOST

Due to the linear control provided from the IK mode, a visual guidance tool is more appropriated to create kinesthetic memory of the range limitation and the force control for the feedback.

The visual guidance tool, should be able to dynamically show the user the boundaries of the stabled configurations of PHUA, while controlled with the IK mode. At the same time, it should demonstrate the path and positions to perform some basic maneuvers.

These maneuvers are produced at a standard speed, providing the user with a target that can be easily followed. While following GHOST the operator can experienced the gradual increases/decreases of the force feedback, as it change throughout the movement, thus understanding the significance behind the amplitude and direction of the force synthesized. Upon reaching the extreme position displayed by the guidance of the visual tool the user can experiment the force that represents an unstable position, further movement from that acknowledged position could jeopardize the balance of robot making it to fall.

The visual guidance tool was placed on the same position of PHUA and its named GHOST, as its shown on Figure 4.1.

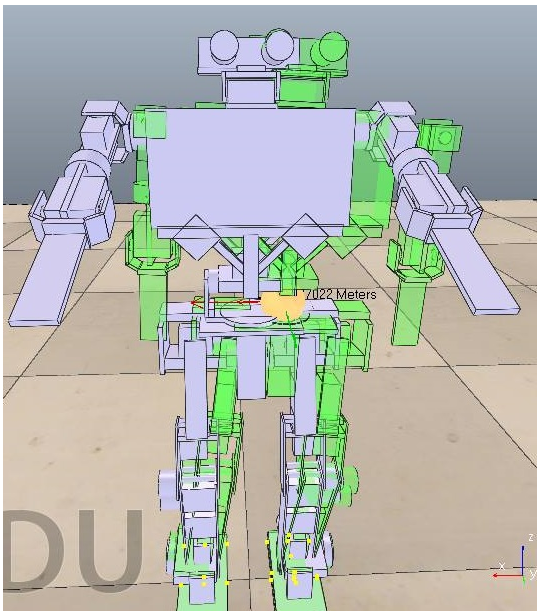


Figure 4.1: PHUA whit a leaning GHOST

GHOST's Properties

GHOST is a replica of PHUA with no dynamic properties, thus it's not able to respond to gravity and collisions. However it can measured kinematics proprieties, detect collisions without affecting the dynamics of the others bodies, renderable and detectable by cameras and sensors.

Since it cannot move through the reaction force on the floor like PHUA, it was necessary to implement new IK elements that allowed this new model to move around its waist similar to PHUA. These new IK elements have the feet as the base and the waist as the tip of the IK chain. To turn the model respondable to this kinetics, a new approach to the hierarchy of the objects that construct GHOST was necessary. To have a center reference of the whole body, the new hierarchy has a central part of the model as the main parent, where the base of each feet are the direct child of this part, the left foot tree is built until the head, where the right part only reaches the waist to save processing power. Figure 4.2 illustrates the new hierarchy implemented on GHOST.

Figure 4.2: Hierarchies for the lower body of GHOST

GHOST displays a green transparent color so it can easily merge with the solid PHUA making it easier for the operator to synchronize PHUA with GHOST while it performs the selected maneuvers. The green color was chosen due to its psychological proprieties on the human being. Studies have shown that the green color can transmit a safe environment to the user's subconscious. This event is present throughout the day by day routines: the traffics lights display the green color when allowed to move and almost every approval stamp has the color green in it. This setting gives more confidence and relaxation to the user while following GHOST, providing a smoother teleoperation during the training process. Figure 4.3 presents the full model of the GHOST.

Search for GHOST's boundaries

It is essential to know the border limits to where its stable to move PHUA in IK control configuration, in order to program the GHOST accordingly, while performing its routines,

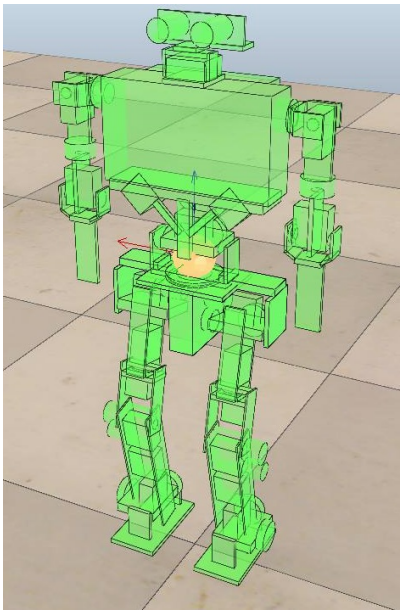


Figure 4.3: Full body of GHOST

thus guiding the users to the furthest stable position of the maneuver. To understand these boundaries, a study was conducted with several subjects that were unfamiliar to the control of PHUA. The study objective is to retrieve the extremity points of the boundaries reached by each subject. The subjects were instructed to reproduce the same maneuvers done in the previous work [1] and hold PHUA on the farthest position that they considered stable, based on the force feedback and the little experience acquired while performing the study. It was analyzed the extremities of the sagittal and frontal plane, where the respective coordinates would be registered. These coordinates are the X component for the frontal plane and the Y component for the sagittal plane. The Z component is not accounted for since it is already limited by default to 0.4m and 0.35m in the settings of PHUA, making it impossible to exceed the workspace, also the force feedback is only synthesized on the XoY plane, displaying a null contribution of the Z component. Although the X and Y components also have a limitation input on the *phantom control* module, the implemented bondaries only contemplated the weight of the lower half of the model utilized in the previous work [1], thus presenting a wider workspace than the present full body model.

Throughout the study, it was constantly received feedback information from the users that participated. It was discovered something unexpected, they would shift more to one of the sides in the frontal maneuver. When questioned why, each user affirmed that the force feedback was stronger on one of the sides. This questioned if the feedback was corrupted or simply unstable. But later confirmation of the force graphics shown no problems.

These issues were resolved with the registered data of Subject F, similar to his colleges, he to displayed a favorite side on the frontal plane, although the data indicated that he would shift to the other extremity. Subject F only differed on his dominant hand, since he was left-handed. Table 4.1 presents the retrieved data from the each participant of the study.

Table 4.1: Data acquired from the study

Subject	Frontal (m)	Sagittal (m)
A	+0.042	+0.028
	−0.040	−0.025
B	+0.043	+0.035
	−0.035	−0.020
C	+0.045	+0.038
	−0.040	−0.022
D	+0.042	+0.036
	−0.038	−0.035
E	+0.052	+0.040
	−0.039	−0.040
F	+0.028	+0.028
	−0.042	−0.024

Evaluating the force feedback and the knowledge that left-handed users shifted to the opposite direction, it was concluded that the stronger force mentioned by the subjects was produced by the configuration of the grip of the user on the pen of the PHANToM. The user displays more support to one side than on the other side, which causes that the force would be distributed by only one finger (thumb) on one of the extremities producing a weaker support.

The study displays the great need of a visual guidance tool like GHOST to solve the uncertainties between different users, that causes different perceptions of unstable robot states. The interpretation of the force feedback varies for each operator, thus becoming difficult for different operators to explore the respective workspace of PHUA.

To resolve this complexity, the operators are guided to the respected position, and consequently they perceive the magnitude of the force produce at the current position of GHOST. This method allows the operators to experienced each factor of the simulator through their on sensibilities and gain the required kinesthetic memory of key configurations.

With the data acquired from the study, it was implemented the boundaries for GHOST that are shown on Table 4.2.

Table 4.2: Boundaries for GHOST maneuvers

Subject	Frontal (m)	Sagittal (m)
GHOST	+0.040	+0.030
	−0.040	−0.025

These boundaries were set so that the values would be symmetric on the frontal plane, although the positive values on the sagittal plane (towards the front of the robot) were decreased a little, due to inertial properties and the fact that subjects did flex the knees, when it is meant for GHOST to produce a pure movement on the assigned plane.

GHOST's Maneuvers

Base on the boundaries acquired from the previous study it was associated a non-threaded script to GHOST that would allow it to dynamically move to the latter boundaries points. To make the movement fluent to mimic the standard maneuvers, the script captures the handle for the main target of the IK elements of GHOST and retrieves its global position in the simulation. According to the type of maneuver selected the script adds small increments to the target's position. The increments are added to a single direction, so that the movement can represent a linear trajectory. They were adjusted so that it would describe a smooth transition in the 50 ms step of the simulation.

As mentioned in the introduction of GHOST, it starts at the same initial position as PHUA. When selected a frontal maneuver for GHOST, it will move to the left extreme border point. Upon arriving there, the increment will change direction, moving GHOST to the other extreme point of the boundary. After achieving that position the increment inverts again making GHOST move to the opposite direction until reaching its initial position. A full course is the name given to the complete movement previous described, and a full cycle represents five full courses.

When selected a plane of movement for GHOST, the scripts initializes the maneuver for a duration of a full cycle. Upon executed the selected maneuver the script is unable to receive a order for a new maneuver.

For design and simplicity a custom UI with two buttons was added to the simulation that allowed the user, with a simple click, to activate the GHOST maneuver cycle in the selected plane of movement. A detailed description of this custom UI is presented further in this chapter.

The initial script created for handling the movements of GHOST was a threaded script. This is the most logical approach since it would be able to insure loops and functions that would simplify the control over it. Unfortunately when the GHOST model was placed in the same simulation as PHUA, the movement implemented on GHOST was too delayed with little response time, due to the others heavy processing non-threaded script existing in the simulation. The cascade execution order of the child scripts in V-REP made the threaded script execute in a small step when paired with several others non-threaded child scripts. To solve the problem the non-threaded version was implemented.

GHOST's Evaluation Tool

With the GHOST model and the movement created, the guidance tool is completed now it is necessary to create an evaluation tool that could express if the user has been able to complete his/her training or if he/she needs more practice, for this objective it was implemented a distance calculation in the simulation.

For evaluation purposes, a distance module was created. It was assumed that the distance module would calculate the minimum distance between a newly added dummy object on the waist of GHOST and the CoM of PHUA. This new dummy on GHOST was added to express a position close to the CoM of it, since it is a massless model, it is impossible to calculate its CoM through an algorithm like the one on PHUA. Thus, it is placed in similar coordinates as the initial CoM of PHUA. The dummy on GHOST doesn't assume the initial coordinates of the CoM of PHUA due to its dynamics properties, the initial value for the PHUA's CoM isn't stable, it sways within a certain range, sometimes, due to delays on the simulation, initializing with null values. If the GHOST's

dummy were to assume those same positions in each initiation of a simulation it would display different evaluations due to the fluctuations on both dummies, with that in light the GHOST dummy was positioned in a fixed central point of GHOST within the initial coordinates of PHUA's CoM to minimize variations on the evaluation.

The calculated distance is showed in real-time on a graph window in the simulation and saved by the *recorded data* module in the hard disk for later analysis. The results of the evaluation tool are only by comparison with several types of users. An expert user could hold the distance to a 0.01 meters throughout the GHOST maneuver cycle, a familiar user would hold around the 0.02 meters while the unfamiliar user would get a distance of 0.03 meter or higher. Figure 4.4 illustrates graphical the display of the calculated distance presented in the simulation.

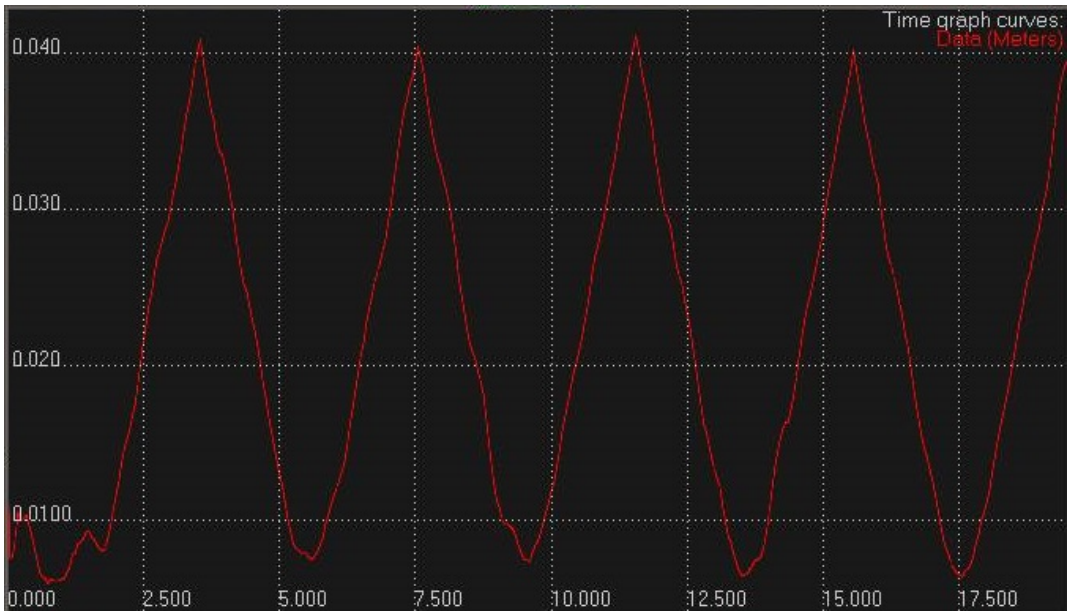


Figure 4.4: Display of the calculated distance on a full cycle while PHUA lays still

4.2.2 Interface

With all the new content added to the simulation the users manifested a need to improve the interface to become more user friendly. For this purpose, it was added new custom UI and new views to the simulation. This new content would allow a better perception of the simulation, as well providing some customization to the inputs and outputs. The personalization over the inputs and outputs of the simulation was implemented so each user could have a unique setting to him, which in turn, would allow for a more refined control and a better overall performance.

Interface Layout

The IK UTI presents a lot of information that the user needs to access or simply view. For this purpose, the main interface was divided in various windows as is shown in Figure 4.5. This is a better approach than to the simple solution of adding floating windows,

because there is no covering of information and we get a steady, good resolution window to display each intended information. The partition referenced by 0 acts as the main display where it is shown the caption from the *DefaultCamera* and where all the custom UIs are placed. The partitions 1 and 2 leave two blank window where the user can display info.

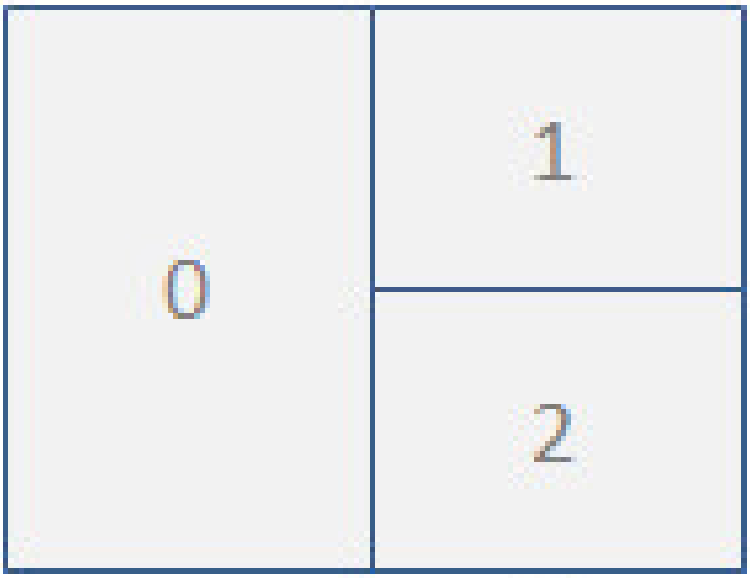


Figure 4.5: New Layout of the Inteface

The standard setup has an auxiliary camera feed shown in the window 1 and in the window 2 it is shown the graphic of the calculated distance between PHUA and GHOST. This setup allows the user to constantly check his performance in the graph window. The auxiliary view allows the user to view shifts in another plane besides the one that it is desired to be moving. Both the default and the auxiliary camera are scripted to position them self in the correct view of the selected plane of action, i.e., if selected the frontal plane the DefaultCamera will automatically be placed on front of the PHUA, while the auxiliary camera will be placed on the side of the PHUA viewing the sagittal plane for possible fading errors. Figure 4.6 illustrates the full display on a running simulation.

Other options are available for the auxiliary windows. There are two non-threaded child scripts, made to view the real position and velocity of the end-effector of the Omni device. Those are tools to evaluate the Omni performance with aspect to the selected input and output options. Although these scripts are disabled in the standard setup due to the overwork of the processing power of V-REP. This processing load would manifest in a delayed simulation after some run time, which is considered inadmissible in the UTI where a real time response is needed.

Custom UI

Being able to build custom UIs is really useful to develop a user friendly interface, a total of four UIs were built for the IK UTI, as follows:

- Maneuver's UI;

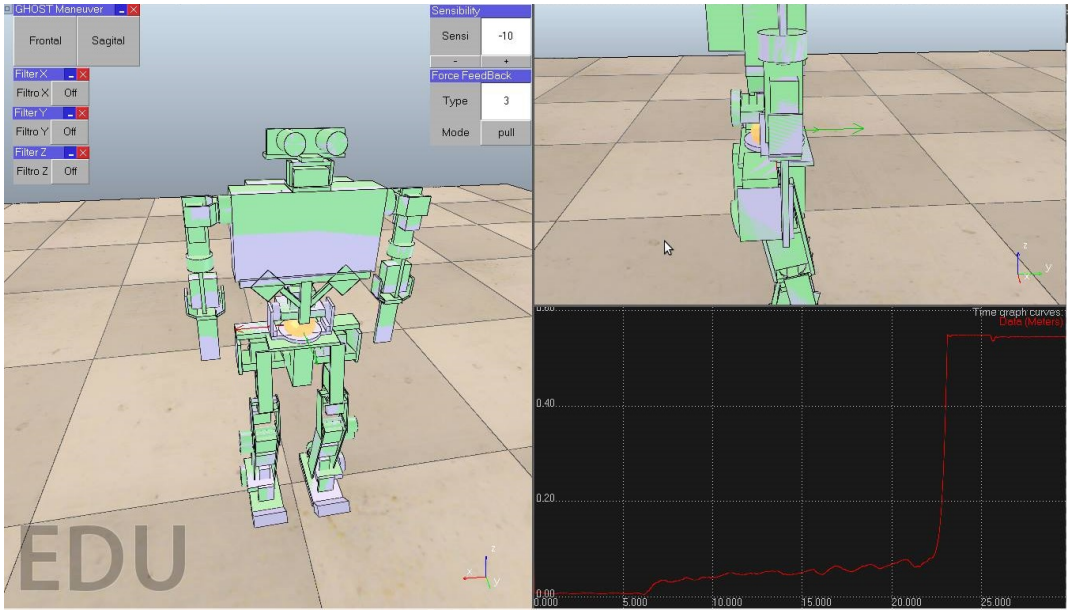


Figure 4.6: Full display of the new interface

- Filter's UI;
- Sensibility's UI;
- Force's UI.

As mentioned before, a UI responsible for the activation of GHOST movements was development, displayed on Figure 4.7. It has a standard visual assembly, with a title bar, a minimize/maximize button and a close button. Moreover, it has a frontal button and a sagittal button. The last two were made bigger by merging various buttons in to one. When clicked, the buttons issue there button number according to its UI that can be read with the appropriate API. In each iteration, the GHOST's script evaluates if that UI's button action is triggered to activate the loop responsible for the selected maneuver. Even if triggered again, nothing would happen due to the script's implemented loop. Only after the maneuver has ended is that the script will be able to be influenced by that specific UI again.

Figure 4.7: Maneuver's UI

The next custom UI is the *filter's UI* that actually is constituted by three separate customs UIs. Due to allocation purposes instead of having one big UI with all three

filters, it was developed a UI for each axis(see Figure 4.8). This UI is responsible for canceling/allowing the control of the Omni device in a respective axis of movement. Isolating the movement in certain directions is a important mechanism, that allows the operator to easily experience and reproduce linear maneuvers. Similar to the *Maneuver's UI*, it has the standard tool bar, plus a label box and a button. This two were made larger by the same method of the previous UI. The label box is to specify which axis is assigned, while the button's label changes between On/Off in contrast with the local variable in charge of communicating with the *phantom control* module that controls the input of the Omni device on V-REP. When the button is pressed by the user, the GHOST's script detects the action and changes a local variable responsible for that filter (this variable assumes the values 1(off) or 0(on)). At the end of each iteration the script transforms the variable's value to a signal and publishes it on a topic. The responsible node uses that value to active or null the Omni's contribution on the assign axis. All filter's UIs start the simulation in Off mode.

Figure 4.8: UIs for the activation of the movements filters

The Sensibility's UI was designed to interact with the input of the Omni device and change its intensity. The Sensibility's UI has no minimize or close button since it is an important information display, being advisable to not let it out of sight. Apart from the title bar this UI has a label box, an edit box and two buttons. The label is just for context, while the edit box is to input values from -17 to 17 (only taking integer). The user can input the number directly to the edit box or he can use the two buttons to lower or raise the value by one value. At the GHOST's script the edit box content is limited to the respective range, being responsible for reading the actions from the UI and converting the edit box content to a signal itself and published it on a topic for later access from the *phantom control* module. In the previous work [1], the position captured from the end-effector of the haptic device was divided by 1700 so it could have a reasonable scale in the V-REP simulation. The scalar was changed to 1701 minus the signal from the UI amplified by 100. This will allow the user to nullify this scalar or double it, covering all the values in between. This approach refines the control of PHUA by a great value, not only the user will be able to select a configuration for his motor aptitude, but he could also change between different sensibilities depending on the desired movement refinement.

The last presented UI is the Force Mode UI, this UI was developed to toggle the new equations of the formulation of the force feedback detailed in further along the chapter.

Figure 4.9: UI responsible for the sensibility of the Phantom

Apart from the intensity of the force, some users called attention that it would feel more natural if the force's direction pointed towards where PHUA is tending to move/fall instead of it being pointed to where the user should locate PHUA for a safe position. This methodology states that the user would be more familiar to the falling sensation and unconsciously would know what to do or where to move. There wouldn't be a force contradicting the intended movement from the beginning. In contrast, using the standard method, the user when presented with an unfamiliar situation would only need to "follow" the force to a safe position.

Similar to the Sensibility's UI, the Force's UI only displays the title bar. Furthermore it has two label box for context, an edit box that accepts values from 1 to 4 and a button. This UI is responsible for the selection of the formulation to use on the force feedback as well the direction that the force is synthesized. The values inputted on the edit box are transform to a signal and then published to a topic where the *haptics feedback* module subscribes to it. Depending on the message received it will select the type of equation use for the formulation of the force: 1 for the standard version, 2 for the stronger beginning version, 3 for the stronger beginning and smother raise version and 4 for no force at all.

The button on the UI is use to toggle between this two concepts during the simulation so that the user can test and choose which one is more suitable for him. The simulation starts with the standard mode with the button presenting "pull" on its label, indicating that the force would pull the user to a stable position. Upon clicking the button the GHOST's script reads this signal and changes a local variable and the button's label itself to "push", indicating that the force now will push the user to where the PHUA is tending to move. The local variable changed by the button is transformed to a signal and published on a topic where the *haptics feedback* module would subscribe and change the output of force.

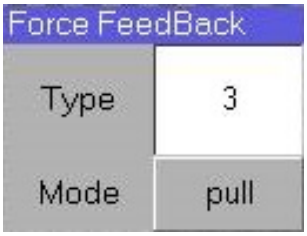


Figure 4.10: UI responsible for changing the type of force.

All the previous functions are enabled and expected to be used with the simulation online. The only condition needed to be known is, when changing the Sensibility input, do it while not controlling PHUA for calculation and references proprieties, or PHUA

will move even if the user didn't made a movement command.

The configuration is only presented when an unidentified user is operating the interface. Afterwards the values of the sensibility, the force formulation and the force direction are recorded by the *archiver* module to the hard disk. This module will upload the last recorded settings of the identified user to the V-REP simulation, thus always providing continuous support.

4.3 Formulations for the Force Feedback

Currently, the formulation of the force feedback is based on the position of the CoP relative to the robot's stability point and its distance to the robot's support base boundary. This was a new concept developed in the previous work of the project [1] to transmit the unbalancing potential of the current position of PHUA to the user through an haptic interaction. This force synthesizes had a single formulation to it, for both the IK mode and Torque mode.

While working with several test users for the previously mentioned studies, it was acquired feedback on several components of the platform. One of the biggest issues was the unique synthesis of the force feedback. Several users where struggling with the sudden upraise of the force or the lack of it in the shorter distances. This issue is inherent to the human nature: there aren't two equal person with the same motor skills. Therefore, a controlled force to one user can translate to a disrupted control due to excess of force to another user or a non perceivable warning to others.

Answering to all of this diversity is nearly impossible, but finding new settings to accommodate most of the needs its possible. Therefore, instead of having a single formulation for the force feedback, a new configuration is created to enable a total of four formulations that are commutable through the custom UI inserted in the simulation. The new configurations are the following:

1. Standard version
2. Quicker beginning version
3. Quicker beginning with a smoother rise on the final threshold version
4. No force at all.

Two different types of equations were obtained by a distribution of points in the space that would transmit the desired behavior for the the new formulation. Utilizing the curve fitting toolbox of Matlab it was possible to obtain the equation of a curve that describes the distribution.

The *standard version* refers to formulation created on the previous work, as follows:

$$F_1(n) = \begin{cases} 0, & \text{if } s \in [0\ 5[\\ m_1 \cdot s - b_1, & \text{if } s \in [5\ 28.449[\\ 1/(\gamma_1 \cdot (1 + e^{-\alpha_1(s-\beta_1)})), & \text{if } s \in [28.449\ 40\ [\\ 3, & \text{if } s \in [40\ +\infty\ [\end{cases} \tag{4.1}$$

$$F_2(n) = \begin{cases} 3, & \text{if } s \in [0\ 16.89733[\\ 1/(\gamma_2 \cdot (1 + e^{\alpha_2(s-\beta_2)})), & \text{if } s \in [16.89733\ 28.449[\\ m_2 \cdot s - b_2, & \text{if } s \in [28.449\ 51.8972\ [\\ 0, & \text{if } s \in [51.8972\ +\infty\ [\end{cases} \tag{4.2}$$

With:

$$a1 = 0.35, b1 = 40, c1 = 0.2; \tag{4.3}$$

$$a2 = 0.35, b2 = 16.89733, c2 = 0.2; \tag{4.4}$$

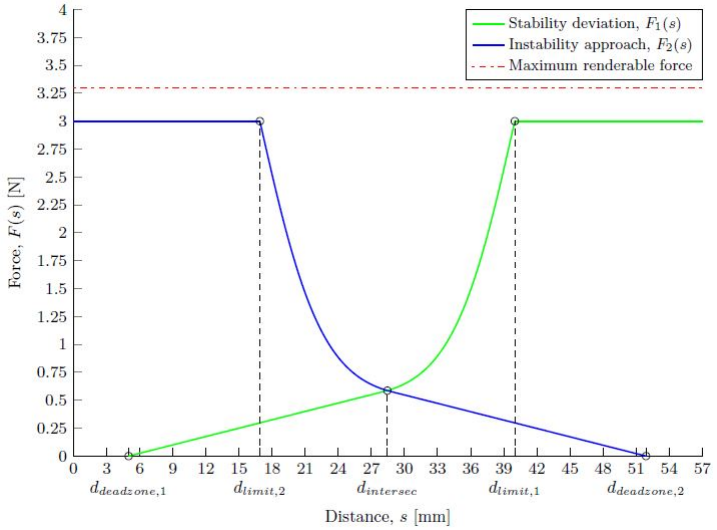


Figure 4.11: Previous Formulation of Force [1]

The new formulations not only reform the domain of each equation but they also apply new behavior to them. The quicker beginning version was implemented to accommodate those in need of a perceptible force at the beginning of the movement for a constant perception of the feedback. This formulation contains the following equations:

$$F_1(n) = \begin{cases} 0, & \text{if } s \in [0\ 3[\\ a_{0,1} + a_{1,1} \cdot cos(s \cdot \omega_1) + b_{1,1} \cdot sin(s \cdot \omega_1), & \text{if } s \in [3\ 28.449[\\ 1/(\gamma_1 \cdot (1 + e^{-\alpha_1(s-\beta_1)})), & \text{if } s \in [28.449\ 40\ [\\ 3, & \text{if } s \in [40\ +\infty\ [\end{cases} \tag{4.5}$$

$$F_2(n) = \begin{cases} 3, & \text{if } s \in [0\ 16.89733[\\ 1/(\gamma_2 \cdot (1 + e^{\alpha_2(s-\beta_2)})), & \text{if } s \in [16.89733\ 28.449[\\ a_{0,2} + a_{1,2} \cdot cos(s \cdot \omega_2) + b_{1,2} \cdot sin(s \cdot \omega_2), & \text{if } s \in [28.449\ 54\ [\\ 0, & \text{if } s \in [54\ +\infty\ [\end{cases} \tag{4.6}$$

With:

$$a1 = 0.35, b1 = 40, c1 = 0.2; \quad (4.7)$$
$$a2 = 0.35, b2 = 16.89733, c2 = 0.2; \quad (4.8)$$
$$a01 = 0.3254, a11 = -0.331, b11 = -0.01698, w1 = 0.08963; \quad (4.9)$$
$$a02 = 0.3254, a12 = -0.1061, b12 = 0.314, w2 = 0.08927; \quad (4.10)$$

This formulation doesn't change the overall perception of the force, it only creates a earlier grow for a sooner perception.

The next formulation is for the users that would like to grasp a strong continuous perceptible force but, have difficulties managing the sudden change in the magnitudes in the final threshold ($s =$ [28.449 40 [for F_1 and $s =$ [16.89733 28.449[for F_2). The sudden changes on the force within minimal displacement of the CoP would insert a chaotic oscillating movement on some users provoked by the wavering force feedback, so new equations were develop to smoother the raise on the force.

The new formulation is as follows:

$$F_1(n) = \begin{cases} 0, & \text{if } s \in 0 \\ a_{0,1} + a_{1,1} \cdot cos(s \cdot \omega_1) + b_{1,1} \cdot sin(s \cdot \omega_1) + a_{2,1} \cdot cos(2s \cdot \omega_1) & \\ +b_{2,1} \cdot sin(2s \cdot \omega_1) + a_{3,1} \cdot cos(3s \cdot \omega_1) + b_{3,1} \cdot sin(3s \cdot \omega_1), & \text{if } s \in [0\ 37[\\ 1/(\gamma_1 \cdot (1 + e^{-\alpha_1(s-\beta_1)})), & \text{if } s \in [37\ 40\ [\\ 3, & \text{if } s \in [40\ +\infty\ [\end{cases} \quad (4.11)$$

$$F_2(n) = \begin{cases} 3, & \text{if } s \in [0\ 16.89733[\\ 1/(\gamma_1 \cdot (1 + e^{\alpha_1(s-\beta_1)})), & \text{if } s \in [16.89733\ 19.89733[\\ a_{0,2} + a_{1,2} \cdot cos(s \cdot \omega_2) + b_{1,2} \cdot sin(s \cdot \omega_2) + a_{2,2} \cdot cos(2s \cdot \omega_2) & \\ +b_{2,2} \cdot sin(2s \cdot \omega_2) + a_{3,2} \cdot cos(3s \cdot \omega_2) + b_{3,2} \cdot sin(3s \cdot \omega_2), & \text{if } s \in [19.89733\ 57\ [\\ 0, & \text{if } s \in [57\ +\infty\ [\end{cases} \quad (4.12)$$

With:

$$a1 = 0.35, b1 = 40, c1 = 0.2; \quad (4.13)$$
$$a2 = 0.35, b2 = 16.89733, c2 = 0.2; \quad (4.14)$$
$$a01 = 0.7896; a11 = -0.7339; b11 = -0.4735; \quad (4.15)$$
$$a21 = -0.1254; b21 = 0.312; \quad (4.16)$$
$$a31 = 0.07023; b31 = 0.0108; w1 = 0.09706; \quad (4.17)$$
$$a02 = 0.7896; a12 = -0.2053; b12 = 0.8489; \quad (4.18)$$
$$a22 = -0.3179; b22 = 0.1097; \quad (4.19)$$
$$a32 = -0.05401; b32 = -0.04618; w2 = 0.09706; \quad (4.20)$$

This formulation has a stronger overall intensity all across. Although it delivers a overall stronger force, the deviation between values is lower than the standard version, thus providing the user with a continuous strong perception of the feedback. The formulation returns to its sudden rise on its final threshold, because it is considered a really unstable area, so it is intended for the user to use that shock as warning. However at that state the user is already subject to a intense feedback so, although there is a strong rise the user is already tense enough to not be affected by a whiplash effect as it was on the previous formulation.

Figure 4.12 represents all of the formulations available on the haptic interface. With the graphic its possible to verify the different behaviors in separated threshold of the spectrum.

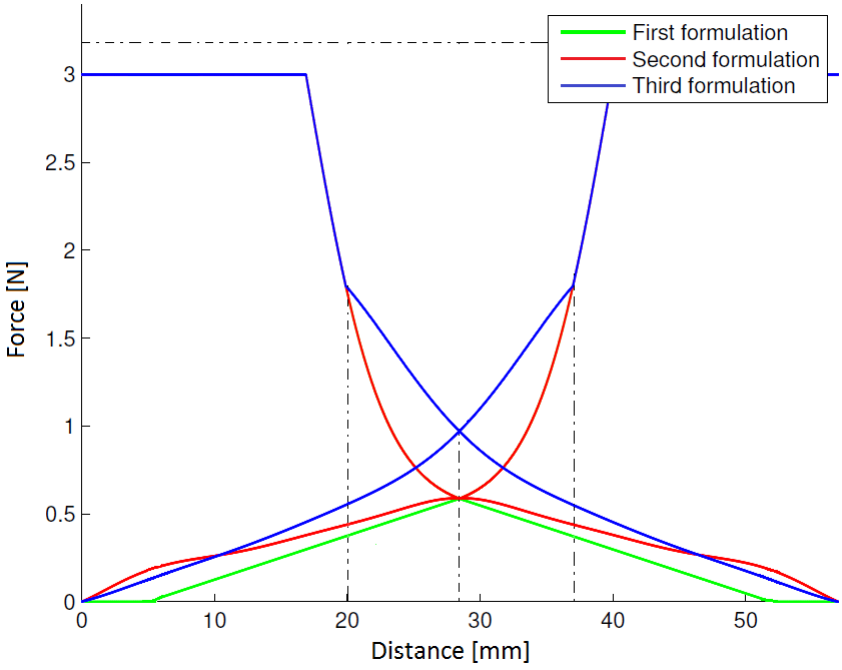


Figure 4.12: All Force Feedback Formulations

The fourth formulation nulls the force generation so the user could understand the importance of the same, toggling between on e off the user can understand the reaction implied by the force simulation.

Chapter 5

Haptic Training Interface for Torque Control

This chapter introduces the approach of the UTI to the complex joint-by-joint control configuration. To reduce the complexity of the control over the platform, a workstation was developed strictly to make the joint-by-joint configuration more intuitive to the operator. The Torque User Trainer Interface was developed to work, in collaboration with the workstation. Identical to the solution presented in the previous chapter, this UTI utilizes a developed guidance mechanism to guide the trainee, while he interacts with the customized interface. Unlike the previous one, this UTI possesses auxiliary tools that indirectly controls the humanoid platform to reduce the complexity of its control. This chapter also presents the new methodology for the force feedback introduced to the joint-by-joint configuration.

5.1 Haptic Interation Workstation (HIW)

The Torque mode exceeds in complexity when compare to the IK mode, the user must control 3 DOF in each hand simultaneously, while the Omni device is placed in a non intuitive configuration. This configuration only complicates more the task of performing motion on PHUA, damaging the performance of the user.

Figure 5.1: Standard Configuration for the Phantom OMNI.

To simplify the matters the Phantom should assume a more intuitive configuration

like the one shown in the Figure 5.2, but to obtain that configuration ,is needed to acquired some sort of support to hold the wished configuration. To answer this needs the Haptic Interation Workstation (HIW) was projected.

Figure 5.2: Vertical Configuration for the Phantom OMNI.

The HIW is a custom workstation with 600mm width, 400mm length and 920mm of height. It supports two PHANToM Omni haptic devices and its able to commute between the standard configuration and the vertical configuration with ease using a trap door system, it contains two custom guiding supports for each Phantom. Apart from others designs, the HIW answers all the ergonomics and mechanics needs, withstanding the forces produced in the feedback without disturbing the workplace, while being easy to use, with a simple screw/nut system to lock/unlock the position of the devices.

5.1.1 Ergonomics of HIW

The ergonomics were a major factor on the development of HIW, many designs were established before HIW but none answered all the ergonomics required. The workstation should be able to present the following ergonomics parameters:

- clear vision to the monitors.
- comfortable gripping in all configurations.
- revert between configurations without effort.

A model for HIW was created using the CATIA V5 software, that same model was subject to ergonomics test using the ergonomic section of CATIA V5. The anthropomorphic data used were the standard deviation of the European population inserted in the CATIA software.

The heights and placements of the parts constructing HIW were adjust to compensate the human model, at the same time it was verified if these new setups wouldn't depreciate the performance of the station. It was verified if both configurations were at the required height, the Figures 5.3 and 5.4 show the approach on both configurations.

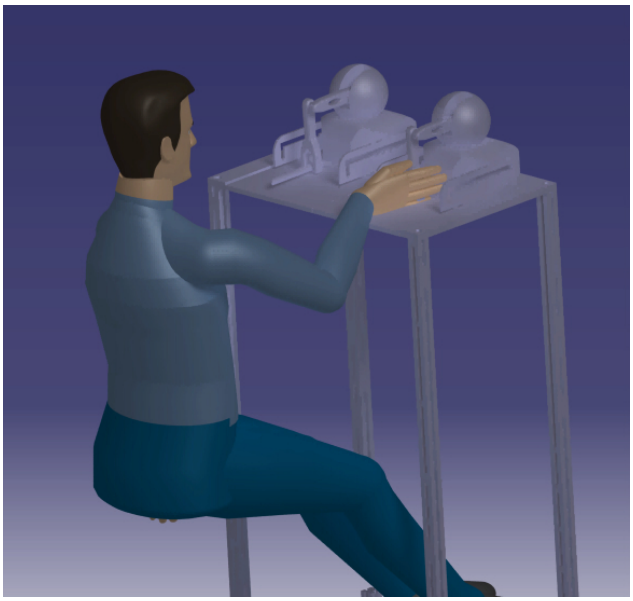


Figure 5.3: Interaction with HIW on standard configuration

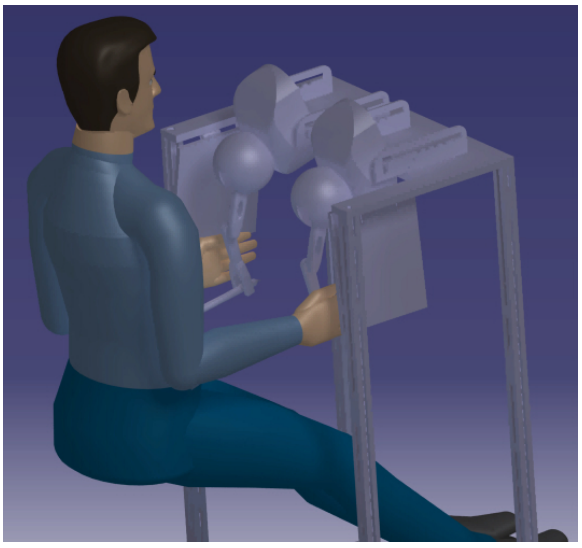


Figure 5.4: Interaction with HIW on vertical configuration

The final design assured all of the ergonomics needs. It doesn't provoke limitation of view to the monitors, it has a simple screw/nut system to release the current configuration, the standard configuration presents the end-effector of the device on a chest height and the fully extended vertical configuration has it placed on elbow height, also with simple steel bolt system to lock/unlock the trap door. All the tools are within the

dynamic anthropomorphic standards so the user can perform all the previous task in the sitting position. This design was also developed to be light weight and easy to carry, so it can be stabilized either with the simulation or with the real PHUA with little effort. Figure 5.5 presents the built design of HIW with one device assuming the standard configuration and the other assuming the vertical configuration.

Figure 5.5: Final Design of HIW

5.1.2 Parts and Materials

Since there isn't a market solution for HIW it had to be manufactured custom parts for the final assembly, these parts were design exclusive for HIW and they were projected using the software CATIA V5.

Not all the parts were custom built, most of the fixation parts were standard material from a hardware store such as 30x30 aluminum profiles for the table legs and infrastructure resistance, stainless steel 30x30 angled brackets for fixating the guiding supports on the table top, steel hinges with spring for the trap doors movement, steel fence bolts for a proper lock system for the trap doors and standard stainless steel screws and wood screws for fixating the several parts.

The custom parts manufactured for HIW consist on the following :

- Table top

- Trap doors
- Guiding supports
- Fixation support

The table top and the trap doors were manufactured from a MDF slab 16mm thick, while each guiding support were manufactured from a sheet 5 mm 220x60 of AW5083-H111 aluminum league, the same league was use on the fixation supports, each of those were manufactured from a 30 mm 50x45 plate. Both the table top and the trap doors were projected to be made from aluminum but the dimensions of the table top were to big to be machined in the CNC machine of DEM. Therefore the parts were projected to be made of MDF so it could be worked without heavy machinery.

The aluminum parts were all manufactured on the CNC machine of DEM, the technical drawings use for the manufactured are located in the Appendix D as also the explosive view of HIW.

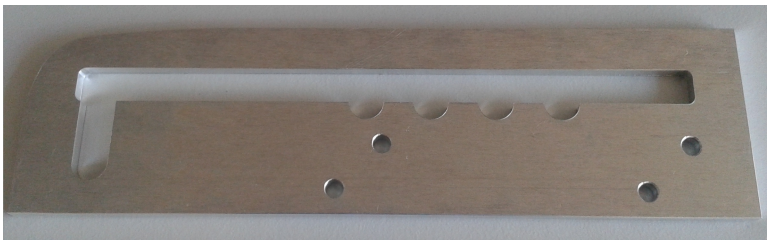


Figure 5.6: Finished Guiding Support

Figure 5.7: Finished Fixation Support

Interaction between Parts

The Guiding supports was projected to withstand the forces and weight of the PHANToM Omni, it also has a guiding channel with several locked positions, including a position for the vertical configuration. These supports were place on the table top and they were secure using the angle brackets.

The Fixation support was projected to replace the weight disks illustrated on Figure 5.8 on the Omni device and secure a M10 screw without the risk of deformity or rapture. As implied these supports were supposed to be attached directly to the Omni device, but that imposed the issue to make an opening in the device's shell to gain access to the M10

screw. During the assembly of HIW it was designed a new approach for the fixation of the device to the workstation. The new approach replaced the massive openings on the lateral of the device with a three small insertion on the bottom of the device.

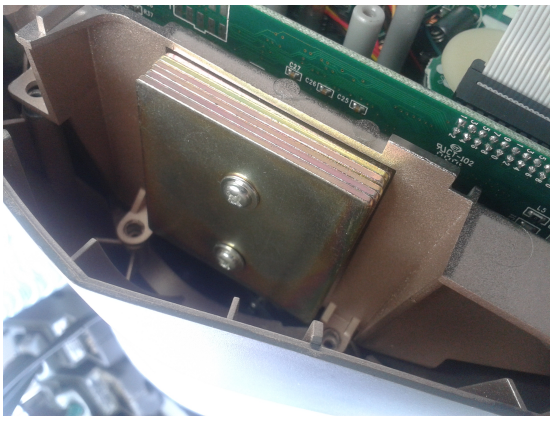


Figure 5.8: Location of the weights disks in the PHANToM Omni device.

On the small insertions on the bottom of the Omni device were secured M4 nuts with the chemical glue Araldite and hot glue. The solidified glue would provide enough endurance for the momentum created on the nuts upon screwing of the screw and the perpendicular forces created during the force feedback. Since the screw is in contact with the device's shell, the parallels forces created in them, are withstand by the shell itself.

The M4 screw/nut system previous described, was used to fixate a projected acrylic slab to the bottom of the device. This slab functions as a support for both the device and the Fixations parts. The Fixation parts are attached to the acrylic slab with a steel 40x40 angled brackets. The brackets are attached to the Fixation parts with M8 screws placed the holes previous designed for the insertion of the part in the device's structured.

A M10 screw acts as the link between the Fixation and Guiding parts, with the support of two M10 nust and two broadband ring to provided a stable and secure fixation. The schematics of the interactions are illustrated in the Figure 5.9.

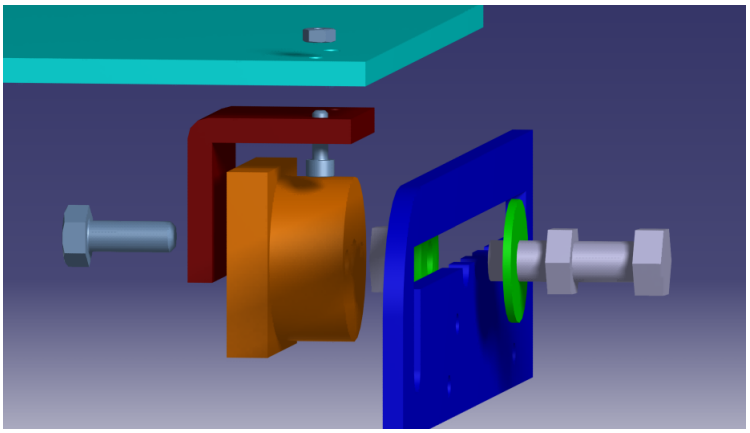


Figure 5.9: Fixation System between the Omni device and the HIW

5.1.3 Terms of Use

The **standard configuration** has the trap door system closed , with the fixation system of the device loss, having the fixation parts settled in the table and the acrylic base settled in a steel bar. The new approach to the fixation system disabled the possibility of configured location of the device, although the height gained and the forward position in the new configuration eliminates any possible collision with the workstation parts. The standard configuration is illustrated in the Figure 5.10.

Figure 5.10: Standard configuration on the HIW

The vertical configuration, utilizes the trap door system open, with the device passing trough it. The device has the fixation system on the final position on the Guiding support, with the device making a 20 degree angle from its vertical position. The M10 screw/nuts system is tight as much as possible. The counter momentum provided by each nut on each side of the Guiding part makes the system locked and stabled, with little possibility to unscrew itself from the assigned configuration. Figure 5.11 illustrates one of the devices in a vertical configuration.

The vertical configuration previous explained doesn't assume a perfect vertical position, this configuration was adopted so that the user could experience more negative movements on the frontal plane, that are fundamental for a most of the human movements. To complement this configuration, an adjustment was made in the configuration of the Omni device joints presented in the Equation 2.3. The adjustment implemented relied in a simple subtraction in the standard joint's position of the device. The new configuration is as the follow:

$$\begin{bmatrix} \theta_{1,phantom} \\ \theta_{2,phantom} \\ \theta_{4,phantom} \end{bmatrix} = \begin{bmatrix} \theta_{2,default} - 0.28 \\ \theta_{1,default} \\ -\pi/2 + \theta_{1,default} - \theta_{3,default} \end{bmatrix} (rad) \tag{5.1}$$

Figure 5.12 illustrates how the inclined configuration provides a larger amplitude to the extended backward position.

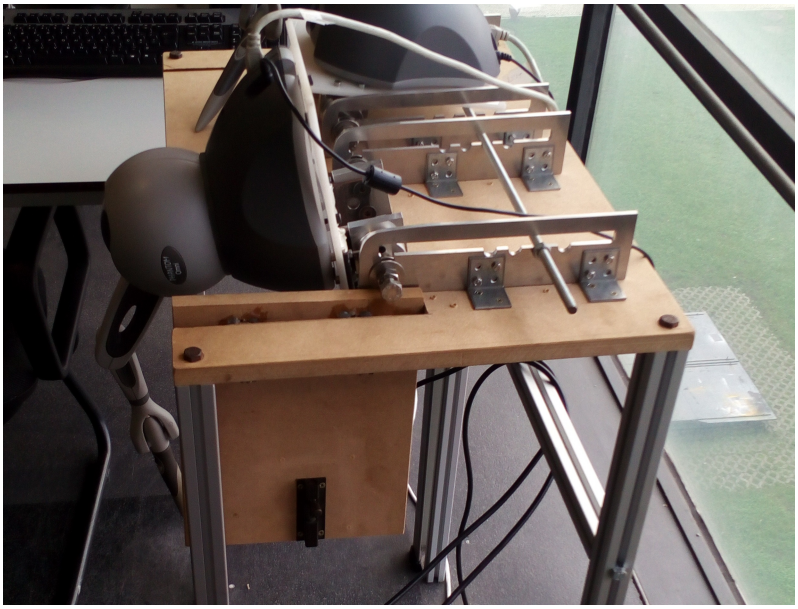


Figure 5.11: Vertical configuration on the HIW

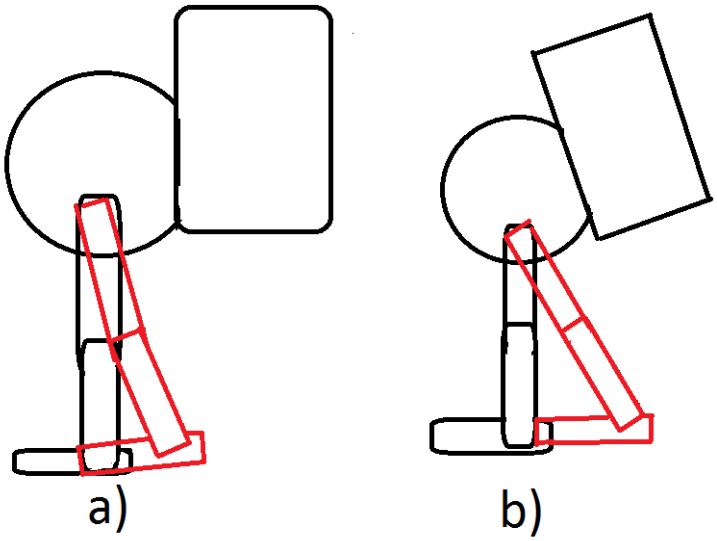


Figure 5.12: a) Standard vertical configuration with the initial configuration at the device's zero, b) Inclined configuration with 0.28rad of difference from the device's zero

5.2 Torque User Trainer Interface

Unlike the IK configuration, the control of PHUA in Torque mode is rather complex and non-linear, due the direct kinematic mapping between the haptic's device and the robot's leg joints. Therefore a visual guidance tool will not have the same practicality as before. The user needs to have vast knowledge of the kinematic mapping between the DOFs haptic device and the humanoid model to perform a smooth maneuver. Even a difference of a few degrees on the joints may provoke an unstable posture, thus causing

failure of the maneuver.

To resolve those issues, a haptic guidance tool is developed to guide the user through the various configurations needed to produce the achieved maneuvers of past works. Similar to the IK UTI the Torque UTI also has a custom interface with new layout and custom UIs to allow an user-friendly interaction with the control of the application.

5.2.1 Haptic Guidance Tool

Due to the elevated complexity of the control of the active DOF's (3 on each leg) and the production of the correct motion and velocity needed to safely perform certain maneuvers, it is necessary a guidance tool capable of instructing the operator of those same conditions. To solve the issue a PI controller was developed to control the torque of the primary joints of the haptic device and guide the user throughout the pretended maneuver.

The principle of the tool changes the control over the system. With the haptic guidance tool enable, the action of the Button 1 makes the robot model to move on its own performing the selected maneuver , in contrast to the previous direct control over the model. Moreover, the active joints on the haptic device guide the user's hands to assume the same configuration as the one shown on the simulation. One of the key features of the mechanism is the synchronization between the autonomous reproduction of the maneuver and the readiness of the user, because, unlike GHOST, it is intended that the user doesn't miss a single configuration, so even if the user releases the button on the device in the middle of the simulation, the replication is stopped, only to resume when the user presses the button again.

This allows the user to aimlessly perform the movements of the same maneuver as the one produced on the simulation, giving him insight on the kinesthetic position, velocity and path require to reproduce the maneuver without the help of the mechanism.

Acquirement and Reproduction of Maneuvers

The maneuvers produced by the haptic guidance tool are, essentially, reproductions of previous maneuvers executed by previous users. It's possible to designate them as replays of prime examples of maneuvers. The selected maneuvers are archived as .csv files in a folder on the home folder.

A maneuver is selected for the archive under certain requirements: it must only show one type of action, meaning that maneuvers reproducing a hip leaning with a considerable flexion on the knees is not acceptable; the maneuver must start and end as close as possible to the initial configuration of PHUA; lastly, the maneuver has to have the minimum of oscillation as possible. If the maneuver meant to be saved is within these requirements, then after detecting the interval of execution of the maneuver, it is manually secure the joints positions from the *recorded_joints_position.csv* saved by the *recorded data* module to the archive folder and given the name *ManeuverX.csv* where the X represents the number of the maneuver. This number is crucial for later selection through a custom UI on the simulation, detailed further in the chapter.

To upload the maneuver back to the simulation the newly *archiver* module accesses the selected maneuver from the archive, where it analyses each line of the csv file, retrieving the position of each correspondent joint and associating it to a vector, producing a vector for each joint of the legs. When the archiver module finishes uploading the whole

file, it publishes each vector to a topic, where the *phantom control* module subscribes to each vector and uploads them to its system. When the Button 1 of the device is pressed the *phantom control* module increments each vector, publishing each configuration to the V-REP simulation.

Upon achieving completion of the maneuver the system resets the increment, beginning again the maneuver. In contrast, when the archiver module perceives a new selected maneuver, it redoes the procedure, wiping clear the previous vectors and increments. Mindful upon uploading a new maneuver when the latter isn't complete, because with the increment re-established the simulated model of PHUA will rush to achieve the initial configuration producing high acceleration and inertia.

It is relevant to mention that due to the 20Hz simulation frame of V-REP and the 1000Hz servo loop of the phantom control module, it was necessary the implementation of a timer so the high servo loop wouldn't skip any vector configuration.

Haptic guidance

The trainer tool is meaningless if the haptic device doesn't guide the user hands throughout the configuration. To execute this purpose, commands that control the torque on the three primary joints of the haptic device are used. These joints are the only one that present active torque and luckly they are the joints responsive for the translation features of the haptic device and the main joints used to control PHUA in torque mode.

A PI controller was developed to control how much torque is sent to each joint of the haptic device. The controller uses the error between the position of the Omni's joint and the position of the correspondent PHUA leg's joint for its proportional control and the sum of the error for the integrative.

Each joint has its own proportional and integrative values due to their difference in weight, and effect on the next joint. The values obtained for each proportional and integrative parameter were obtain through trial and error, experimenting with various configuration until the force felt in the user hand and the movement provoke by it, with constrain on part of the user, were reliable and with few errors from the estimate position. The guiding force has to be high enough to guide the user hand but, at the same time controllable so that it doesn't escape the grip of the user.

The following equations of the controllers allow for the user to know the necessary translation for accompanying the maneuver, guided by the guiding force. The controller is only active when the error measured is higher than 0.01 radians or 0.57 degrees.

$$first\_joint\_torque = 1030 \cdot (-error) - 60 \cdot (-sum\_error) \tag{5.2}$$

$$second\_joint\_torque = 680 \cdot error + 40 \cdot sum\_error \tag{5.3}$$

$$third\_joint\_torque = 480 \cdot (-error) + 35 \cdot (-sum\_error) \tag{5.4}$$

With:

$$error = PHUA_{virtual}\_joint\_position - Omni\_joint\_position \tag{5.5}$$

$$sum\_error = \sum_{k=1}^{n} error \tag{5.6}$$

On each instance the force generated will act as a position guide to the user, becoming similar to the Haptic Guidance in Position methodology. However with the incrimination of the simulation the moving model proves a force in the user's hand guiding him/she towards the intended path, thus providing a guidance similar to the Haptic Guidance in Force. Moreover all the maneuvers utilized in the haptic guidance tool are replication of previous maneuvers teleoperated by previous users, thus the guidance provide has a human component within it, kinda similar to the Remote Haptic Collaboration methodology often used in the medical surgery training.

This tool efficiency isn't restricted to the humanoid platform. The simplicity of it allows that any similar model construction is able to use it. Thus, if managed to achieve a really complex maneuver, the reproduction can be sent to other platform and start the immediate training of user to achieve the same completion. Or vice-versa, highly complex maneuvers achieved by another platform with similar specs to the humanoid platform can be imported and immediately reproduced.

5.2.2 Interface Layout

Identical to the IK UTI, the Torque UTI presents split screens layouts, allowing for visualization of multiple data useful for a good performance. Torque UTI also present the same disposition as the IK UTI, illustrated in the Figure 4.5.

In the window 0 prevails the feedback of the *DefaultCamera*, and maintains as the main window with the custom UIs placed within it. Unlike the IK UTI, the *AuxiliaryCamera* has its feedback broadcast in the resulting window 2 on the Torque UTI, helping users to visualize other planes of movement. The *AuxiliaryCamera* on Torque USI has a constant replacement and reorientation assigned to it, moving constantly in reference to the position and orientation of the DefaultCamera. A non-threaded child script is assigned to it. The script is responsible for gathering the AuxiliaryCamera handle, forcing it to assume a position with fixed coordinates and fixed orientation relative to the DefaultCamera. This allows for both cameras to be almost perpendicular with one and other, offering full coverage of up close models. Figure 5.13 illustrates the complete display of the Torque UTI.

The remain window 1 has a X/Y graph display on it, exhibiting the CoP and feet positions (Figure 5.14). The graphic operates in explicit handling, meaning that its activation and data treatment isn't perform by the main script but individually approached by the user in the child script. This allows to place a constraint in the number of points present at any given time and how many are retrieve in each simulation step. The graphic present three streams of data, two of the streams used are the X and Y global position of the dummy targets placed on the feet of the model, the remain stream is the X and Y components of the global position of the CoP. This last stream isn't present on the V-REP simulation, it is subscribed from the *robot state* module, thus the attribution of it to the graph is done with APIs in the non-threaded child script mention before. With all of the streams set in the graph and handling it so it only shows the last 10 values allows the user visualize, the position of CoP in comparison with the position of the feet

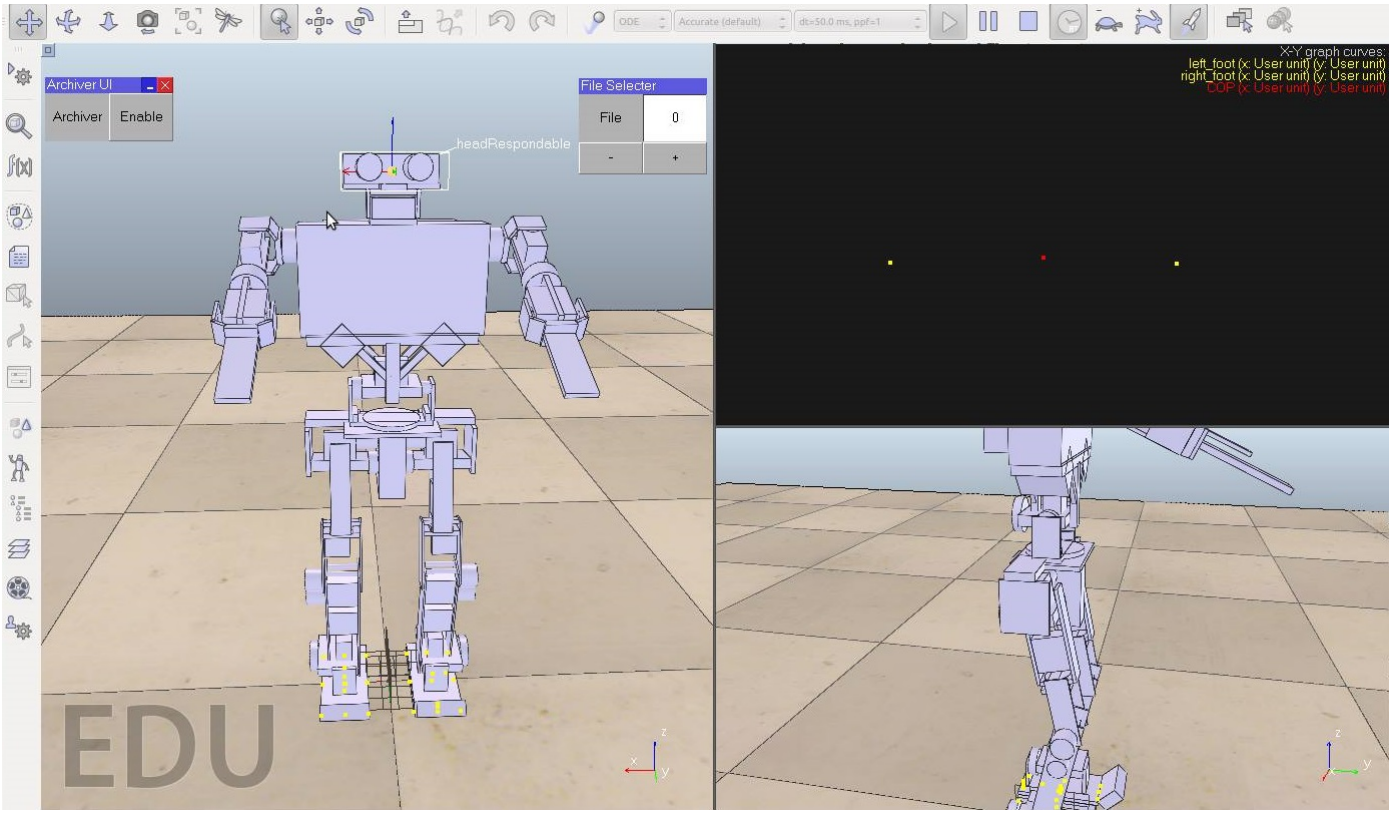


Figure 5.13: Full Interface of the UTI.

with the simulation online. This visual tool is substantially important to the user when performing maneuvers with refine control on weight shifting, *e.g.* raising a foot.

Figure 5.14: Display of the graph exhibiting the CoP and feet positions

Custom UIs

The Torque UTI has only two custom UIs placed in it. The complexity of the control configuration doesn't allow for the same customization present in the IK UTI. Since it is a direct configuration of the device's joint positions with the model's joints, a cus-

tumization of the inputs to the simulator is inadvisable. Any expansion or contraction of the workspace would damage the overall control. Thus, the only UIs developed are associated with the *archiver* module.

The first UI is responsible for the activation of the haptic guidance tool (Figure 5.15) responsible for signaling the archiver module to upload the selected maneuver. It has a simple layout with only a label box and a button. The label box is only used to framework the purpose for the button next to it. However pressing the button publishes a signal to the archiver and phantom control modules enabling/disabling the haptic guidance tool, while also changing the proprieties of the second UI (Figure 5.16) turn it visible/invisible in the simulation.

Figure 5.15: Archiver's UI

If activation of the haptic guidance tool is enabled, the phantom control module disables its control over the simulation and waits for the vectors from the archiver module. Moreover, the archiver module only uploads a maneuvers when a file is selected. Selection of the files is done by the second UI place in the Torque UTI. If deactivated the phantom control module rehabilitates the control of the haptic device over the simulation module.

The second UI is only visible when the activation of the first UI is obtained. When visible the UI displays a label box, a edit box and two buttons. Similar to the other label boxes, its purpose is to reference the utility of the element beside it, the edit box accepts only integers from 0 to the number of maneuvers archived on the archive and the buttons increment the number in the edit box. Both buttons have a assigned label implying the type of increment, positive or negative. If the number of the file is higher than the restriction it turns back to the highest possible number before any signal is sent to prevent the crash of the archiver module. If more maneuvers are indexed to the archive, its necessary to alter the restriction on the edit box, or it won't be possible to access the new files.

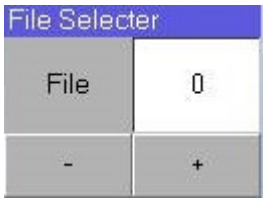


Figure 5.16: Filer's UI

When a file is selected, a signal is sent to the archiver module allowing it to access the respective maneuver. Both of the UIs are controlled by the same non-threaded child script as the AuxiliaryCamera and the CoP position graph.

5.2.3 Auxiliary Tools

Mostly of the simulations end in failure (robot module falls down) due to the complexity of the control of the robot during Torque mode. To reasonably reduces the failures, auxiliary tools were developed to help the user with the simulation. These tools do not affect the control methods over the model, however, they improve the control over it.

Two auxiliary tools where developed, a Joint Configuration Finder and a Assisted Torque Configuration. The first tool, when the haptic device isn't controlling the robot model, utilizes torque from the primary device's joints to place the haptic device with the same configuration as the robot model possesses, enabling always a safe beginning in the control. The second tool supports the user with a indirect control over the upper body joints to establish a more stable configuration.

Joint Configuration Finder

One of the biggest causes for a failed simulation is the beginning of the control. If the Omni device is not correctly positioned, the model would try to assume the incorrect configuration as fast as possible. This behavior would provoke high accelerations and inertia that would crash the model to the ground. Even a small difference of a few degrees on certain joints would provoke failure. With the vertical configuration of the Omni devices, the task gets even harder, aligning the joints correctly while the device is placed in a vertical configuration is nearly impossible for users with little experience on the platform.

To solve the problem the Joint Configuration Finder was developed, similar to the haptic guindance tool previously presented in the chapter. This mechanism utilizes a PI controller to input torque to the primary joints of the device thus moving it until reaching the final configuration. This version is a weaker version of the previous PI mentioned, since it only has to move its own components. The controller uses the error between the position of the Omni joint and the position of the correspondent PHUA leg joint for its proportional control and the sum of the error for the integrative, just like the controller mention before. Although this controller is mention later on the work, it is the source of the controller for the haptic guidance tool. Figure 5.17 displays the devices assuming a complex configuration of the robot's model.

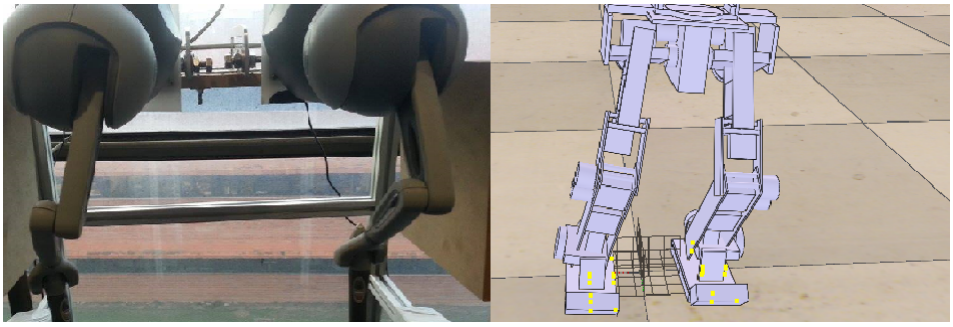


Figure 5.17: Demonstration of the Joint Configuration Finder in a complex robot state.

The equations supporting the controller for each joint are as follow:

$$first\_joint\_torque = 230 \cdot (-error) + 8 \cdot (-sum\_error) \tag{5.7}$$

$$second\_joint\_torque = 280 \cdot error + 12 \cdot sum\_error \tag{5.8}$$

$$third\_joint\_torque = 80 \cdot (-error) + 8 \cdot (-sum\_error) \tag{5.9}$$

The *error* and *sum_error* are calculated through the same equation as before, reference back to equation 5.5 and 5.6 for details.

Before developing the controller, it was made a study to find the torque necessary to implement movement and to stank a joint in a certain configuration without moving. The controller's proportional parameters were implemented with those values in attention. Just like the other controller, these controllers have a safe zone, being only active with errors larger than 0.01 radians (deadzone).

Assisted Torque Configuration

The Assisted Torque Configuration was implemented on UTI the control the upper body during maneuvers, thus improving the balance of the robot model. For that purpose, it was implemented a new IK module involving the joints of the torso and neck. Only the three marked in the Figure 5.18 were changed to IK mode configuration and used in the IK chain.

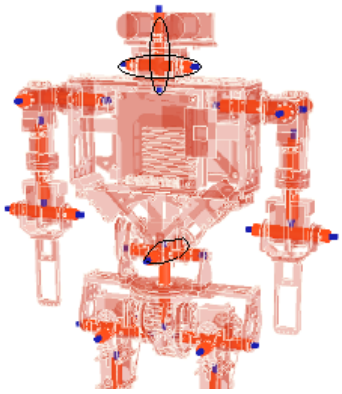


Figure 5.18: Torso's Joints modified to IK mode

The IK element has the center object of the robot, referenced as *PHUA* in the tree hierarchy, as its base, with the tip dummy object place in the same position as its parent object, the *headRespondable* object, the target dummy is placed initially in the same position as the tip dummy. Figure 5.19 illustrates the hierarchy of the objects involve in the upper IK chain.

As previous mentioned the *torso_joint*2 joint does not belong in the IK calculation. This configuration was approached due to the unstable behavior provoke by the Y component of the IK element, even if, rather important, the Y component solver of the IK element was disable and with it the joint responsible for the movement. The head joints were left active in the IK element, because the connective work done by both can resolve small oscillations without evolving the movement of the heavy mass of the torso.

With a non-threaded child script is was possible to dynamically move the target dummy of the IK element in contrast of the inclination of the the hip of the robot

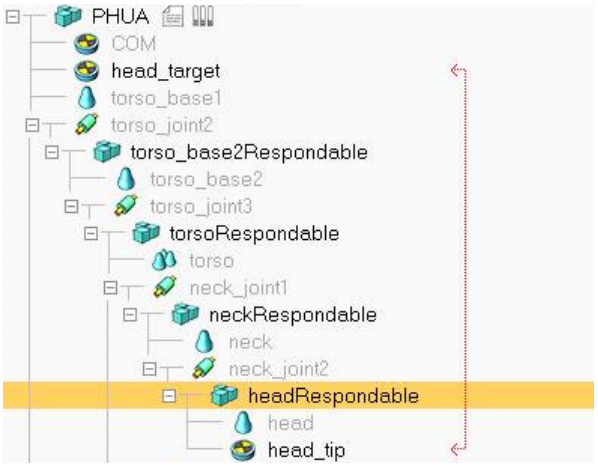


Figure 5.19: Hierarchy of the IK element of the torso

towards its model. To analyze the hip's inclination a relative orientation between a dummy in the right foot and the hip object was implemented, since the feet are always parallel to the floor the resulting orientation express the rotation of the hip from its standard orientation in the model. The pitch component of the resulting orientation is analyzed and converted to a translation that is implemented in the target dummy of the IK element, thus producing a responsive lean of the torso in virtue of the hip's inclination. Some examples are illustrated in the Figure 5.20.

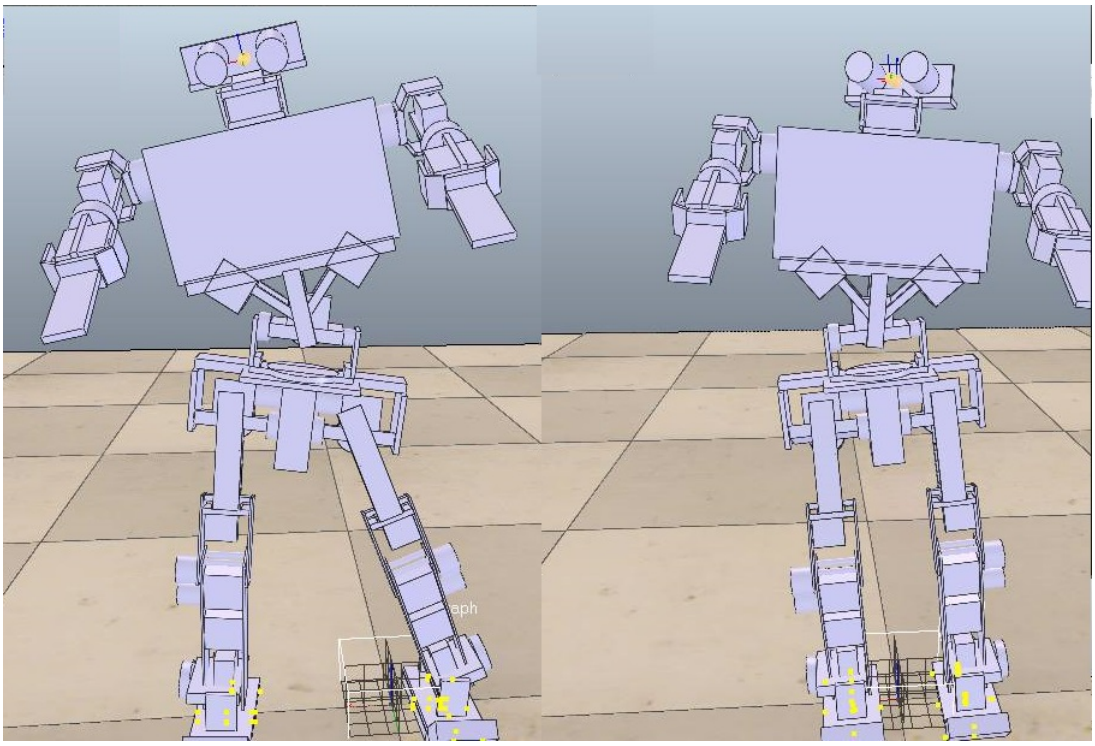


Figure 5.20: Examples of the Assisted Torque Configuration

Apart from the dynamic increment in function of the the hip inclination, the target dummy of the IK element is forced to assume a position of 0.2413 meters higher than the global position of the hip and its same Y coordinates of it. This ensures that the target always follows the body in its global position. If these restriction weren't made the dummy target would suffer the same transformations as it parent, resulting in deviations of its trajectory.

5.3 Force Rendering for the Torque mode

The presented bi-dimensional force formulation, can only manifest feedback in the XoY plane and, due to the rotation of planes of control, the Y axis of the IK mode is the Z axis on Torque, thus producing misguiding guidance. Moreover the present formulation transmits a linear manifestation due to its vectorial application, where to in the Torque mode assuming a safe configuration may not be so linear.

The versatile control provided in the Torque mode allows the user to produce more complex maneuvers. Some of those maneuvers, *e.g.*, lifting a foot, balancing in one foot, make the CoP assume a position in the border of the robot's support base. Others even necessitate to project the CoP over the projected area, *e.g.*, walking. All the previous situations would express a full magnitude force vector with the present force formulation, becoming unreliable and uncomfortable for the user during the teleoperation.

The new approach utilizes a formulation of force by directly applying torque to the joints. This method allows the generation of tri-dimensional forces, thus providing information to user about the flexion and how it is affecting the balance.

The formulation uses the real-time values of torque present in the joints of the simulated model, in function to the distance between the CoP and the foot of assigned leg, to generate the correspondent torque of the device's assigned joint. Only the first three joints of the Omni haptic device are active. Thus, the force feedback only utilizes the hip roll, the hip pitch and the knee roll joint's torque of the respective leg to manifest the feedback.

This configuration allows the user to perceive dynamics elements like the weight of the robot manifested in the knee joints, or the collision with objects manifested on high torque values. This elements allow the user to emerge in the haptic experience with a more intuitive approach, with force reaction common to the human experience.

The equations of the formulation are the following:

$$Omni\_first\_joint_{torque} = PHUA_{virtual}\_hip\_roll_{torque} \cdot 2000 \cdot relevance \quad (5.10)$$

$$Omni\_second\_joint_{torque} = PHUA_{virtual}\_pitch_{torque} \cdot 3000 \cdot relevance \quad (5.11)$$

$$Omni\_third\_joint_{torque} = PHUA_{virtual}\_knee\_roll_{torque} \cdot 500 \cdot relevance \quad (5.12)$$

As shown with the equations, the torque generated is based on the torque read form the joints model multiplied by a simple scalar to fit in the Omni device's interval of

values allowed, and by a value called *relevance*. The relevance value is calculated from the distance between the COP and the assigned foot, in the equation 5.13:

$$relevance = \begin{cases} a_0 + a_1 \cdot cos(COP\_Foot_{dist} \cdot w) & \text{if } COP\_Foot_{dist} \in [0\ 80\ [\\ +b_1 \cdot sin(COP\_Foot_{dist} \cdot w), & \\ 0.3, & \text{if } COP\_Foot_{dist} \in\ [80\ +\infty\ [\end{cases} \tag{5.13}$$

Whit:

$$a0 = 0.7638; \tag{5.14}$$
$$a1 = 0.4317; \tag{5.15}$$
$$b1 = -0.5074; \tag{5.16}$$
$$w = 0.01859; \tag{5.17}$$
$$\tag{5.18}$$

The COP_Foot_{dist} isn't a simple distance, when calculated the component Z is given five times more importance. This choice is made due to the fact a lifted foot will always have far less importance as a support of weight and balance then the grounded foot. Thus, COP_Foot_{dist} is calculated by the following equation 5.19:

$$range_left = \sqrt{(COP\_x - foot\_x)^2 + (COP\_y - foot\_y)^2 + 5 \cdot (foot\_z)^2} \tag{5.19}$$

Equation 5.13 is a simple Fourier function developed using the Matlab to describe the intended behavior shown in the Figure 5.21. Each joint has a different approach based on its behavior. The knee joint suffers the biggest torque variation so it presents a small scalar, while the hip pitch has small perturbations, thus needing a large scalar for the force to be perceived by the user. The values of the scalar were calculated by studying the behavior of each joint on basic maneuver, analyzing peek values and minimums. Refinement was made with trial and error between simulations, until achieving a reliable feedback.

Figure 5.22 illustrates the behavior of the value read and the value produced on the knee joint during a simple maneuver of flexion.

As it can be analyzed the torque generated is unstable, due to oscilation of the control and of the simulation. This unstableness produces oscilations on the control of the user and thus disrupting the performance of the simulation. To smooth the oscilation a low-pass filter was develop to be applied on the torque generated. A study was made to hypothesize the best filter for the application. Data of previous simulations were analyzed using Matlab's filter function utilizing Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters.

FIR filters are digital filters with finite impulse response. They are also known as non-recursive digital filters as they do not have feedback (a recursive part of a filter), even though recursive algorithms can be used for FIR filter realization. IIR filters are digital filters with infinite impulse response. Unlike FIR filters, they have the feedback (a recursive part of a filter) and are known as recursive digital filters therefore [26].

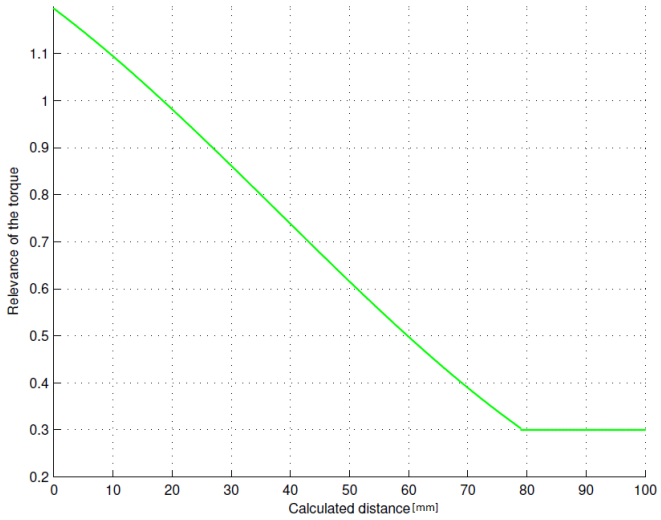


Figure 5.21: Behavior of the relevance in function of the distance of the CoP to the foot.

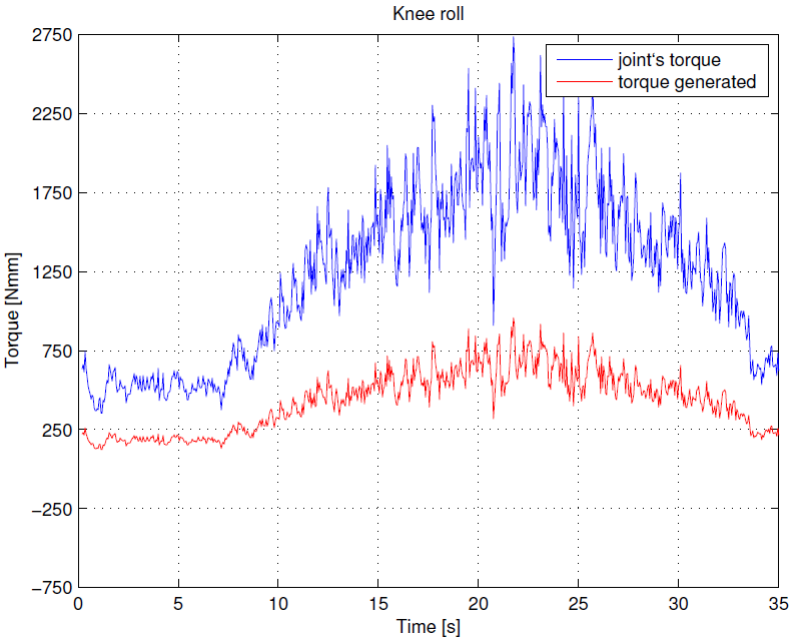


Figure 5.22: Generated Torque in function of the Torque read

With the study, it was concluded that a IIR filter of sixth order would provide the behavior desired. A similar FIR would be of a far greater order, thus delaying the procediment of the force, losing the real-time aspect of the component.

Equation 5.20 was implemented in the C++ script of the *phantom control* module, where the B components represents the array with the last six values of the unfiltered torque multiplied by their respective importance, and the A components represents the array of the last six filtered values with their respective importance. At the start of the simulation these arrays are initialized with zeros. This will produce a small delay on force feedback at the start of the control. This delay was intended, it works as a

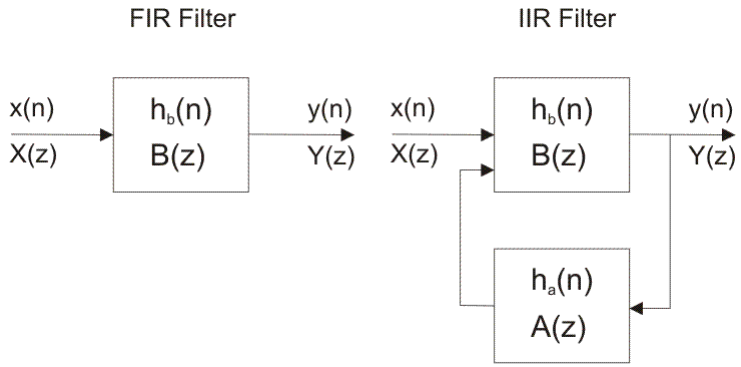


Figure 5.23: Block diagrams of FIR and IIR filters [26]

safety precaution for the user so that he isn't affect by sudden unexpected forces at the initial grip, such as the weight of the robot, instead a progressive build up of the start up force is implemented so the user can successfully counter it. The small increment present throughout the simulation doesn't suffer the same delay providing the real-time experience.

$$new\_y = B - A \tag{5.20}$$

$$B = b1 \cdot x(6) + b2 \cdot x(5) + b3 \cdot x(4) + b4 \cdot x(3) + b5 \cdot x(2) + b6 \cdot x(1) \tag{5.21}$$

Whit:

$$\begin{aligned} b1 &= 2.768871415481655e-08; && (5.22)\\ b2 &= 1.384435707740828e-07; && (5.23)\\ b3 &= 2.768871415481655e-07; && (5.24)\\ b4 &= 2.768871415481655e-07; && (5.25)\\ b5 &= 1.384435707740828e-07; && (5.26)\\ b6 &= 2.768871415481655e-08; && (5.27) \end{aligned}$$

$$A = a2 \cdot y(6) + a3 \cdot y(5) + a4 \cdot y(4) + a5 \cdot y(3) + a6 \cdot y(2) \tag{5.28}$$

Whit:

$$\begin{aligned} a2 &= -4.796681599817807; && (5.29)\\ a3 &= 9.207242375092010; && (5.30)\\ a4 &= -8.840369682500992; && (5.31)\\ a5 &= 4.245786473289920; && (5.32)\\ a6 &= -0.815976680024278; && (5.33) \end{aligned}$$

Figure 5.24 illustrates the new behavior of the filtered force with the non-filter version of the same maneuver as before.

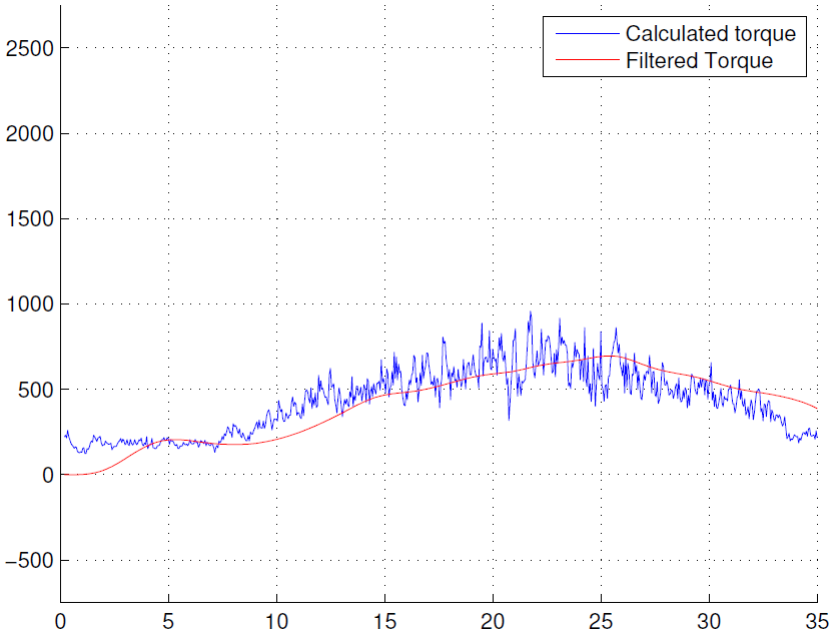


Figure 5.24: Processed Torque with the IIR filter

Chapter 6

Experiments and Results

This chapter details the experiments utilized to test the effectiveness of the User Trainer Interfaces developed, as also the relevance of the haptic feedback on a complex multi DOFs teleoperation of a humanoid robot. The intended stages of training are explained for each tool developed in the course of this work.

6.1 Conducted Experiences

The experiences made in this work aimed to test the impact of the newly developed support for the users in their training. A few persons were invited to participate in the trials of the IK and Torque UTI. The trials for the IK UTI aim to create a comparison of the performance contrast of a user that did not had access to the trainer interface and a user that utilized the training plans and procedures. For the Torque UTI there were made two type of experiences: the first experience consisted on the evaluation of the force feedback and the auxiliary tools developed in this work in users placed in three different configurations of the platform (with haptic and visual feedback, only haptic feedback and only visual feedback); the second experience consisted in testing the efficiency of the haptic guidance mechanism implemented in the Torque UTI and compare it to the user subject to haptic and visual feedback in the latter experience. Finally it was attempted to produce maneuvers of balancing the robot model in one foot.

Figure 6.1 displays a basic scheme with the simulation experiences conducted in this work.

6.2 Application and Evaluation of the IK UTI

The IK UTI has several components to interact with the user, configuring almost every aspect of the simulation control, but if the user starts aimlessly changing the settings he can lose a large amount of time without gaining any useful experience. All the components were developed with a purpose and with an intended way to interact with each other.

The IK UTI is intended to have several steps of interactions to increase its efficiency. The intended steps for an unfamiliar user are as follows: **first step** is to filter all the axial movement apart for the one where the maneuver takes place; **second step**, activate GHOST's movement by selecting the desired plane of movement, but only travel to

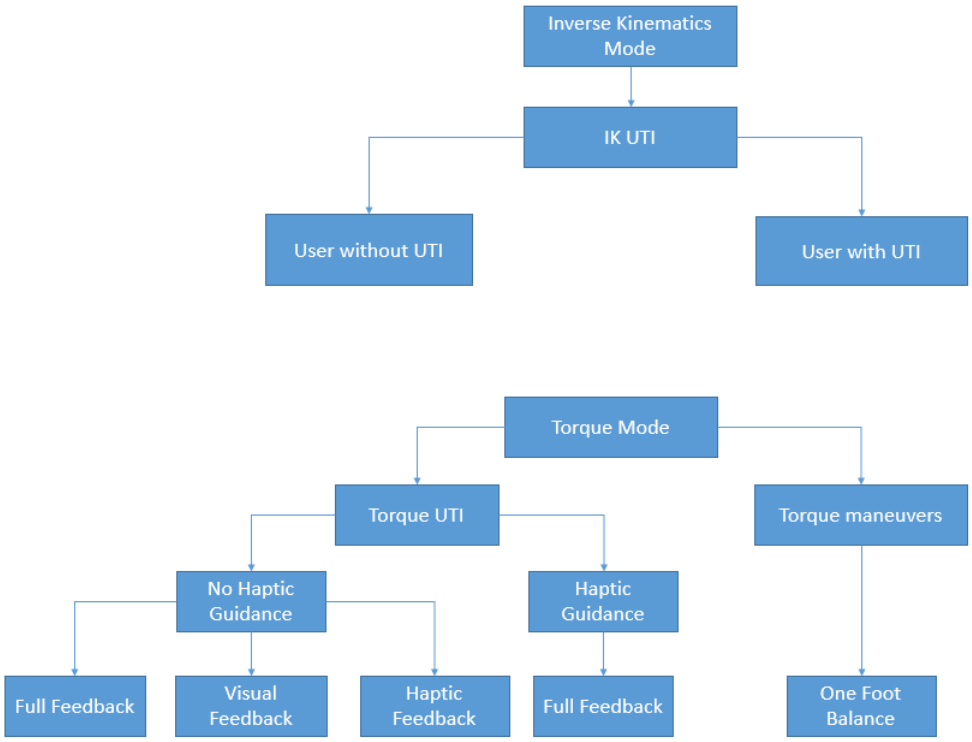


Figure 6.1: Summary of the experiences.

the its extreme positions and sense the magnitude of the force that represents a really unstable position, repeat for both sides of the plane since it was proven that the users have different sensibilities on each side; **third step**, attempt to follow every movement of GHOST throughout its maneuvers; **fourth step**, change the sensibility and force parameters based on the experience gather from the third step and repeat until establishing comfortable settings; **fifth step**, undo the first step and repeat the third step while analyzing the performance from the distance module. If the evaluation of the performance is within the thresholds referred in Chapter 4 the training is complete, if not repeat the fourth step.

The first step allows the user to aimlessly remove all the undesired drifts, thus producing a linear trajectory. The second step forces the user to experiment the extreme positions and thus acquiring the kinesthetic memory for the force and positions of those strategic points. While producing the third and fourth step the user experiences the struggles of the current settings and thus constantly improving the refinement of the interface as well the understanding of the humanoid platform. The fifth and final step puts to test all the experience gather from the latter phases and evaluates the user without any system constrains. However if the user is struggling on achieving a satisfatory evaluation he/she could try to approach the final step with one active filter, only to remove it once the performance gets better.

An experience was made to verify if the training procedure has impact in a unfamiliarized user of the platform. Two subjects were selected for the experience, both of them

without any experience in the platform. Both the subjects had 20 minutes to practice with the humanoid platform in IK mode control, however one of the subject would practice aimlessly having only its intuition, while the other followed the previous training procedure.

After 20min of training they were asked to follow GHOST while a experienced user was controlling it using the *ghost control* module running on the other machine. It was asked that the experienced would freely control GHOST with the condition that he would have to execute the same maneuvers for both the users. The user selected to produce a frontal maneuver in slow speed, followed by a sagittal maneuver with the same speed with no waiting period between maneuver, after completing the sagittal maneuver he positioned GHOST in the initial position for about 1 sec, producing, afterward, another frontal maneuver but with higher speed. Each subject was given three trials to produce the best performance they could.

Figure 6.2 displays the graphic with the recorded distance from GHOST during evaluation maneuvers.

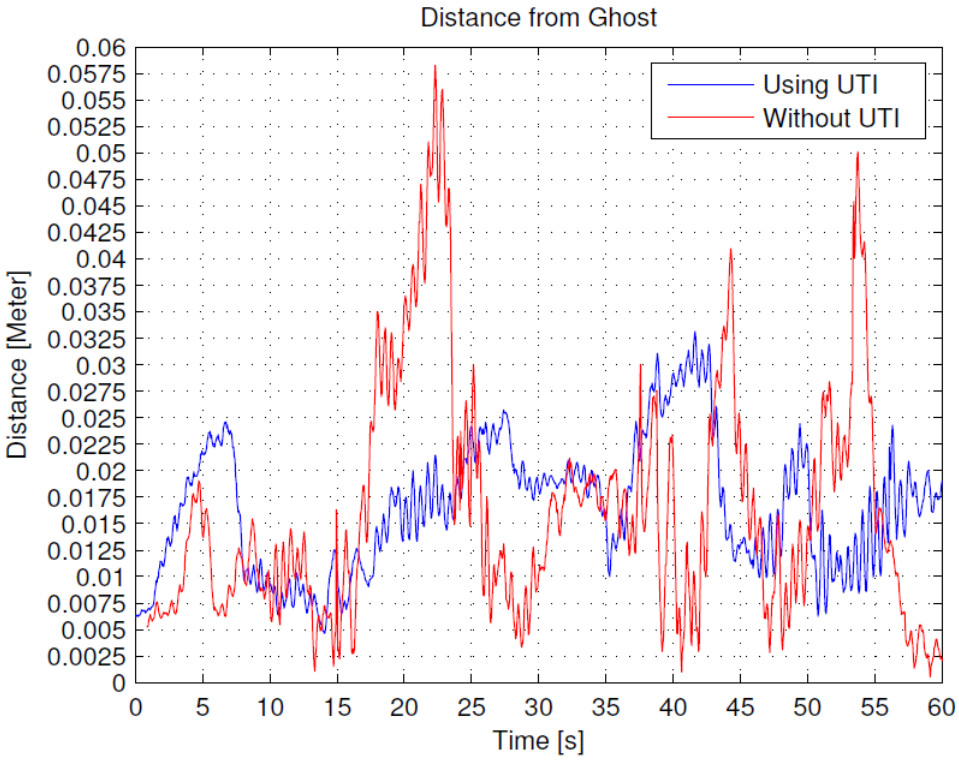


Figure 6.2: Performance of both subjects

6.3 Experiments With Torque mode

6.3.1 Affect of haptic feedback on teleoperating

The first experiment done for the newly refined Torque mode tested the effect of haptics feedback on controlling a humanoid robot. The experiment consisted of using three

subjects with no experience in the platform to produce simple maneuvers. Each subject is subjected to different configuration of the same maneuver. The different configurations were as the following:

- Subject A: With visual and haptic feedback;
- Subject B: Only with visual feedback;
- Subject C: Only with haptic feedback;

The subjects were only susceptible to two basic maneuvers, a hip lateral movement and a knee flexion/extension movement, these maneuvers where chosen due to the inexperience of the subject to be able to reproduce a more complex maneuver within the designated period of time, these maneuvers also provide extensive exploration on the force feedback magnitude and the boundaries of balance.

Each subject had 45 min to practice the maneuver, after that period, the subject had 15 min to perform the best simulation possible. The simulation was considered complete when the user successfully made the simulated model perform three full courses of the assigned maneuver.

All the subjects had the auxiliary mechanism developed for the Torque mode enabled, even the user with haptic feedback disabled experienced the *Joint Configuration Finder* mechanism. Based on personal experience without it, the practice time would had to be greatly increased.

The first maneuver presented to the subjects was the knee flexion/extension movement. To evaluate the performance of each subject, the dispersion of the the robot CoM was recorded. Moreover, all the other parameters are also recorded, and can be referenced in the Appendix A.

Figure 6.3: Configuration of users with visual feedback

Figures 6.6, 6.7 and 6.8 display the CoM distribution of each subject from the selected knee flexion/extension maneuver.

With the analyses of the CoM projection it is possible to verify that Subject A quickly compensated deviations bigger than 0.01 m, producing condense distribution of

Figure 6.4: Configuration of users with no visual feedback

the CoM in the intended path. Although, due to simulation oscillations, the model had translations on the workspace, the operator was skilled enough to regain control over it.

Subject B was able to produce a concentrated distribution pattern, however the path produced is a little deviated from the intended pattern, presenting a slight shift to the side on full extension. This behavior is the effect of the lack of haptic feedback on a complex organism, without it the user was unable to perceive the uneven weight distribution throughout the whole simulation.

Figure 6.4 illustrates the configuration of the setup for the subject C. He had to complete the maneuvers without any visual feedback. However, he was fully aware of when and how the model had fallen, thus being able to correct his approach in the next attempted. Even though he presents a larger distribution area and longer time in the simulation, he was able to produce the intended orientation on the maneuver, even being able to detect potential shifts and slowly correcting them, the frontal projection of the CoM verifies this correction on a large shift to the left.

Table 6.1 presents the data gathered from the selected maneuvers, it presents the maximum deviation from the pretended path, the mean deviation of the whole maneuver and the length of the simulation time. The deviation was calculated from the minimum distance of CoM position, in the given time, to a line segment with the vertices in the coordinates (0,0,0.34) and (0,0,0.4) meters. The length of the simulation expresses the overall affinity to the platform, with a shorter time representing a higher affinity and comfort over the control.

Table 6.1: Data acquired knee flexion/extension experiment

Subject	Max deviation(m)	Mean deviation(m)	Time(s)
A	0.0558	0.0162	70
B	0.0678	0.0186	100
C	0.075	0.0235	120

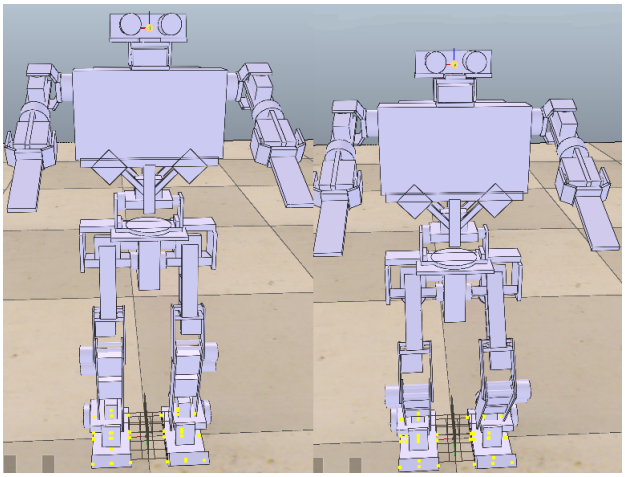


Figure 6.5: Illustration of the knee flexion/extension maneuver.

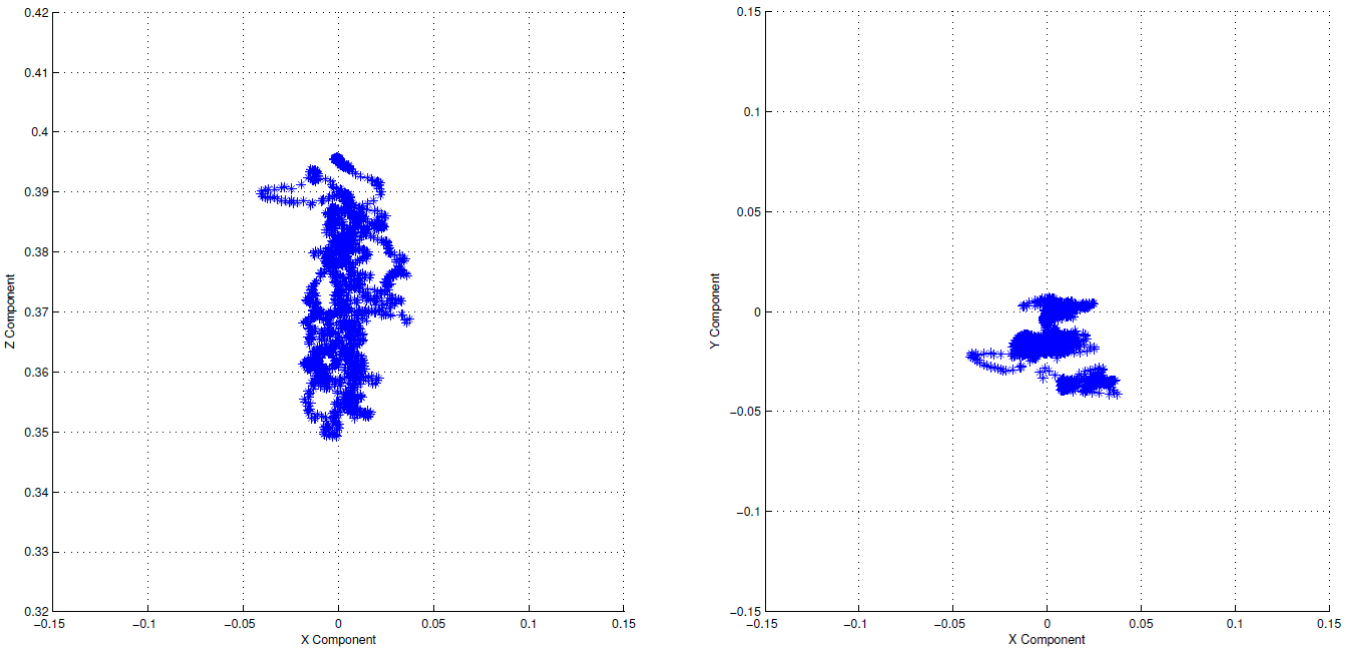


Figure 6.6: COM caption from Subject A performing knee flexion/extension movement on XoZ and XoY planes

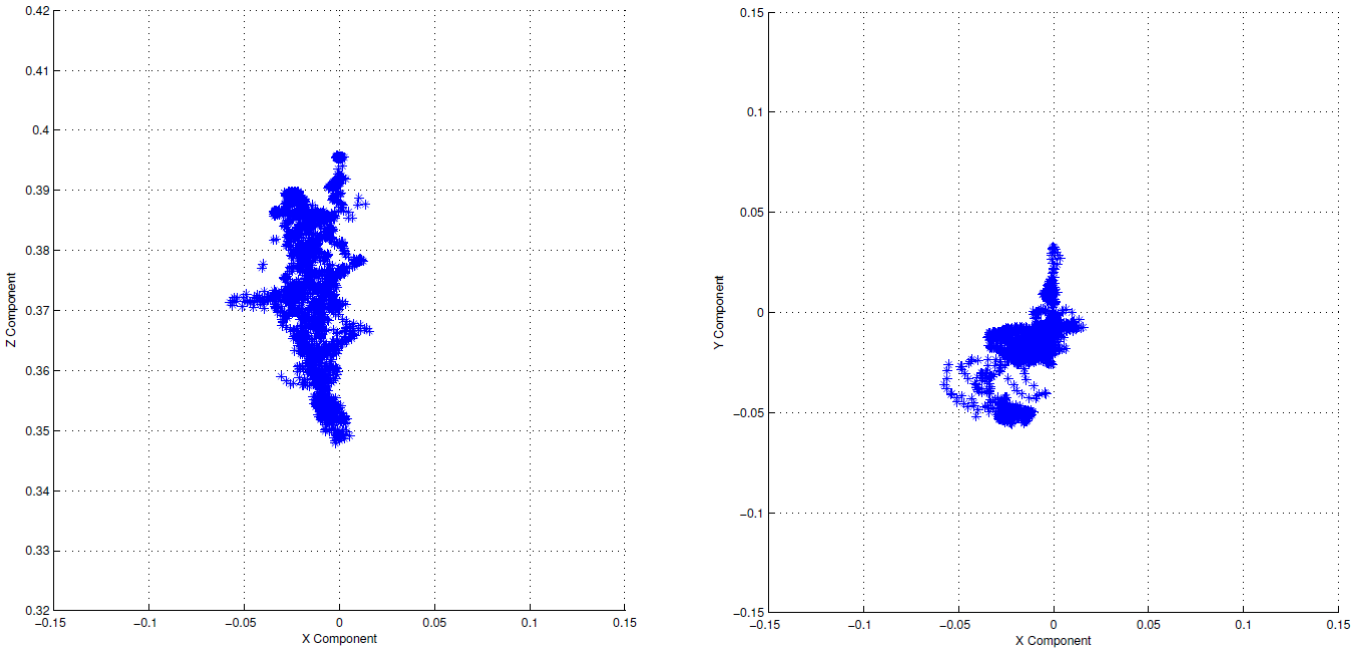


Figure 6.7: COM caption from Subject B performing knee flexion/extension movement on XoZ and XoY planes

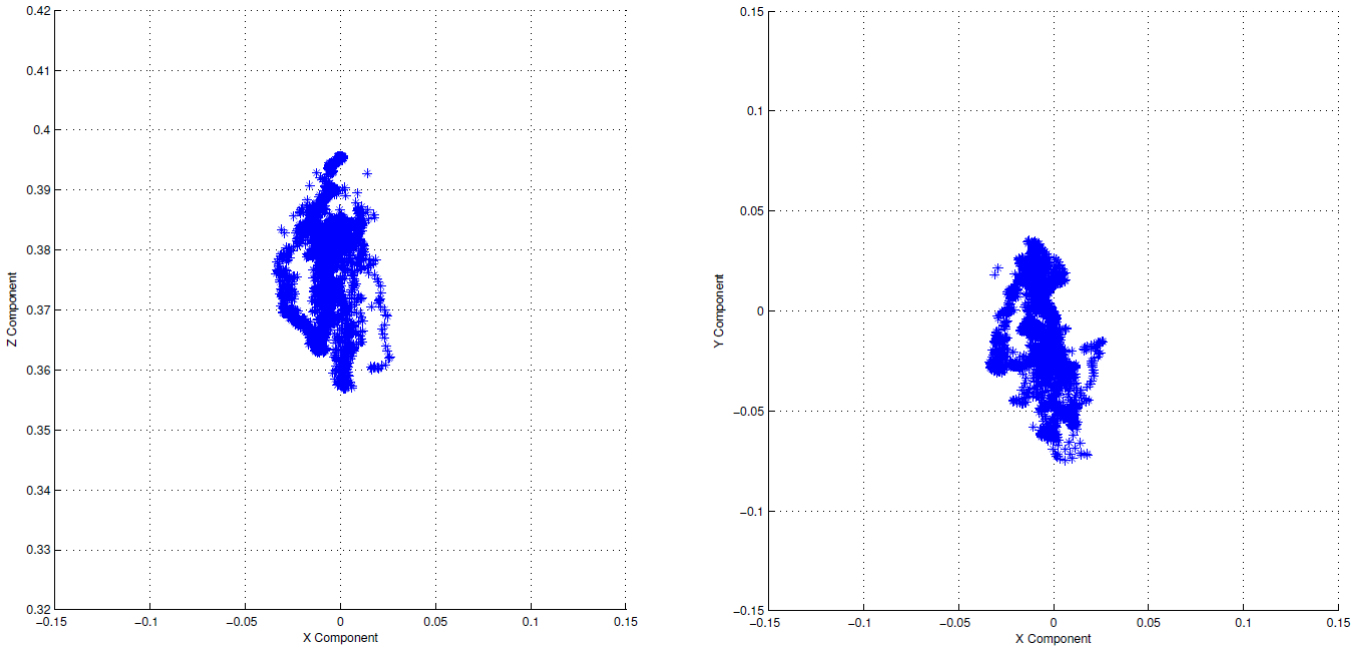


Figure 6.8: COM caption from Subject C performing knee flexion/extension movement on XoZ and XoY planes

Before proceeding to the second maneuver, the subjects had a waiting period of at least 24h. This configuration allows them to demise the little experience obtained during the execution of the first maneuver, thus remaining the most inexperienced as possible.

The second maneuver consists of a hip lateral move. This maneuvers is much more propitious to the production of failed maneuvers, since the users are consistently teleoperating the robot model to the limits of its workspace. To the visual disabled subject it proved to be a far greater challenge than the first maneuver. Similarly to the previous maneuver, the subjects had 45min to practice the maneuver, and after 15min to produce the most reliable maneuver. The settings for each individual subject remain the same as in the previous maneuvers. Figures 6.6, 6.7 and 6.8 display the caption on the CoM of the robot simulated model each operator.

Subject A displays a centralized distribution which represents a good control over the sway over on Y axis, the deviations are easily correct, thus positing the robot on the correct configuration. He also displays a confidante control over its workspace, exceeding the projected limitations with some refine leg configuration.

Subject B presents a non linear path distribution on the XoY plane, however he successfully achieved the extension of the workspace with little shift on the Y axis. The clusters is desviated due to the simulation *unstable jump*.

Unexpectedly the caption from Subject C presents far less sway on the Y axis than the caption of Subject A, thus demonstrating the great contribution on control from only haptic feedback on three dimensional operations. However the effect of the robot's weight, guides unconsciously the subject to a lower configuration, but upon establishing on a certain configuration, the subject has a compact and linear path, exploring the entirely the workspace of the maneuver.

Similar to the the previous maneuver, Table 6.2 details the max deviation and the mean deviation from the pretended path, also it contains the time spent to reproduce the simulation.

Table 6.2: Data acquired from the hip lean experience

Subject	Max deviation(m)	Mean deviation(m)	Time(s)
A	0.0359	0.0083	50
B	0.0593	0.0366	80
C	0.0358	0.0127	140

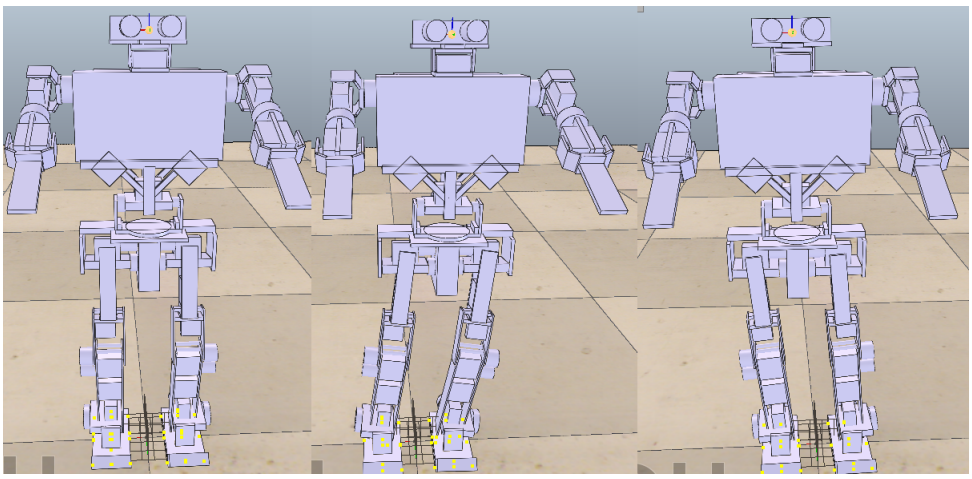


Figure 6.9: Illustration of the hip leaning maneuver.

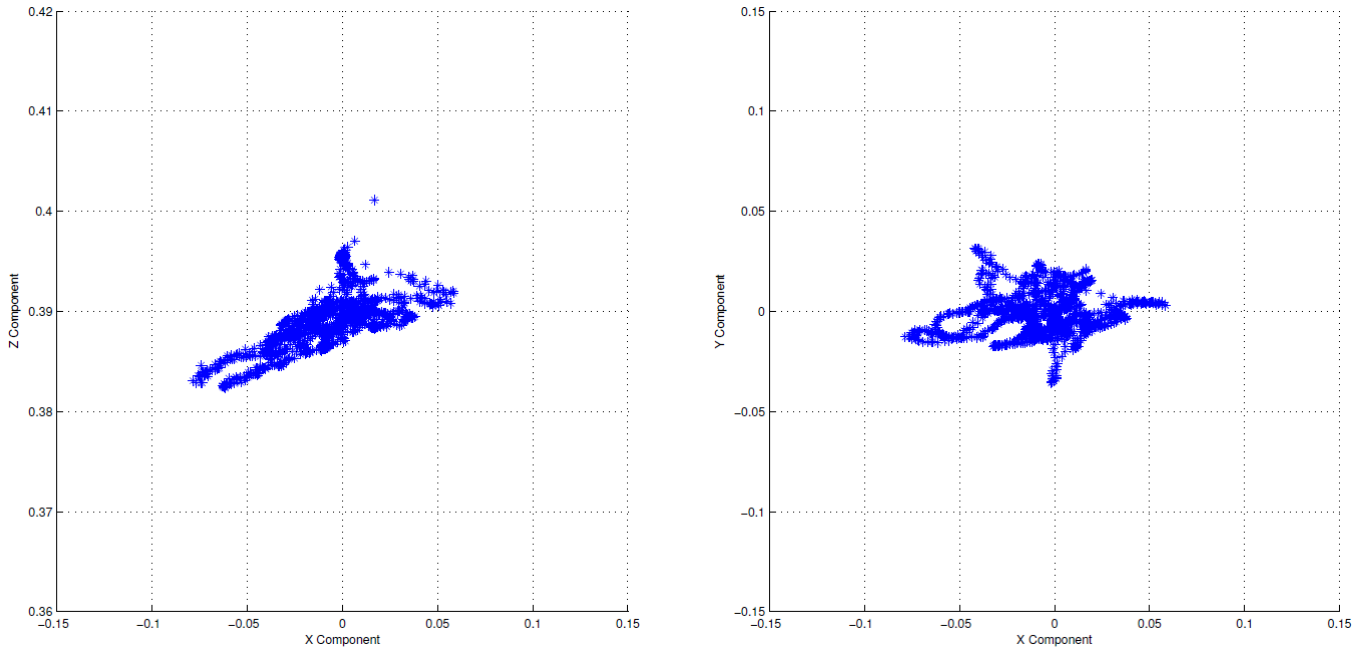


Figure 6.10: COM caption from Subject A performing hip lateral move movement on XoZ and XoY planes

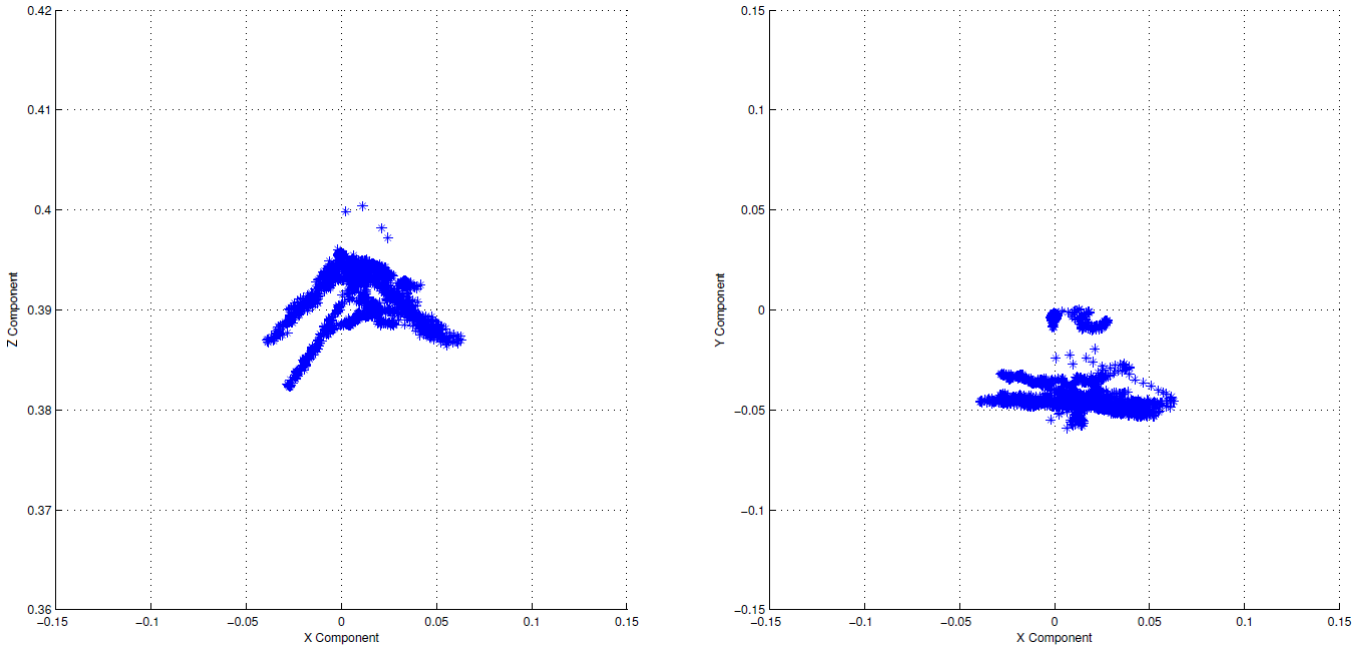


Figure 6.11: COM caption from Subject B performing hip lateral move movement on XoZ and XoY planes

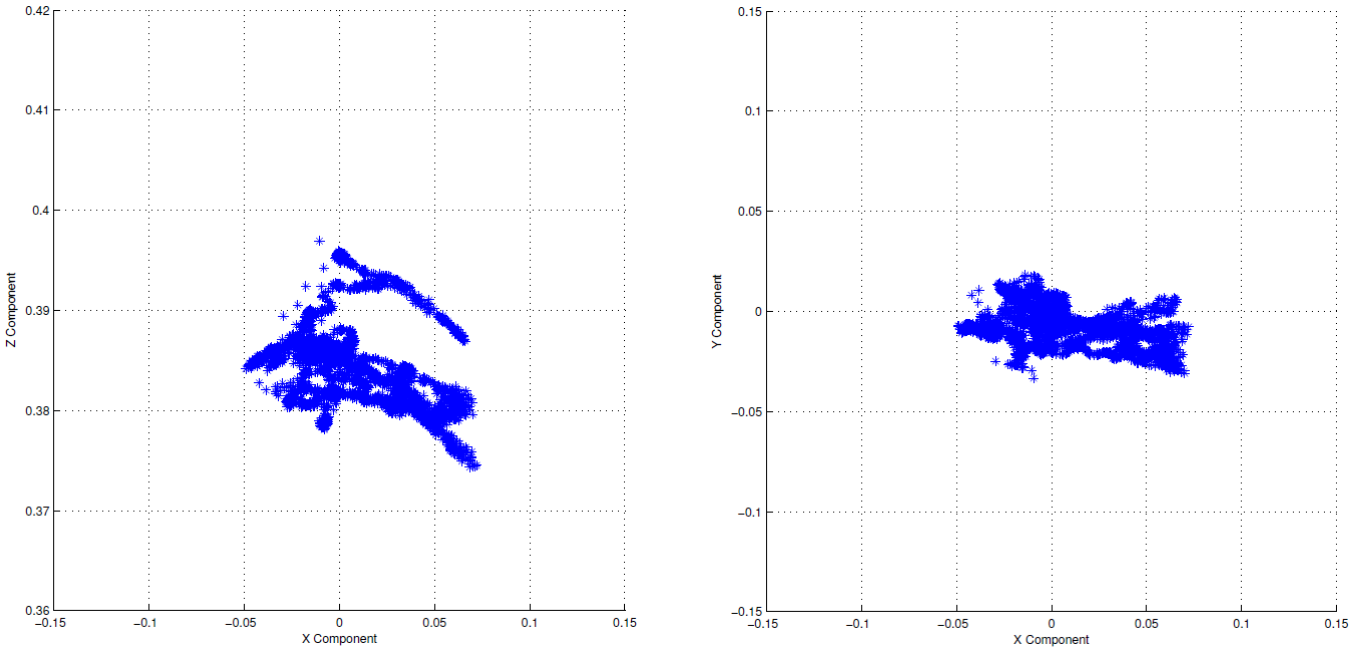


Figure 6.12: COM caption from Subject C performing hip lateral move movement on XoZ and XoY planes

6.3.2 Effectiveness of the haptic guidance tool

To test the effectiveness of the haptic guidance tool presented in Chapter 5, a new subject was subdued to the same experience mentioned on the previous section. He had to replicate the same two maneuvers as the others subjects did, but instead of having 45min to practice and 15min to produce the maneuvers, the new test subject only had to 20min of training, with 10min to reproduce the best maneuver as possible.

Within the training period, Subject D alternated from the haptic guidance tool to the control of the robot as he thought best, refining his kinesthetic memory as much as possible. However, after the training period, the guidance tool was disabled leaving him with full control of the simulation.

Similar to Subject A, Subject D has full haptic and visual feedback, thus providing him with full immersion of the platform. The following Figure 6.13 illustrates the distribution of the COM of the simulated model during the reproduction of the three full course.

As it can be analyzed from the CoM distribution, Subject D performance is displayed concentrated clusters with the proper orientation. Within the short period of training, the operator displays an impressive reaction to the small shifts in the path, even presenting a deeper flexion than the other users. The length of the simulation is surprisingly low, being less than half of the length of the previous users.

Table 6.3 presents the data gathered from Subject D and compares it with the data of Subject A. The comparison between both displays similar patterns, with Subject A presenting a slightly better mean deviation, however on terms of time spend to reproduce the maneuver, Subject D presents an outstanding 20 seconds.

Similar to the other subjects, Subject D had a rest period of at least 24h before

Table 6.3: Data comparison between Subject A and D on the knee flexion/extension experience.

Subject	Max deviation(m)	Mean deviation(m)	Time(s)
A	0.0558	0.0162	70
D	0.0544	0.0195	20

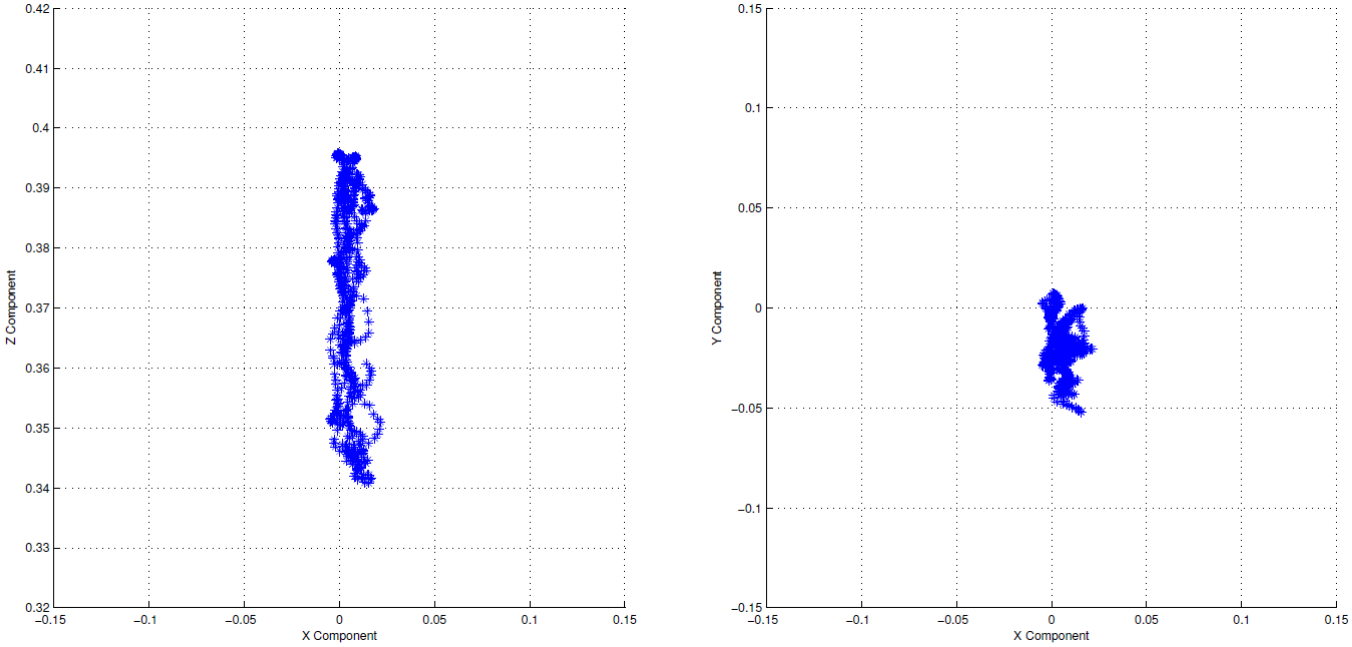


Figure 6.13: COM caption from Subject D performing knee flexion/extension movement on XoZ and XoY planes

returning to the platform to perform the second maneuver test. The conditions were the same as the previous one.

Figure 6.14 illustrates the distribution of the CoM of the simulated model during the reproduction of the full simulation.

The distribution presents some oscillation on the Z plane manly due to the weight response in the force feedback. Apart from that, the distribution shows a controlled response with elevated speed and small shifts on the Y axis, also the subject shows deep knowledge on the robots boundaries, easily controlling it on its edges. The performance of Subject D was similar of the Subject A, presenting a slight improvement on the deviation and half of the length of the simulation.

Further information about the robot state during the simulations is posted on Appendix B.

Table 6.4: Data comparison between Subject A and D on the hip leaning experience.

Subject	Max deviation(m)	Mean deviation(m)	Time(s)
A	0.0359	0.0083	50
D	0.0345	0.0063	30

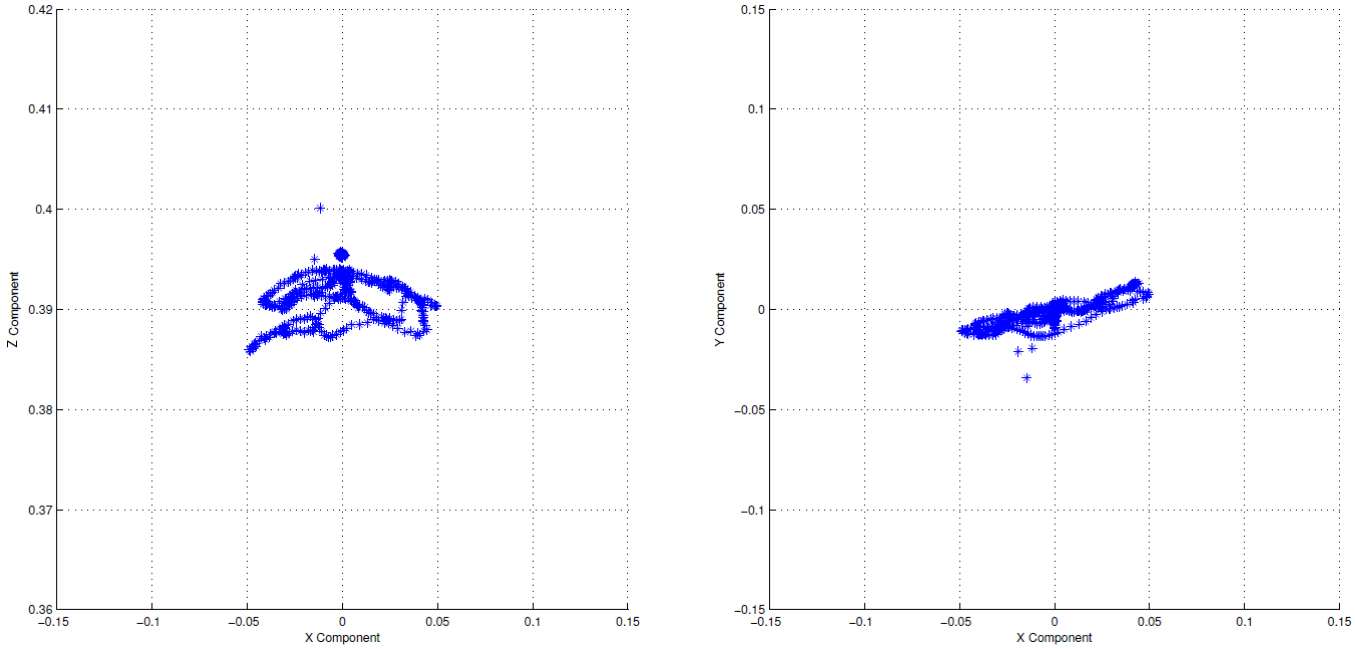


Figure 6.14: COM caption from Subject D performing hip lateral movement on XoZ and XoY planes

6.3.3 Foot Lift Maneuvers

To advance with the library of movements produced by PHUA, a foot lift maneuver was attempted. The reproduction of this maneuver also puts to test the new implementations of the platform, such as the Assisted Torque setting and the force feedback.

The movement of lifting a foot is a greatly important maneuver in the reproduction of the human locomotion and in balance studies.

The fundamentals behind the performance of the maneuver is the placement of the CoP as close as possible on one of the foot and after, to lift the opposite foot to where the CoP is located. The oscillations of the CoP are battled with responsive upper body of the robot, where the only recommended movement provide by the operator would be on the lifted leg.

However it wasn't possible to manifest a successful lift with a balanced posture. It was however, possible to obtain a lift with a controlled impulse of the whole body of the robot. This method allows only a temporary lift of the leg, where the upper body quickly corrects the weight distribution making the leg to land again.

To perform the swing method the user has to follow the step illustrated in the following Figure 6.15.

The first step to the reproduction of the maneuver is to open the legs at level to some degree, a little further than shoulder wide, like it is represented in the previous Figure 6.15, upon achieving a shoulder wide configuration start flexing a knee with some momentum, when the knee reaches closely to haft of its work rotation, extend the acquired flexion. The momentum built during the flexion should propel the robot enough to lift the leg by a small amount, flex after gaining liftoff to produce more height on the foot. While the propulsion is undergoing don't adjust the support leg to correct the unbalanced

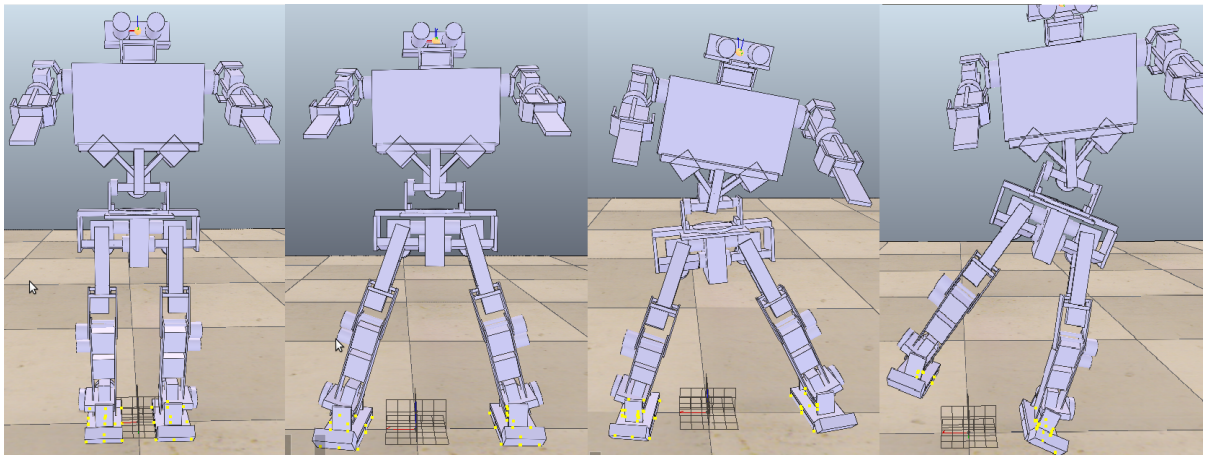


Figure 6.15: Visual schematic of the swing method steps.

configuration, the upper body action is enough to provide counter inertia to return the whole body and the legs to the previous open legged configuration.

This is a difficult repeatable action, but its the first controlled maneuver where the if wished the user can continuously use the momentum produce to lift the other leg. This procedure is illustrated in the Figure 6.16.

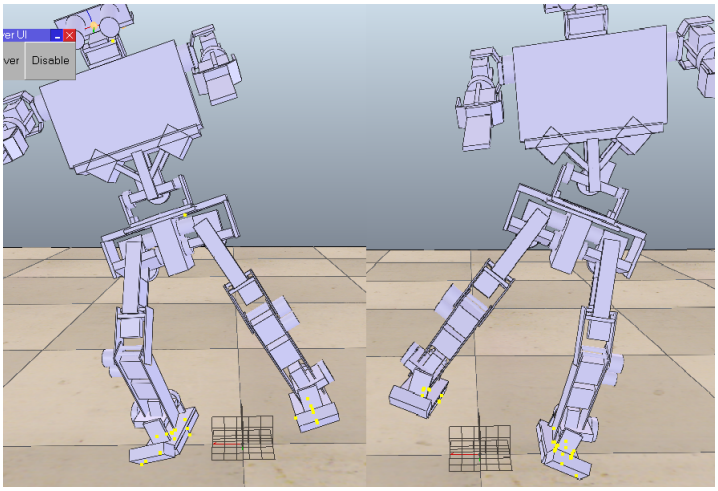


Figure 6.16: Illustration of the a continuous shift of leg support.

To continuously produce the maneuver, a user must not move the support leg when the the opposite leg is still in the air, and only start the propulsion of the second after the first leg has reach the ground, as the user "feels" the momentum generating on the leg as it lands. It is notable to mention that the momentum needed for the second lift is far less than the first, thus necessitates far less impulse. If not properly timed, the model can gain to much momentum and fall, although the shifting in the Y axis is more alarming then the extra momentum.

Chapter 7

Conclusions and Future Work

This chapter details the final conclusions for this dissertation, explaining the different effects of various feedback from the users. Future works are suggested to further improve the platform and to expand the human/haptic interaction.

7.1 Conclusions

The main objectives of this dissertation were to reduce the required adaptation time for user unfamiliar with the humanoid platform, so the continualty of PHUA wouldn't be affected by the continuous change of its users.

This work achieved some contributions for the project, such as the new workstation that allows for a ergonomic approach to the Omni devices in both control configurations, a training interface for each type of control allowing for a reduction of the time required to gain full control over the humanoid platform. These interfaces also produce a renewed approach to the force feedback synthesized in the haptic devices, as well as auxiliary mechanims that help the operator with the control and balance of the robot. Training plans and procedures were also projected for each interface, instructing the user to a faster habilitation in the platform.

The addition of the upper body to the simulated body makes the CoP to exhibit larger oscillations, also giving the model an increase on its inertia. This configuration generates difficulties on the teleoperation of the robot model, which translates in large periods of time necessary to gain some control over the platform. Previously to the completion of this work it would be required a minimum of weeks even months, depending in the affinity of the user to the setup, to gain common control over the kinematic mapping between the DOFs haptic and humanoid. This evaluation is base on my personal experience, it toke close to two week to gain control on the IK control, and nearly a extra full month to be a capable operator in the Torque control.

The implementation of the *User Trainer Interfaces* and the *Auxiliary Tools* to the project made the required adaption period to be reduced considerably, presenting reductions from weeks to a few hours for both the IK and the Torque control configurations.

The insertion of the IK UTI described in Chapter 4, makes a primal example of the personalization capacity that V-REP has to offer. With the addition of several custom UIs, it is possible to alter the settings of the controls over the model without the necessity of new compilations of the nodes, nor the need to stop the simulation, with the operations

being accessible with the simulation online. This allows for a rich in-time personalization to the control over the robot and also the response felt on the user through the force feedback. Each person has its own personalized settings, thus producing an optimized control of the simulation. The creation of GHOST also supports this trainer environment, with the dynamic demonstration of the desirable paths and limitations for each maneuver. With the guidance of GHOST the user can experience all the stimuli presented along the course, thus acquiring the kinesthetic memory representative of the trajectory and force feedback. The linear control over the model in IK mode allows the user to intuitively follow the visual guidance tool without too much effort, thus justifying the advantage of visual guidance mentioned in other works [10]. The evaluation of the IK UTI presented in Chapter 6, shows that the user subjected to no interface interaction had low response time, always overshooting its placement or simply unable to reach the desired location due to variation of the force feedback. In contrast, the user that was subjected to the UTI interface exhibited a great response time always achieving the desired position without too much effort. He displayed great control of the model and a comfortable behavior when responding with the force feedback, even correcting oscillation produced by the simulation instability.

The insertion of the new formulations for the force feedback had a great impact on the control over the simulated model in both the IK mode and the Torque mode configurations. The new formulations implemented in the IK control, described on Chapter 4, made the subjects greatly satisfied with the new behaviors inputted to the force feedback. The quick selection between the formulation has the users quickly finding their fitting, with no preference ever discovered. Each user is unique and thus the approach that he/she makes differs from the others before him/her. Even though the new simulation doesn't answer the needs of all the users, three distinct behaviors were implemented that accommodates most of desirable profiles.

The formulation of force feedback developed for the Torque mode presented in Chapter 5 made the users capable of the detection weight distribution in each leg, as well as inertia patterns. Through the inertia patterns the user was able to decipher potential deviations from the intended course, thus being able to correct the posture and avoid falling down. The great mass of the full body produces enough inertia with small deviations for the hip joints to produce responsive torque that its felt through the haptic device, and thus alerting the operator to a unforeseen movement. This pattern was observed in the evaluation of the haptic contribution in Chapter 6, where Subject C, during the training period, could tell when the robot had fell and how it fell based on the feedback received from the haptic device, thus adjusting to it in the next simulation.

Subjects A & B, while teleoperating the simulated robot during their maneuvers of evaluation describe in Chapter 6, confessed that most of the time, the visual feedback used to guide their control was the auxiliary window displaying the feet and the CoP position, only converting to the real visual image of the robot to evaluate the configuration of the leg. This information enforces the idea that the CoP is an essential piece of information for the users, informing them of the weight distribution of the robot that could predict the robot state.

The implementation of the UTI in the torque configuration allows users to further shorten the period of training, already shortened by the implementation of the *Auxiliary Tools* presented in Chapter 5. With the implementation of the HIW and the Auxiliary Tools, the time to gain control of the platform under the Torque mode is lower to a

few hours. This decrease of the required time is due to the more intuitive approach handle by the HIW and the stability produce by the Assisted configuration and the Joint Configuration Finder tools detailed in Chapter 5.

Those few hours can be even more reduce, when using the haptic guidance tool. As its proven in Chapter 6, a user subjected to this type of training, gains full kinesthetic memory of the assigned maneuver, gaining enough control to reasonably perform the task withing a few minutes. The human ability to remember limb position and velocity is explored with this tool and its relevance is presented with the success of the training of the user submit to it. The caption from the performance of the Subject D in Chapter 6, illustrates that same fundamental conclusion. Just under 30 minutes the subject is able to fully grasp the motion and the forces behind the assigned maneuver.

When conjugating the use of the Support Tools with and experienced user, it was possible to produce a lift of a foot. Even if not exposing a balanced position, the executed maneuver of Chapter 6 was only possible with the new implemented applications. The maneuver explores weight acceptance dynamics that can be later used in learning algorithms. A balanced version can be achieved with a refinement in the assisted configuration, but the control over the position of the target dummy is still too raw. With a more specific solution between the lower and upper body parts, it will be possible to balance the robot in one foot. The main issue with the completion of the maneuver of the unplaced upper body mass, the high density of the torso, if too leaned, can sweep the support leg.

7.2 Future Work

To further produce more complex maneuver with PHUA a refined conjugation between torso and legs configurations is necessary. The Assisted Torque Configuration is the base foundation towards a replication of the human locomotion, although it needs further refinement. As it stands now the configuration balances the torso to a opposite rotation as the waist display in comparison with the foot, but there are maneuvers where further rotation towards the waist if more efficient than a counter rotation. Other maneuvers incorporated in the human locomotion required a slight lean to the front or to the back. Those configurations are unavailable with the calculations for the Y component of the IK element of the robot's torso disabled due to the oscillations caused by its implementation, but if a different IK element would be define to control that element with its own dynamic proprietors could further help the simulation balance as the effectiveness of some maneuvers.

It is also notabel to mention that the joints in the torso IK elements have no PID study done, they still possess the basic PID configuration of V-REP, presenting only a proporcional value of 0.1. A study to find the ideal PID configuration for the torso's joints may improve its response type and filtered some unwanted configurations and oscillations.

The archived maneuvers are a tool to provide haptic guidance to the user, but they can easily be used to duplicate maneuvers, this features allows for a easy method to update or create data sets for the learning algorithms. New configuration or information latter implemented can be easely study with the safe reproduction of the archived maneuvers, unlike programmed maneuvers these maneuvers, even if produced via script, contain the

behavior of the a human operator.

Another feature of the haptic guidance is the simplicity of its inputs, which allows its usage for platforms with a similar structure. When developed, it was intended for this mechanism to interact with as many different models as possible, thus creating a large amount of maneuver saved within its archive. One of the most interesting interactions would be with a model retrieve from a motion caption session with a real human. The weight distribution of the motion caption would rather be different from the weight distribution of PHUA, but with a few adjustments in the tool, it is possible to enable the control over the simulation with the Omni device but still have the guiding force with the device, thus moving the joystick with the same configuration the imported model. This could allow the operator to correct any differences between the PHUA and the outside model while still performing the same maneuver.

This dissertation shown that the kinesthetic memory of a human is a powerful tool to reproduce movements and maneuvers. The same interaction method developed in this work could be applied to a physic application, similar to the Ghost arm band [23], a sleeve for the leg can be developed. With the analyses of the position of the hip, knee and ankle joints, utilizing the same method as the haptic guidance tool, the sleeve could provoke minor force in specific locations to instructed the user of deviations in the intended joint placement. The finality of that project would point to rehabilitation purposes where the user would had to undergo with motion caption previous to any disability to archive the natural movements of his joints for later use.

Other medical approach for this work, would be to use the synthesize of force feedback developed for the Torque mode and apply it in exoskeletons. Most of the exoskeletons have problems to express the conditions present in their own parts. Force feedback on this machines are the relevant importance, since most of their users have lack of perception of their lower limbs, thus a force feedback able to perceive existing or lack of obstruction is searched for.

References

[1] J. Barros, "Cooperative haptics for humanoid robot teleoperation", Master's thesis, University of Aveiro, 2015.

[2] P. Cruz, "Haptic interface data acquisition system" Master's thesis, University of Aveiro, 2012.

[3] E. Estrelinha, "Tele-operation of a humanoid robot using haptics and load sensors" Master's thesis, University of Aveiro, 2013.

[4] P. Cruz, V. Santos. and F. Silva. "Tele-kinesthetic teaching of a humanoid robot with haptic data acquisition" in IROS'2012 Workshop on Learning and Interaction in Haptic Robots, 2012.

[5] K. Salisbury, F. Conti and F. Barbagli, "Haptic rendering: introductory concepts" IEEE Computer Graphics and Applications, vol. 24, pp. 24-32, Mar. 2004.

[6] M. A. Srinivasan and C. Basdogan, "Haptics in virtual environments: Taxonomy, research status, and challenges" Computers and Graphics (Pergamon), vol. 21, no. 4, pp. 393-404, 1997.

[7] V. Hayward, O. R. Astley, M. Cruz-Hernandez, D. Grant. and G. Robles-De-La-Torre, "Haptic interfaces and devices" Sensor Review, vol. 24, pp. 16-29, 2004.

[8] N. F. Vaibhav, V. S. Mohit, and A. A. Anilesh. "Applications of haptics technology in advance robotics" in INTERACT-2010, pp. 273-277, IEEE, Dec. 2010.

[9] O. J. Rösch, K. Schilling, and H. Roth, "Haptic interfaces for the remote control of mobile robots" Control Engineering Practice, vol. 10, pp. 1309-1313, Nov. 2002.

[10] D. Feygin, M. Keehner, and F. Tendick. "Haptic guidance: experimental evaluation of a haptic training method for a perceptual motor skill" in Proceedings 10th Symposium on Haptic Interfaces for Virtual Envimnment and Teleoperator Systes. HAPTICS 2002, pp. 40-47, IEEE Comput. Soc, 2002.

[11] F. J. Clark and K. W. Horch, "Kinesthesia" Handbook of Perception and Human Performance, vol. 1, pp. 13-62, 1981.

[12] A. M. Howard, "Transfer of skills between human operators through haptic training with robot coordination" in 2010 IEEE International Conference on Robotics and Automation, pp. 229-235, IEEE, May 2010.

[13] J. Barros, F. Serra, V. Santos, and F. Silva, "Tele-kinesthetic teaching of motion skills to humanoid robots through haptic feedback" in IEEE-RAS Humanoids2014 Workshop on Policy Representations for Humanoid Robots, 2014.

[14] M. Whittle, Gait Analysis: An Introduction. Butterworth-Heinemann, 2007.

[15] CoppeliaRobotics, V-REP User Manual. [Online]. Available: http://www.coppeliarobotics.com/helpFiles/index.html, 2015.

[16] S. P. M. M. S. S. M. Kumar, P, "Gravity compensation for phantom omni haptic interface" in First Joint International Conference on Multibody System Dynamics, IMSD-2010, 2010. I

[17] SensAble, PHANToM Omni User's Guide. USA: SensAble Technologies, 2008.

[18] SensAble, OpenHaptics Toolkit Programmer's Guide. USA: SensAble Technologies, 2008.

[19] SensAble, OpenHaptics Toolkit API Reference Manual. USA: SensAble Technologies, 2008.

[20] A. J. Silva, O. A. D. Ramirez, V. P. Vega, and J. P. O. Oliver, "PHANTOM OMNI Haptic Device: Kinematic and Manipulability" in 2009 Electronics, Robotics and Automotive Mechamcs Conference (CERMA), pp. 193-198, IEEE, Sept. 2009.

[21] A. Martinez and E. Fernández, "Learning ROS for Robotics Programming", Packt Publishing, Sept. 2013.

[22] ROS.0rg, Documentation - ROS Wiki. [Online] Available: http://wiki.ros.org/, 2015.

[23] J. Cheah, B. Copping, S. Grover and I. Rasouli, "Vibrating armband helps athletes make the right moves", Imperial College of London, 2012.

[24] J. Lee and S. Choi, "Effects of Haptic Guidance and Disturbance on Motor Learning:Potential Advantage of Haptic Disturbance", Haptics Symposium, 2010 IEEE

[25] N. Kusumoto ,T. Sohmura ,S. Yamada ,K. Wakabayashi ,T. Nakamura and H. Yatani "Application of virtual reality force feedback haptic device for oral implant surgery.", Clin Oral Implants Res. 2006.

[26] MikroElektronika, Digital Filter Design, [Online] Available: http://www.mikroe.com, 2015

[27] Z. Jiang, Z. Gao, X. Chen and W. Sun ,"Remote Haptic Collaboration for Virtual Training of Lumbar Puncture", Journal of Computers, VOL. 8, NO. 12, 2013.

[28] S. Saga, N. Kawakami and S. Tachi, "Haptic Teaching using Opposite Force Presentation", World Haptics Conference Pisa, Italy, 2005.

[29] Minerva Brooks Memorial Library, Inc., "Haptic Guidance ", 2012.

[30] Geomagic Products [Online], Available: http://www.geomagic.com , 2015.

[31] S.H.L. McAmis, and K.B. Reed, "Simultaneous Perception of Forces and Motions Using Bimanual Interactions", IEEE TRANSACTIONS ON HAPTICS, vol. 5, no. 3, pp. 220-230, 2012.

[32] J. Bluteau ,S. Coquillart ,Y. Payan and E. Gentaz, "Haptic Guidance Improves the Visuo-Manual Tracking of Trajectories.", PLoS ONE 3, 2008.

[33] J. Solis, C.A. Avizzano and M. Bergamasco, "Teaching to write Japanese characters using a haptic interface", Haptic Interfaces for Virtual Environment and Teleoperator Systems, pp. 255-262, 2002.

[34] K. Henmi and T. Yoshikawa, "Virtual lesson and its application to virtual calligraphy system", Robotics and Automation, vol.2, pp. 1275-1280, 1998.

[35] P.T.C.P.B. Gillespie ,S. O'Modhrain and D. Zaretsky, "The virtual teacher" International Mechanical Engineering Conference and Exposition, pp. 171?174, 1998.

[36] R. Palluel-Germain, F. Bara, A. Hillairet de Boisferon, B. Hennion, P. Gouagout and E. Gentaz,"A Visuo-Haptic Device - Telemaque - Increases Kindergarten Children's Handwriting Acquisition", IEEE World Haptics 2007,pp. 72-77, 2007.

[37] G. Srimathveeravalli and K. Thenkurussi, "Motor skill training assistance using haptic attributes.", Haptic Interfaces for Virtual Environment and Teleoperator Systems, pp. 452-457, 2005.

[38] D. Morris, H. Tan, F. Barbagli, T. Chang and K. Salisbury, "Haptic feedback enhances force skill learning.", EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, pp. 21-26, 2007.

[39] S. Calinon, P. Evrard, E. Gribovskaya, A. Billard, and A. Kheddar, "Learning collaborative manipulation tasks by demonstration using a haptic interface." in 2009 International Conference on Advanced Robotics, ICAR 2009, 2009.

[40] P. Kormushev, D. N. Nenchev, S. Calinon, and D. G. Caldwell, "Upper-body kinesthetic teaching of a free-standing humanoid robot." in 2011 IEEE International Conference on Robotics and Automation, pp. 3970-3975, IEEE, May 2011.

[41] A. Completo and F. Fonseca, "Fundamentos de Biomecânica", Gráficas Anduriña, 2011.

Appendices

Appendix A

Experiments without haptic guidance Additional Data

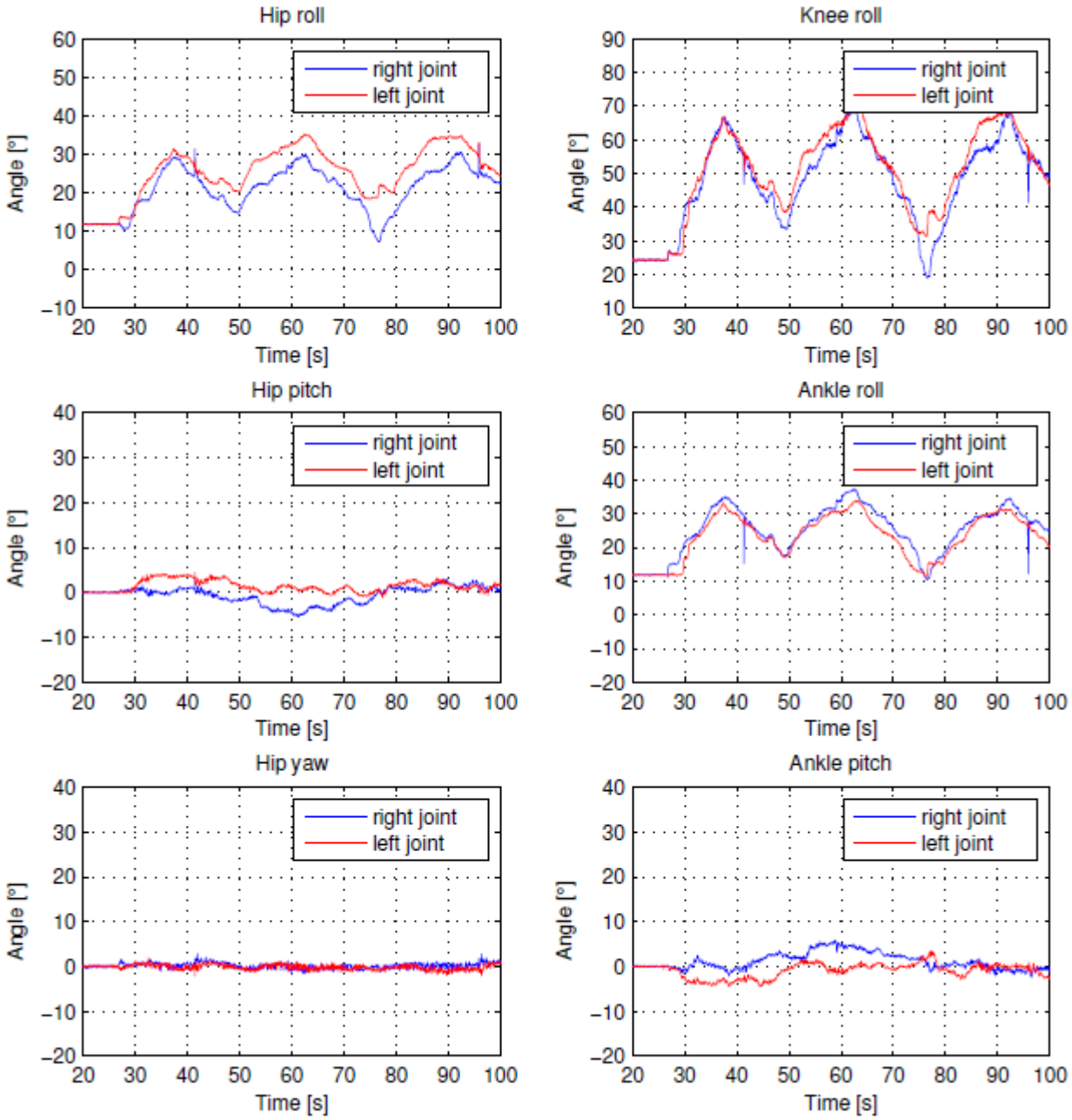


Figure A.1: Robot's joint state evolution during Subject A knee flexion/extension maneuver.

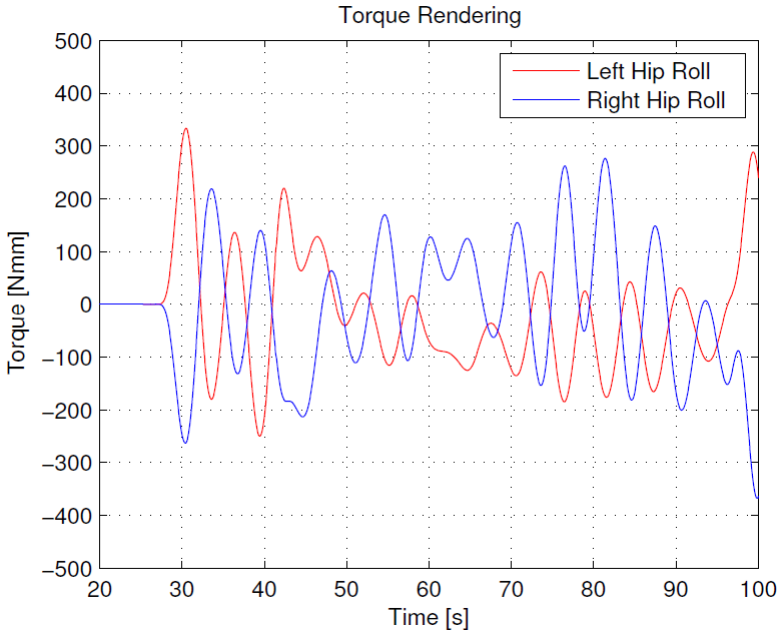


Figure A.2: Torque rendered in the device's second Joint during Subject A knee flexion/extension maneuver.

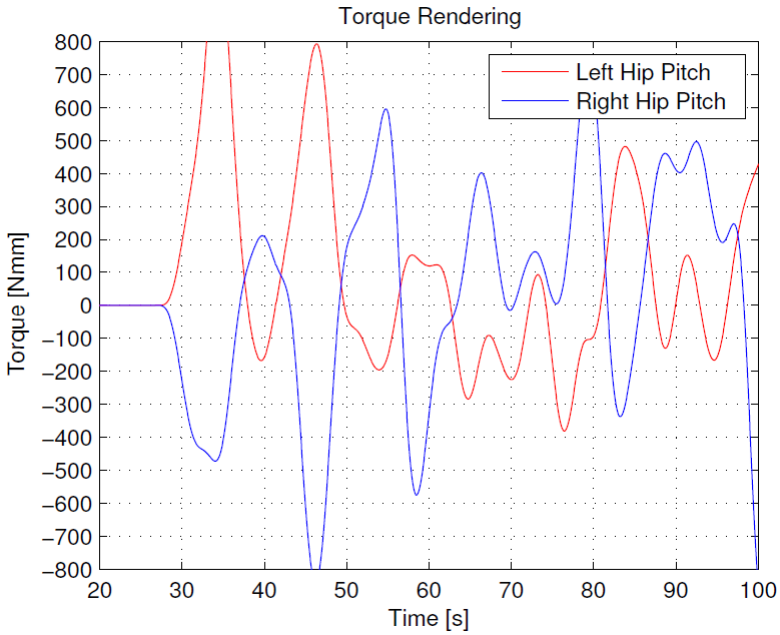


Figure A.3: Torque rendered in the device's first Joint during Subject A knee flexion/extension maneuver.

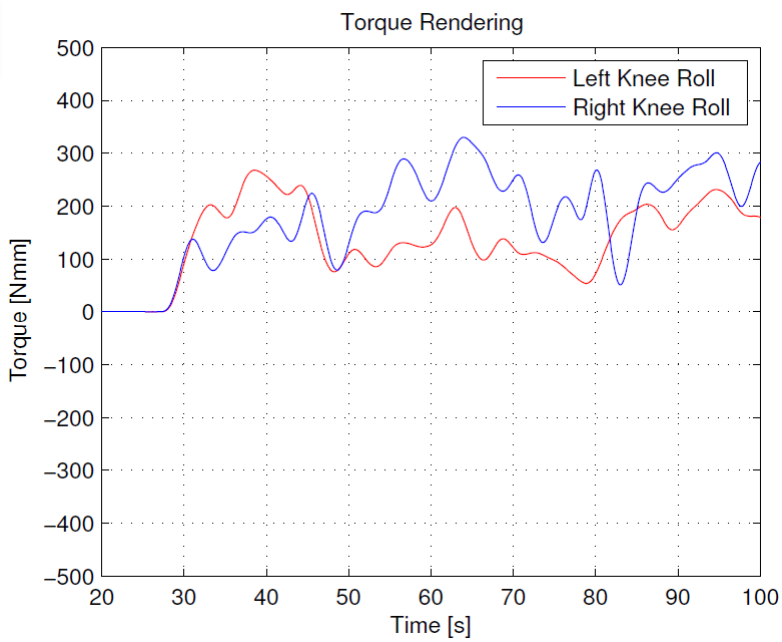


Figure A.4: Torque rendered in the device's third Joint during Subject A knee flexion/extension maneuver.

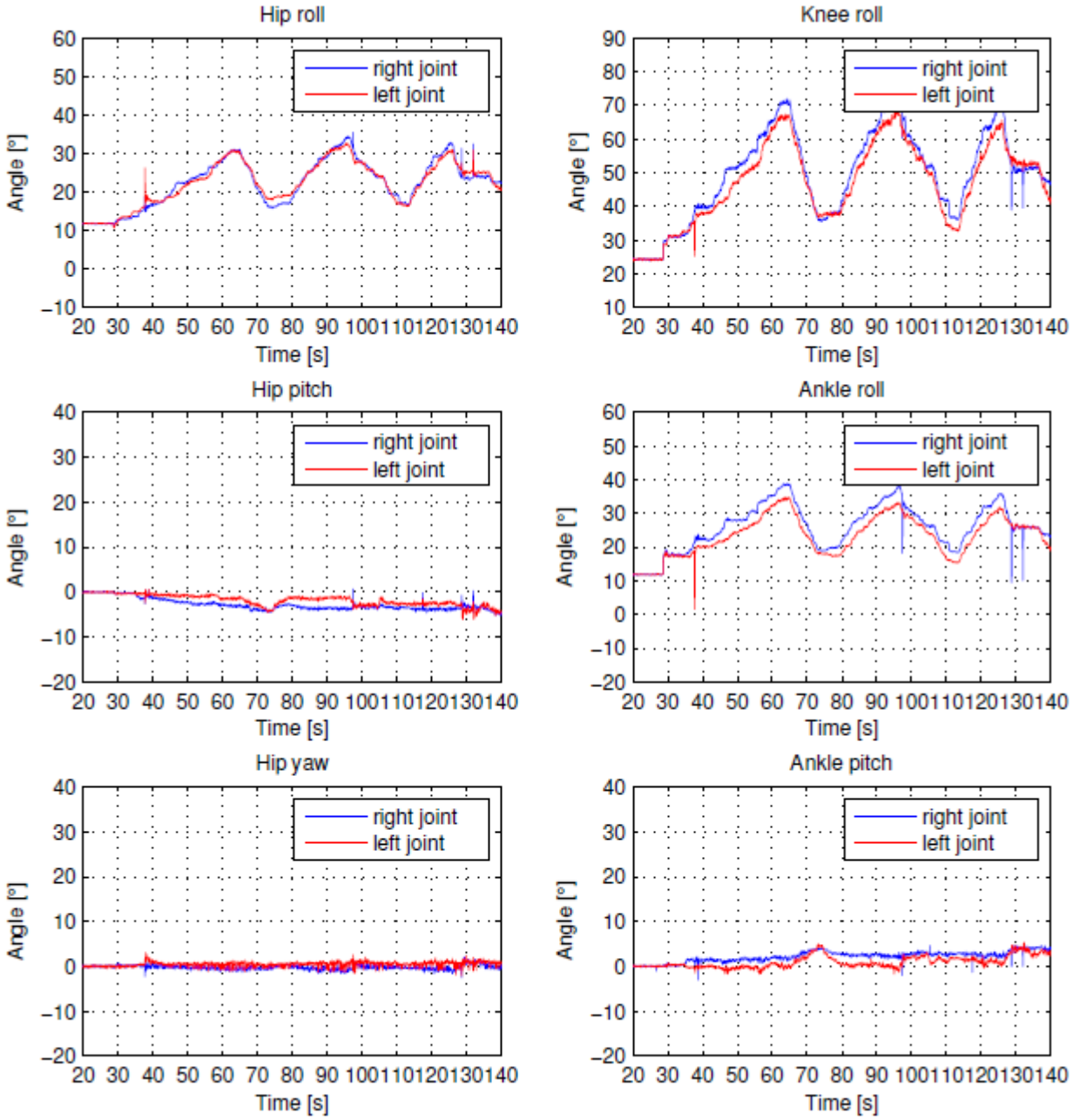


Figure A.5: Robot's joint state evolution during Subject B knee flexion/extension maneuver.

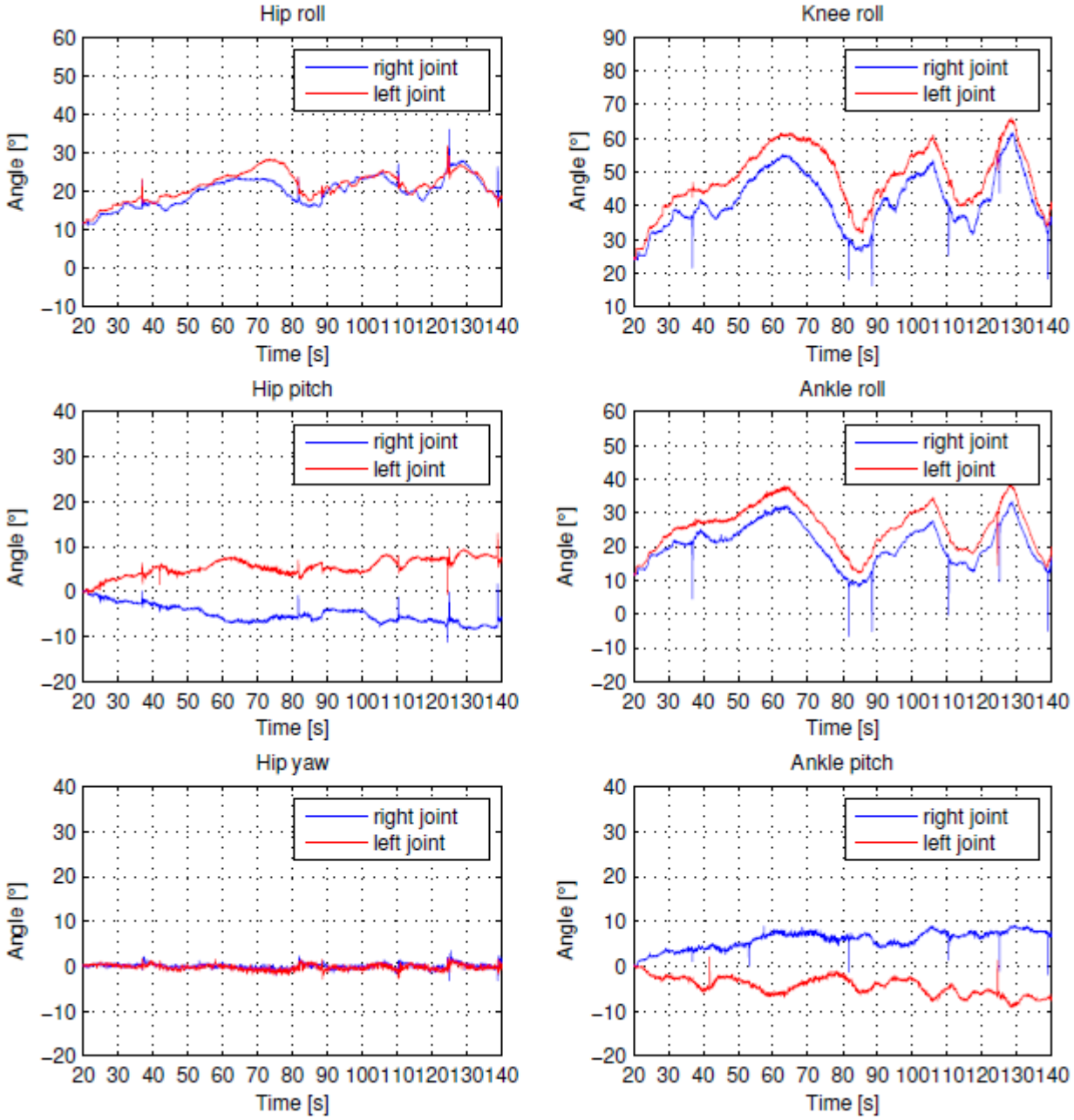


Figure A.6: Robot's joint state evolution during Subject C knee flexion/extension maneuver.

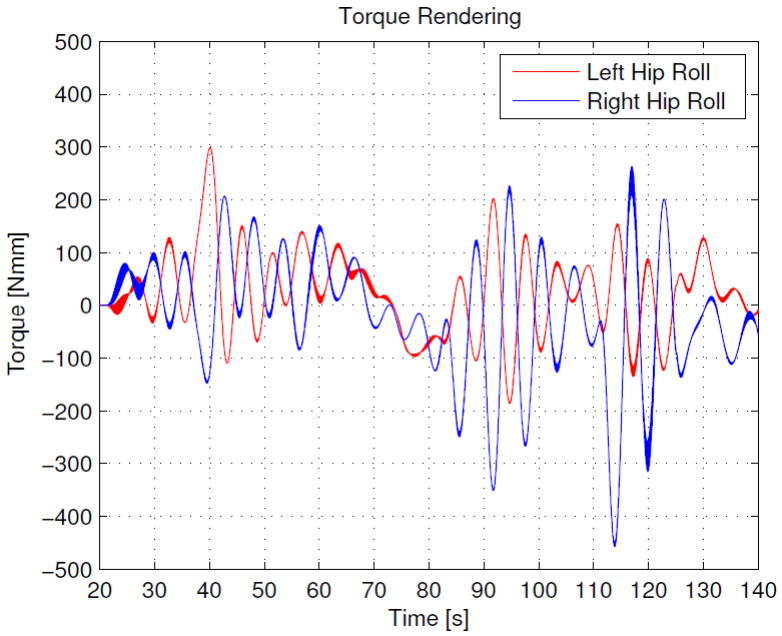


Figure A.7: Torque rendered in the device's second Joint during Subject C knee flexion/extension maneuver.

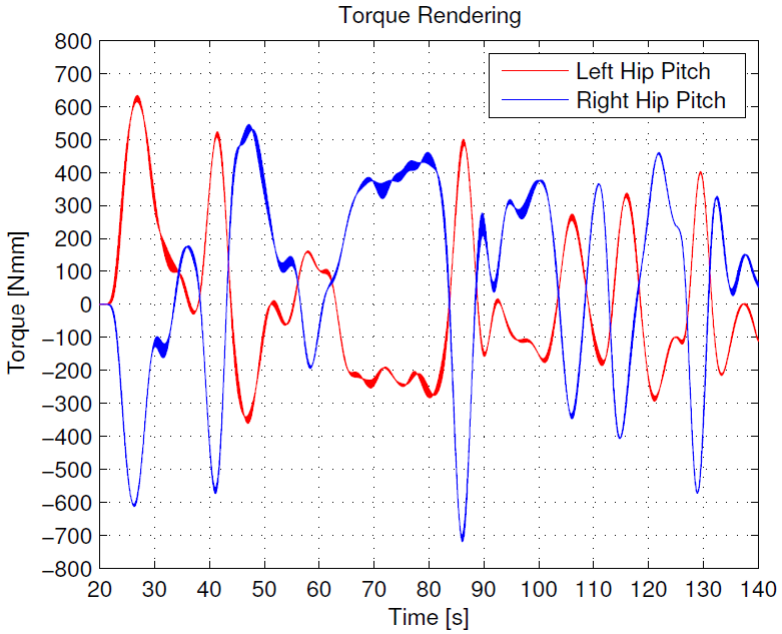


Figure A.8: Torque rendered in the device's first Joint during Subject C knee flexion/extension maneuver.

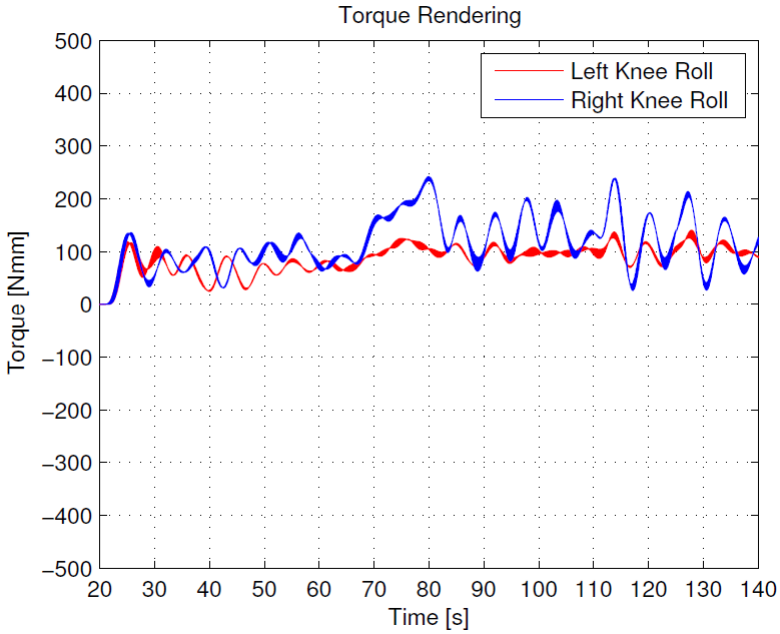


Figure A.9: Torque rendered in the device's third Joint during Subject C knee flexion/extension maneuver.

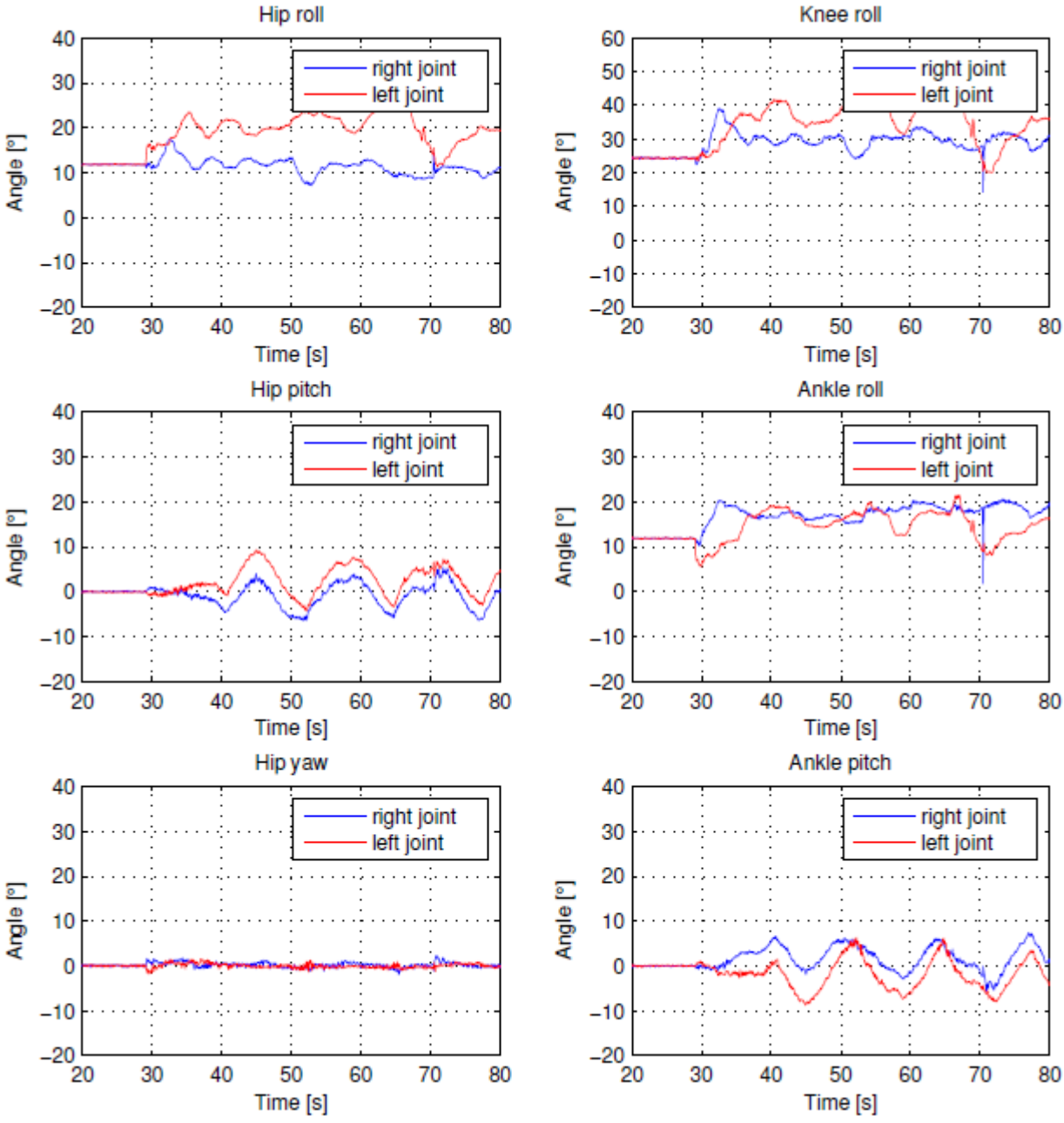


Figure A.10: Robot's joint state evolution during Subject A hip lateral maneuver.

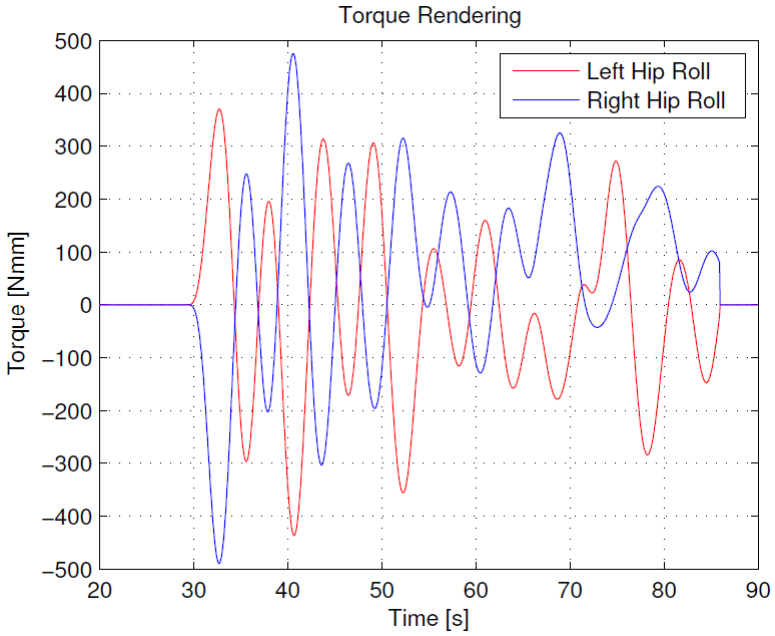


Figure A.11: Torque rendered in the device's second Joint during Subject A hip lateral maneuver.

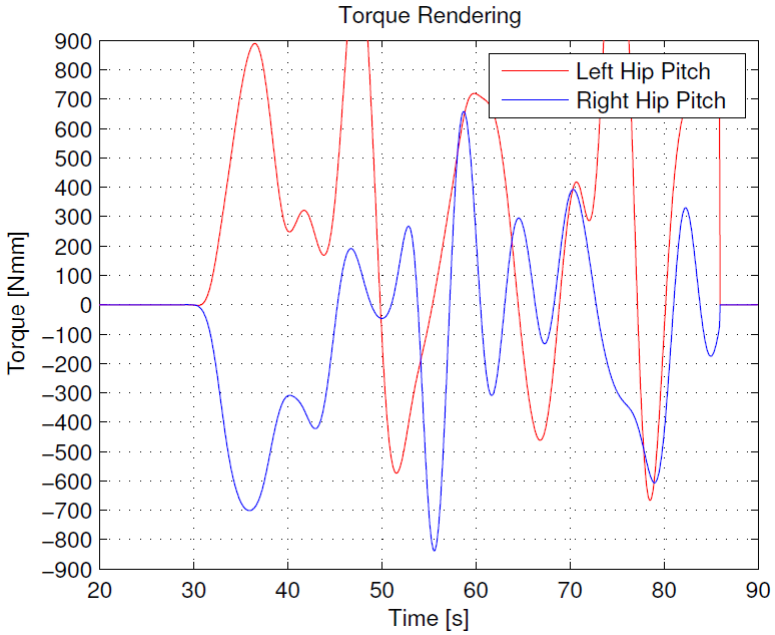


Figure A.12: Torque rendered in the device's first Joint during Subject A hip lateral maneuver.

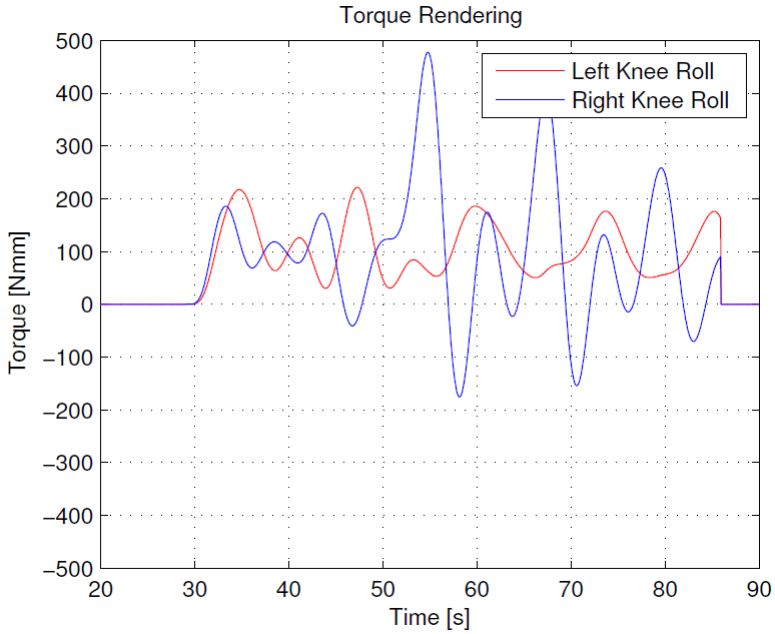


Figure A.13: Torque rendered in the device's third Joint during Subject A hip lateral maneuver.

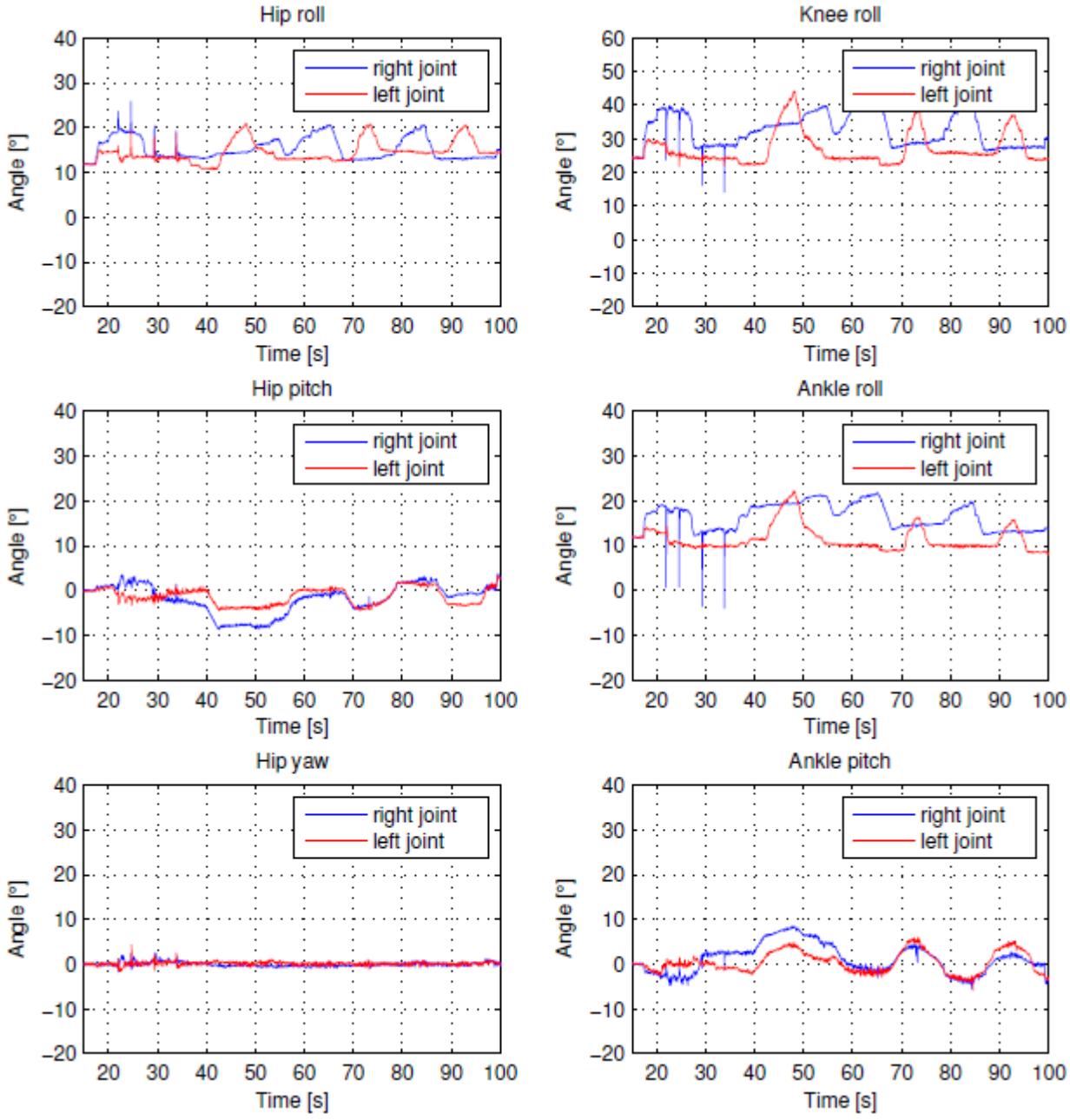


Figure A.14: Robot's joint state evolution during Subject B hip lateral maneuver.

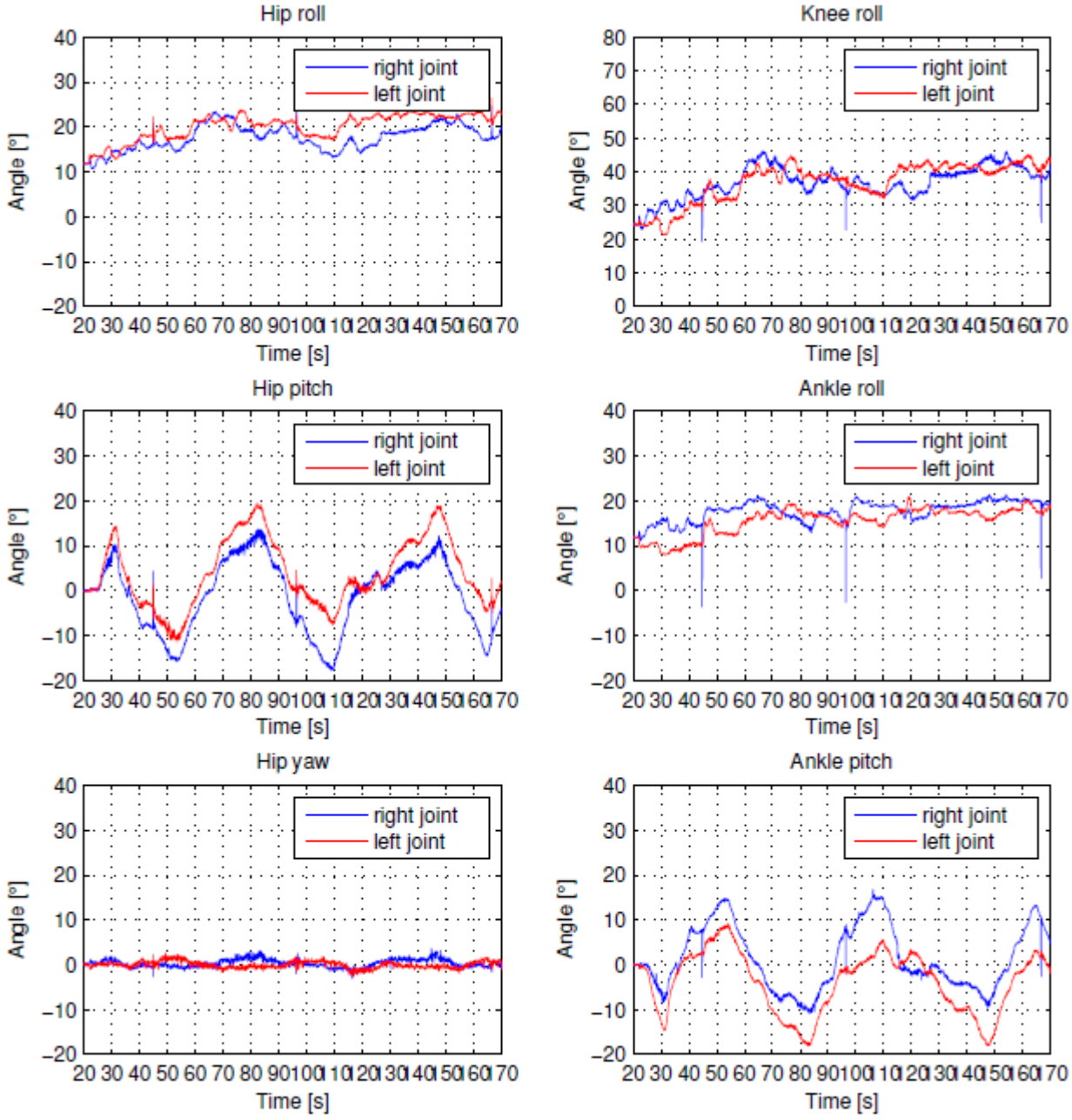


Figure A.15: Robot's joint state evolution during Subject C hip lateral maneuver.

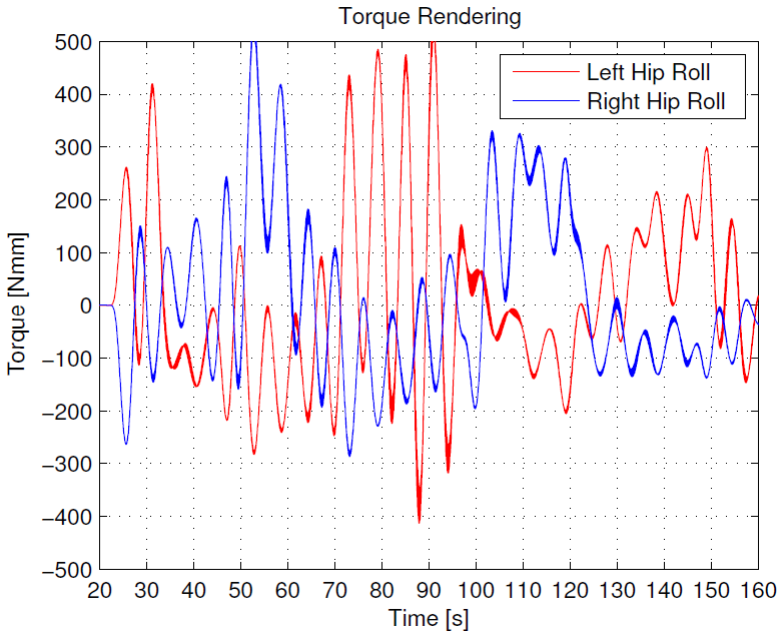


Figure A.16: Torque rendered in the device's second Joint during Subject C hip lateral maneuver.

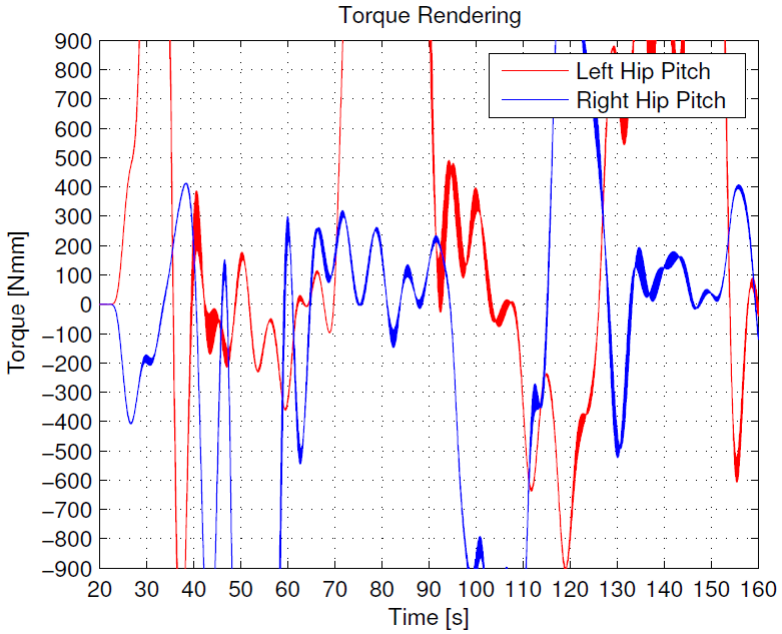


Figure A.17: Torque rendered in the device's first Joint during Subject C hip lateral maneuver.

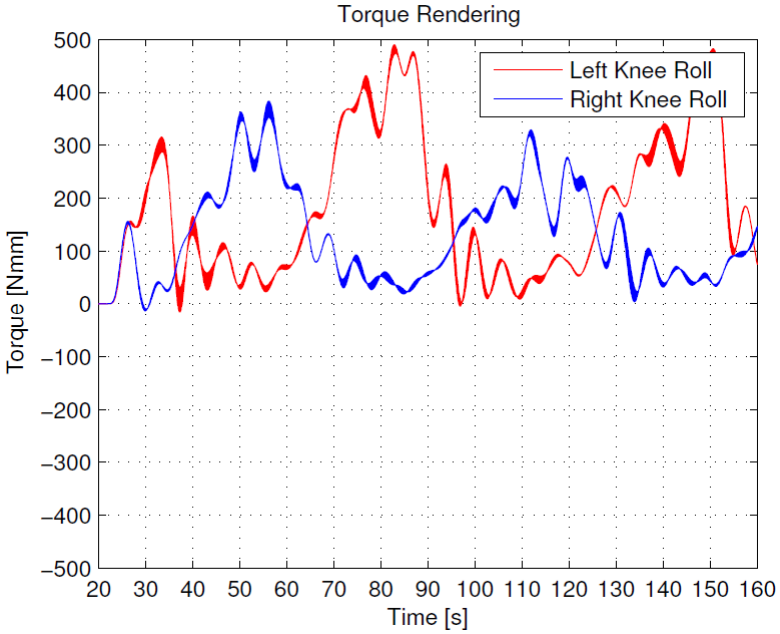


Figure A.18: Torque rendered in the device's third Joint during Subject C hip lateral maneuver.

Appendix B

Experiments haptic guidance Additional Data

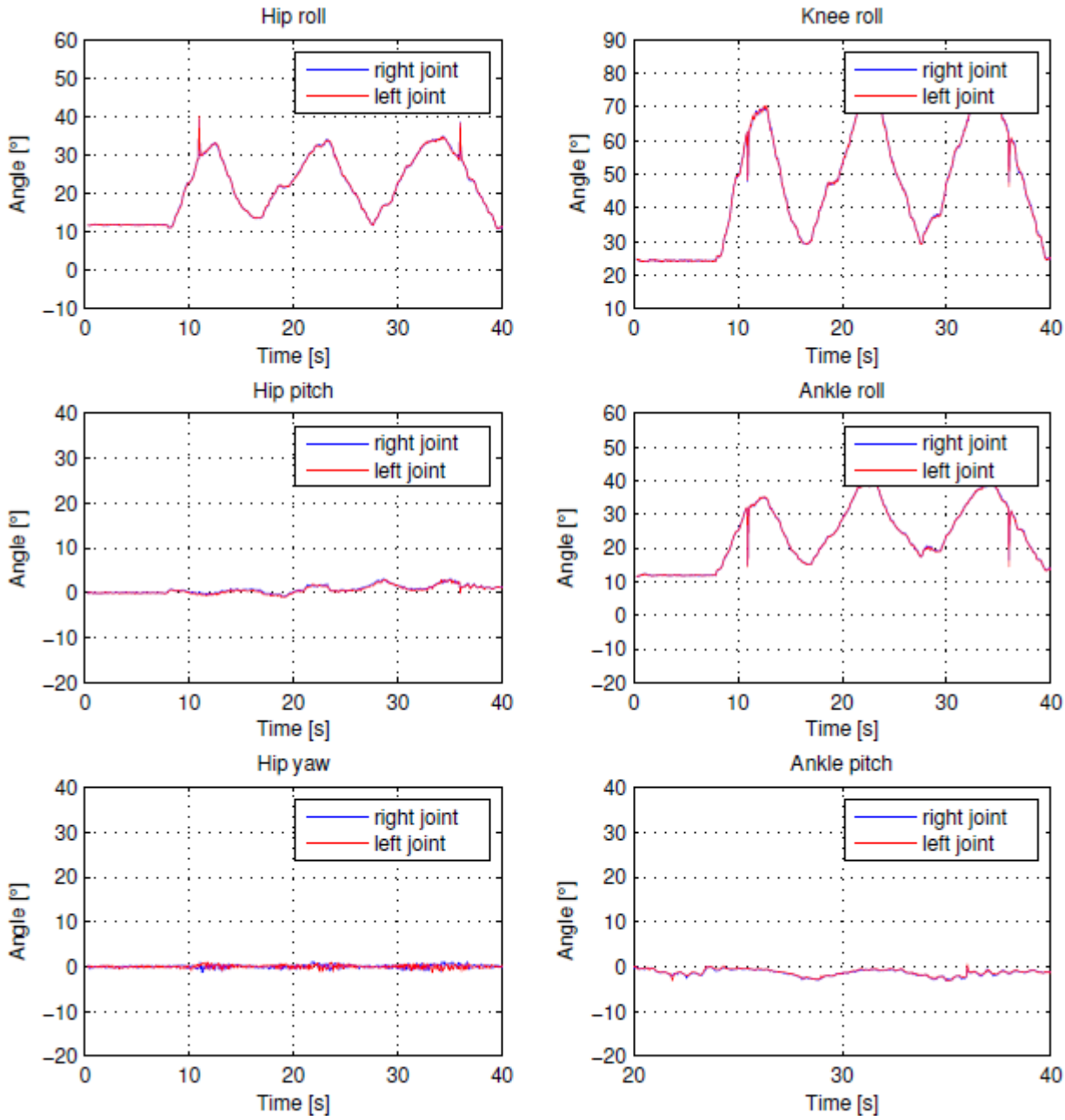


Figure B.1: Robot's joint state evolution during Subject D knee flexion/extension maneuver.

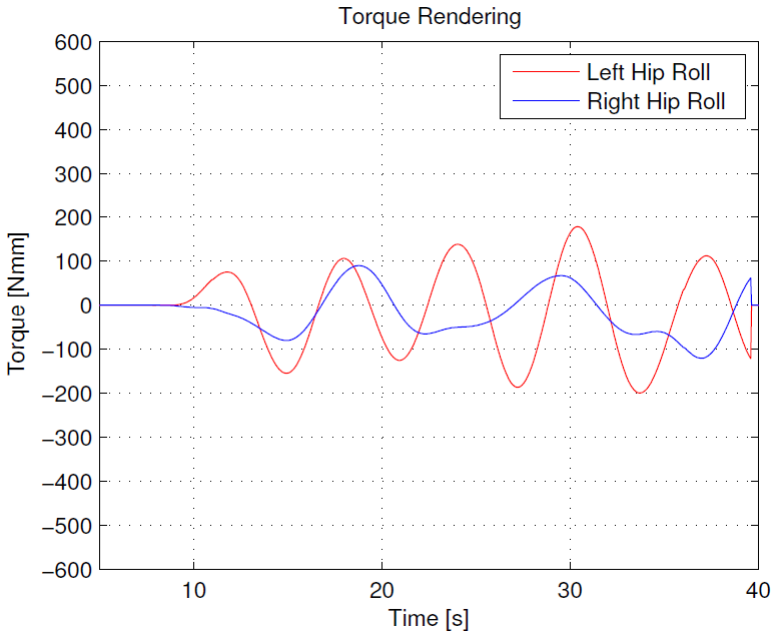


Figure B.2: Torque rendered in the device's second Joint during Subject D knee flexion/extension maneuver.

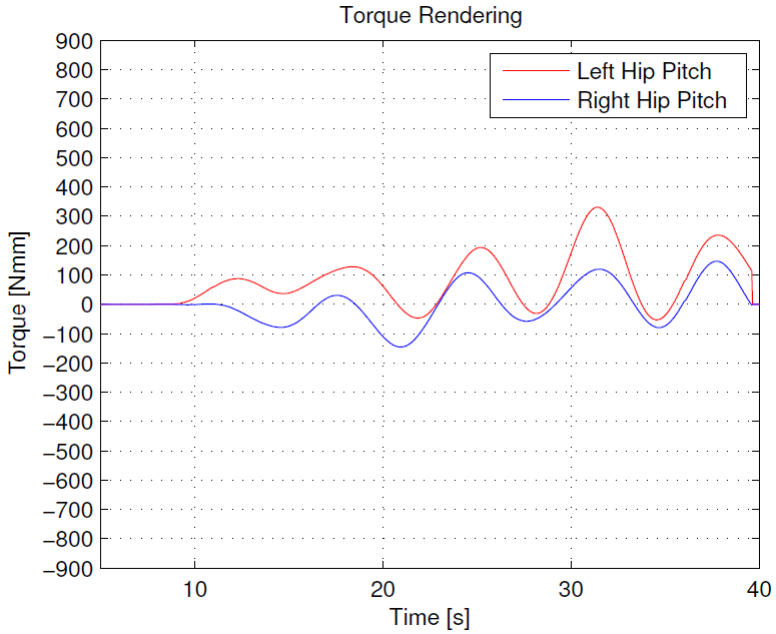


Figure B.3: Torque rendered in the device's first Joint during Subject D knee flexion/extension maneuver.

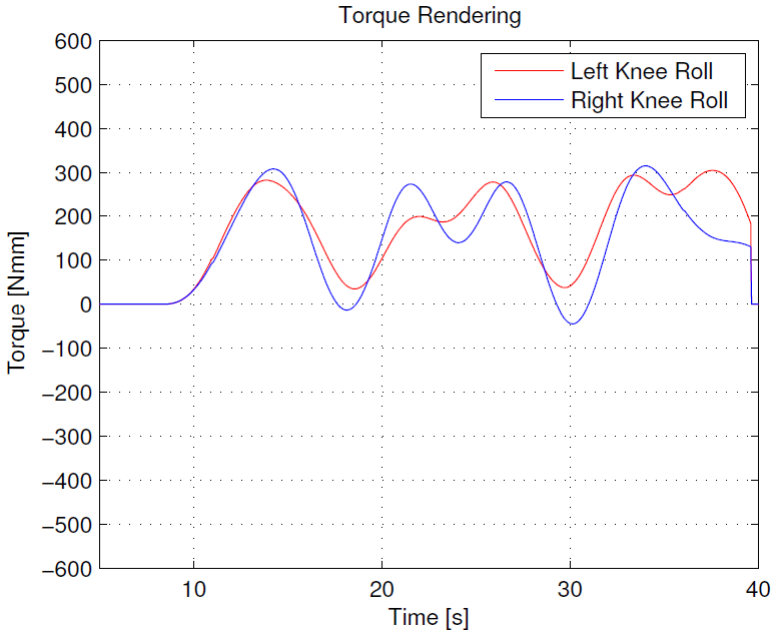


Figure B.4: Torque rendered in the device's third Joint during Subject D knee flexion/extension maneuver.

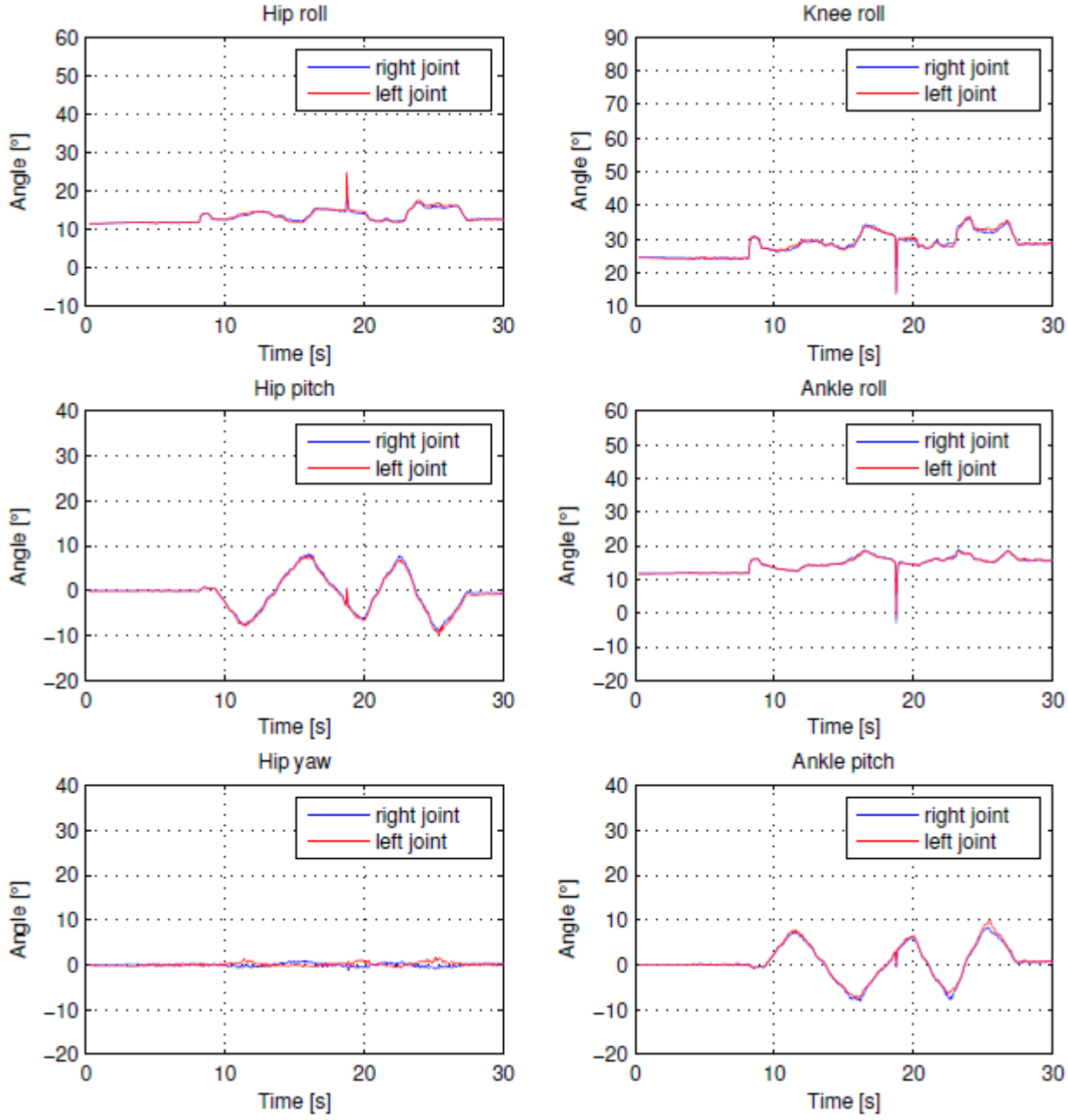


Figure B.5: Robot's joint state evolution during Subject D hip lateral maneuver.

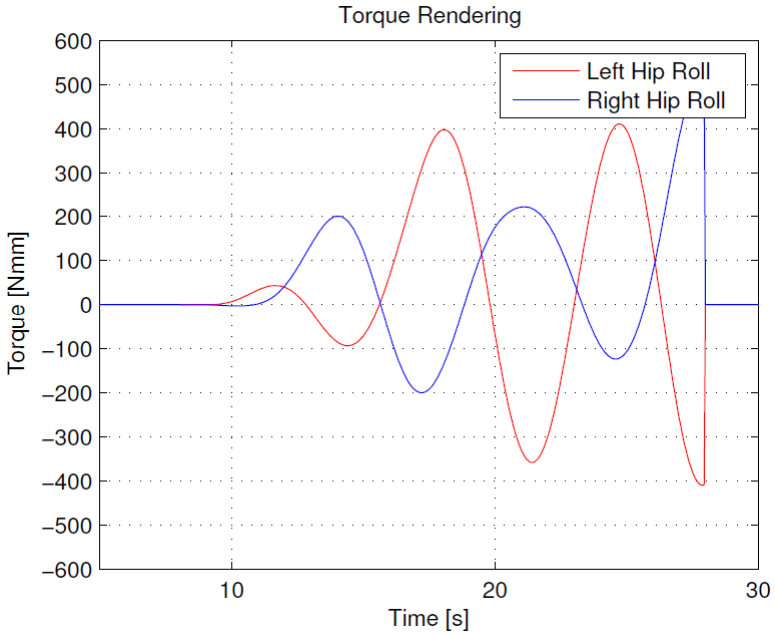


Figure B.6: Torque rendered in the device's second Joint during Subject D hip lateral maneuver.

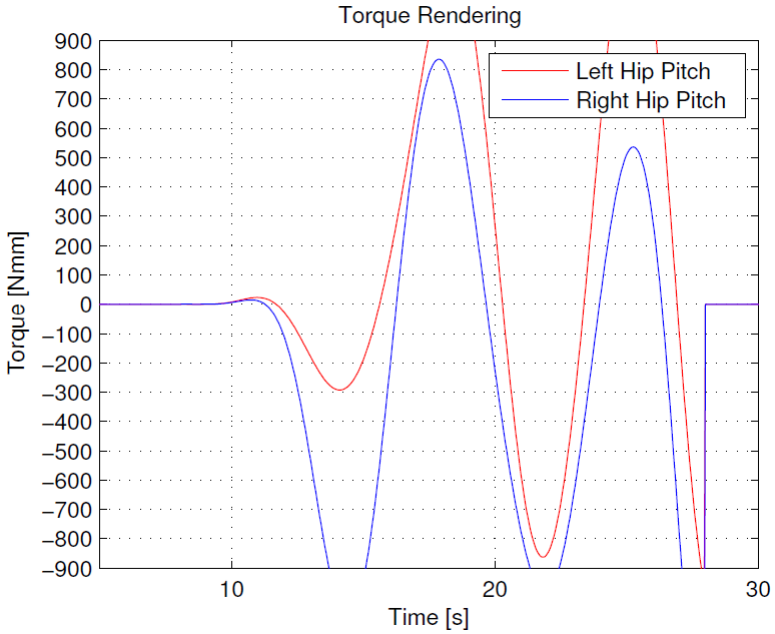


Figure B.7: Torque rendered in the device's first Joint during Subject D hip lateral maneuver.

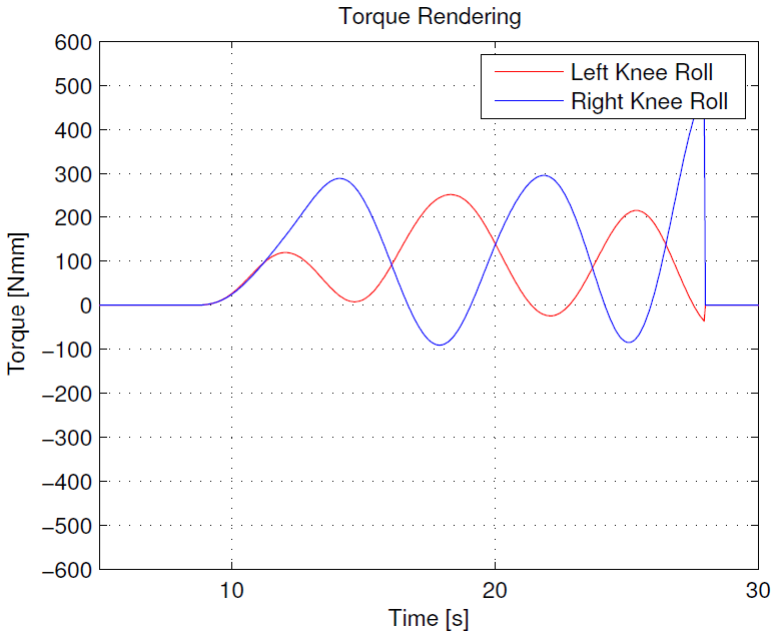


Figure B.8: Torque rendered in the device's third Joint during Subject D hip lateral maneuver.

Appendix C

Software Notes

C.1 PHANTOM Device Drivers and OpenHaptics

Installation of the PHANTOM device software is hereby described for future reference.

The default OpenHaptics Academic Edition drivers are not compatible with Ubuntu 12.04 LTS, the oldest version of the system supported by ROS Hydro. The installation procedure configures a beta version of the PDD (PHANTOM Device Drivers) based on the new Fire Wire kernel driver stack (alias JUJU), that replaces the older *ieee1394* stack. The code base of the new stack is smaller, cleaner, and closer follows modern Linux kernel programming paradigms, allowing for proper functioning of the device in recent systems.

As Ubuntu is a Debian-based Linux distribution, software installation files come frequently in the form of *.deb* packages. Other Linux distributions have similar packages but the instructions for installation and setup may differ. Also, distribution-dependent repositories may have to be updated, and dependencies set, for system components.

Instructions for the migration to the Fire Wire driver stack are detailed next, with special reference to the Ubuntu 12.04 64-bit filesystem. Dependencies and requisites mentioned on the file "HW_userguide_Linux.pdf" are installed, before attempting to install the PHANTOM drivers.

C.1.1 Sensable Software Pre-Requisites

After signing up (ask for credentials), download the OpenHaptics Academic Edition for Linux v3.0 installer from DSC at `http://dsc.sensable.com/3dtouch/OpenHaptics_Academ1c_linux/software_downloads.asp`, and extract it to the /home/user folder.

Download the JUJU PDD PRE-BETA package for Linux from dsc.sensable.com/forum.

Install freeglut:
sudo apt-get install freeglut3-dev

Download Mesa version 7.5.2 from `ftp://ftp.freedesktop.org/pub/mesa/older-versions/7.x/7.5.2/MesaLib-7.5.2.tar.bz2`.

For DRI-based hardware acceleration with Mesa, there are some requirements. Download dri2proto version 1.99.3 from `http://xorg.freedesktop.org/releases/individual/proto/dri2proto-1.99.3.tar.bz2` and extract it to the local folder. Then configure, compile, and install it with the commands (sudo is optional if the working sub-directory is within the /home directory):

./configure
make
sudo make install

Build libDRM and Mesa by doing:
sudo apt-get build-dep libdrm
sudo apt-get build-dep mesa

If necessary, install libwayland0 package to build mesa:
sudo apt-get install lib'wayland0

Install libmotif-dev package:
sudo apt-get install libmotif-dev

Go to the extracted Mesa-7.5.2 folder and run:
./configure –enable-motif

Compile the libraries and install:
make
sudo make install
Make sure the symbolic links for *libGLw.so* and *libGLw.so.1* point properly to the newly build libGLw.so.1.0.0. Go to /usr/local/lib and type *ls -la* to check. If necessary symlink:

sudo ln -s libGLw.so.1.0.0.libGLw.so.1
sudo ln -s libGLw.so.1.0.0.libGLw.so

C.1.2 Sensable Software Installation

Go to /home/user/OpenHapticsAE_Linux_v3_0/PHANTOM/Device/Drivers/64-bit/ folder, and install the drivers .deb package:
sudo dpkg -i phantomdevicedrivers_4.3-3_amd64_deb

Copy the libPHANToMI0.so.4.3 into /usr/lib64 and overwrite the existing one. Go to Linux_JUJU_PDD-64-bit folder and type:
sudo cp -i libPHANToMI0.so.43 /user/lib64

If not created, create the following symbolic links in */usr/lib64* folder:
sudo ln -s libPHANToMIO.so.4.3 libPHANToMIO.so
sudo ln -s libPHANToMIO.so.4.3 libPHANToMIO.so.4

Copy the binary PHANToMConfiguration from Linux_JUJU_PDD_64-bit folder into /usr/sbin and overwrite the old one as well:

sudo cp -i PHANToMConfiguration /usr/sbin

When applied to 64-bit systems, the installation procedure described above defines some libraries out of the /usr/lib folder. This can lead to some errors when trying to run *PHANToMConfiguration* and *PHANToMTest*. To avoid it, ask the dynamic linker to check for other locations, /usr/local/lib and /usr/lib64 too. With root privileges edit the file /etc/ld.so.conf and add the following lines:

/usr/local/lib
/usr/lib64

Ta update the cache run: *sudo ldconfig*

These two small applications are installed by default with the drivers. The PHANToMConfiguration is an application for managing the connected hardware, defining the default device. The PHANToMTest is a small application designed to test the device's capabilities, and calibrate it. They must be executed before using the device.

PHANToMConfiguration and PHANToMTest need libraw1394.so.8 to run, so in /usr/lib/x86_64-linix-gnu do:

sudo ln -s libraw1394.so.11.0.1 libmw1394.so.8

PHANToMConfiguration and PHANToMTest should work by now. However, during the procedure some problems can be experienced with respect to the libPHANToMIO.so.4.3 and PHANToMConfiguration copying process. In case of " PDD Error: raw1394 module not loaded, modprobe raw1394 to correct" occurs, try to rename the existing files and copy the original again from Linux_JUJU_PDD_64-bit folder. Verify symbolic links.

Although the applications work, the device cannot be detected yet. The PHANToM drivers do not have access to the FireWire communication port object, so it falls to the user responsibility to change the access rules. To set the permissions do:

*sudo chmod 777 /dev/fw**

This command have to be run before any call for device applications, or in alternative, added to the skell's Startup script, or to the system's startup routines.

To install OpenHaptics libraries and examples, go to `/home/user/OpenHapticsAE_Linux_v3_0/PHANTOM/Device/Drivers/64-bit/` folder and run the OpenHaptics package, by typing:

sudo dpkg -i openhaptics-ae_3.0-2_amd64.deb

The OpenHaptics Toolkit deploys compilable examples to demonstrate both the device and the libraries capabilities. If correctly installed, the compilable source code should be on the directory /usr/share/3DTouch/examples. All provided documentation is also installed in the hard-drive on the directory /usr/share/3DTouch/doc.

io run the examples, it is necessary to setup the OpenHaptics environment variable 3DTOUCH_BASE. However, the Ubuntu bash does not allow to initiate an environment

variable started by a number. A workaround for this problem is to install *tcsh* shell or *csh* shell, and old C-shell still supported on Linux, and run the commands:

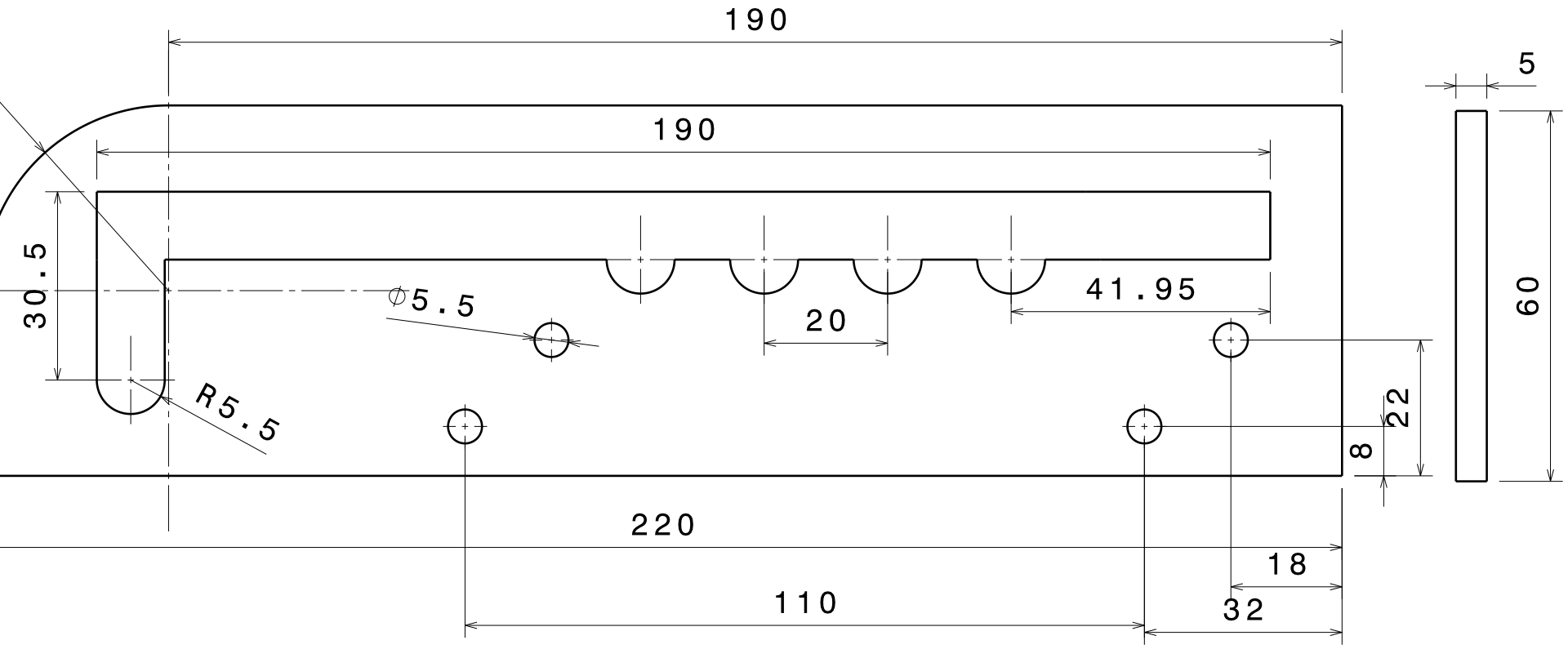
sudo csh
setenv 3DTOUCH_BASE /usr/share/3DTouch

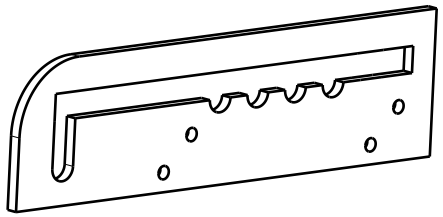
Type *exit* command to return to bash. In an alternative way, it is possible to do:
PATH=$PATH:/usr/share/3DTouch/
export PATH

Before testing the OpenHaptics examples, compile them. For HD and HL there is a *makefile* in both console and graphics examples. Run make, with root privileges if necessary, in each one of those folders to compile the examples.

Appendix D

Tecnical Dossier





PHUA

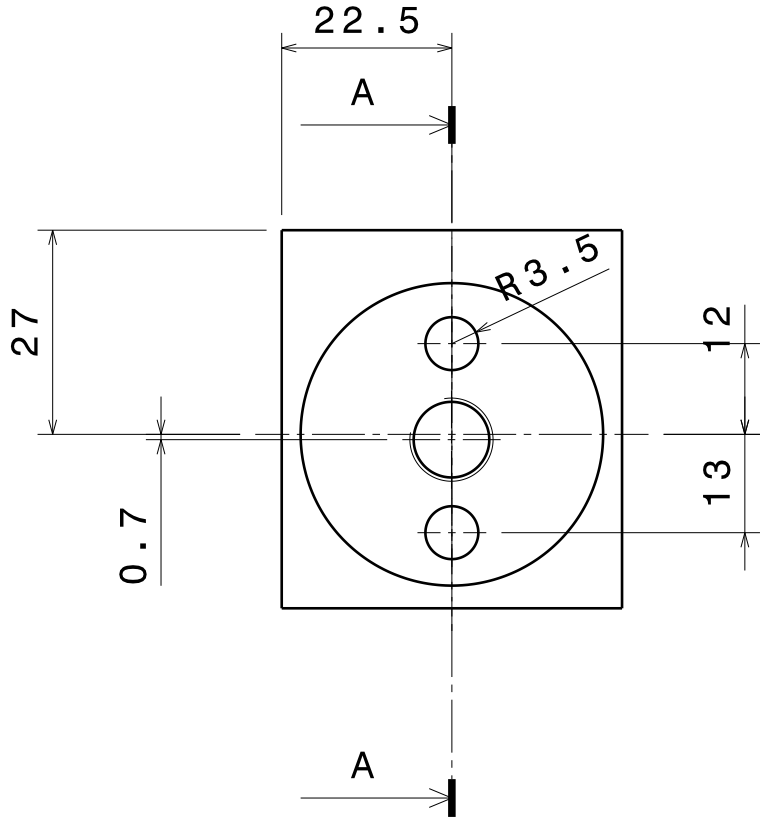
LAR 2015

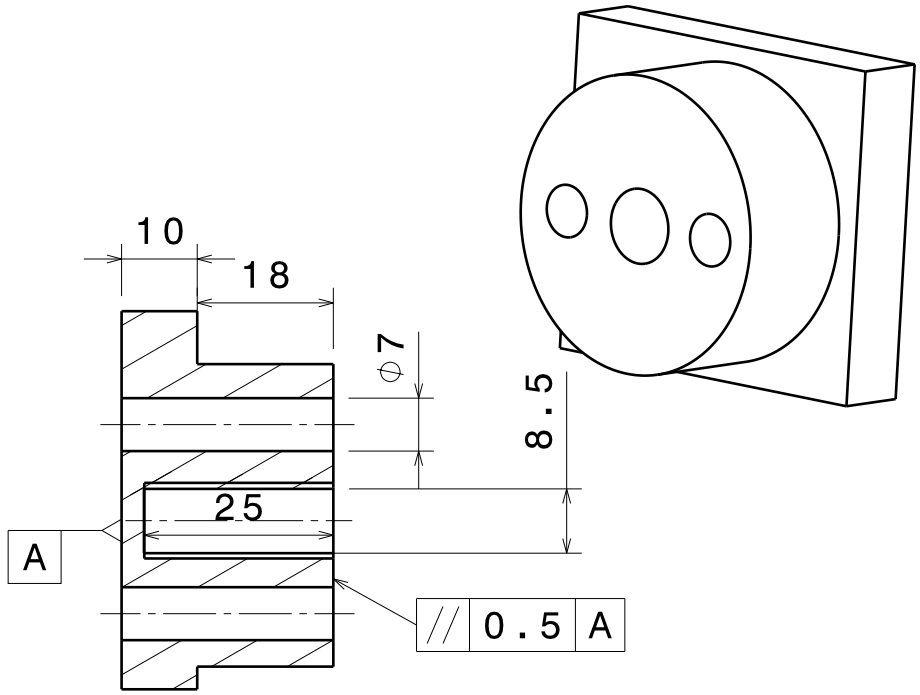
DRAWN BY	DATE
DANIEL MARQUES	04/07/2015
Orientador: Vitor Santos	
Quantidade: 4	

DRAWING TITLE: Guias de Suporte

SIZE	Material	REV
A4	Aluminio AW5083-H111	X

SCALE 1:1 | SHEET 1/1



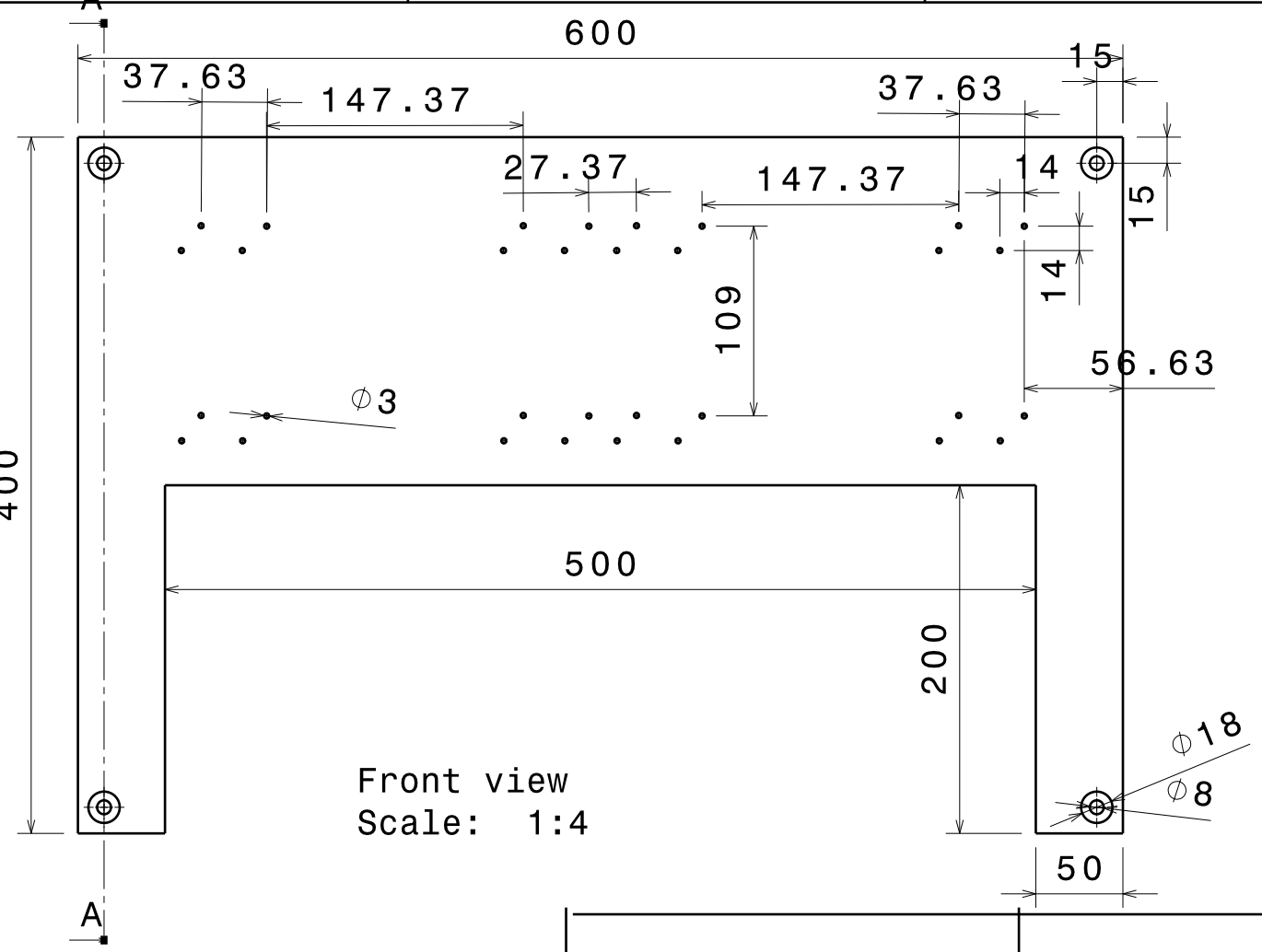


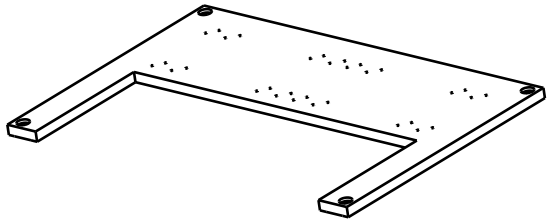
Section view A-A

PHUA

LAR 2015

DRAWN BY: DANIEL MARQUES	DATE: 14/07/2015	DRAWING TITLE: Phantom base suporte	
Orientador: VITOR SANTOS		SIZE: A4	Material: Aluminio AW5083-H111 — REV: X
Quantidade: 4		SCALE: 1:1	SHEET: 1/1





PHUA

LAR 2015

DRAWN BY: DANIEL MARQUES	DATE: 04/07/2015	DRAWING TITLE: Topo de Mesa
Orientador: Vitor Santos		SIZE: A4 / Material: Placa de MDF / REV: X
Quantidade: 1		SCALE 1:4 / SHEET 1/1