University of Trento
University of Brescia
University of Padova
University of Trieste
University of Udine
University IUAV of Venezia

Ph.D. Candidate
Omar A. Daud

HAPTIC SYSTEMS FOR POST-STROKE REHABILITATION: FROM VIRTUAL REALITY TO REMOTE REHABILITATION

Advisor
Prof. R. Oboe

2011

UNIVERSITY OF TRENTO
Graduate School in Structural Engineering
Modelling, Preservation and Controls of Materials and Structures
XXIII Cycle

Ph.D. Program Head: prof. D. Bigoni

Final Examination: 25 March 2011

Board of Examiners:
Prof. Roberto Oboe, Università di Padova
Prof. Robert McMeeking , UCSB
Prof. Ettore Pennestrì, Università di Roma - Tor Vergata
Prof. Antonio De Simone, Sissa Trieste

Summary

Haptic devices are becoming a common and significant tool in the perspective of robotic neurorehabilitation for motor learning, particularly in post-stroke patients. As a standard approach, this kind of devices are used in a local environment, where the patient interacts with a virtual environment recreated in the computer's screen. In this sense, a general framework for virtual reality based rehabilitation was developed. All the features of the framework, such as the control loop and the external communication, as well as the haptic and graphic rendering, were implemented inside Matlab/Simulink using Handshake proSENSE toolbox, guaranteeing a real-time system. As an example, a five-bar linkage haptic device with two active degrees-of-freedom (DOF) was designed and integrated within the proposed framework, as well as a device for grasping operations.

An extension of this standard approach is verified when the therapist is allowed to feel and interact remotely and in real time with the patient. We applied the proposed concept to a single degree-of-freedom master/slave system. One hand orthosis was used as a master device at the therapist's side, while the other was applied to the patient's hand, and used as a slave device. In order to achieve this issue, we proposed two bilateral control systems in order to guarantee an stable interaction between the master and the slave, even in case of variable network conditions (i.e. Internet). By using the master device, the therapist can remotely move the patient's hand and, at the same time, perceive the patient's resistance to the motion, allowing the assessment of important parameters, such as the residual level of spasticity. In this way, it can be remotely assessed the conditions of the patient and consequently can be proposed a proper rehabilitation program.

Sommario

Le interfacce aptiche stanno diventando uno strumento comunemente utilizzato per la riabilitazione nella prospettiva del ri-apprendimento motorio, particolarmente nei pazienti colpiti da ictus. Convenzionalmente, questi dispositivi vengono utilizzati in ambienti locali, dove il paziente interagisce con degli ambienti virtuali ricreati nello schermo del computer. In questo senso è stato realizzato un framework generale per la riabilitazione basata sulla realtà virtuale. Tutte le caratteristiche del framework, come l'algoritmo di controllo e la comunicazione con scheda acquisizione dati, oltre al rendering aptico e grafico, sono stati implementati in Matlab/Simulink usando Handshake proSENSE toolbox, garantendo un sistema che funzioni in tempo reale. Come esempio, è stato progettato ed integrato dentro il framework proposto un dispositivo a pentalatero, a due gradi di libertà, oltre ad un dispositivo per esercitare l'operazione di grasping.

Un'estensione di questo approccio si verifica quando il fisioterapista interagisce da remoto e in tempo reale con il paziente. Questo concetto è stato applicato ad un sistema master/slave ad un grado di libertà. Un'ortesi è stata adoperata come master dal lato del fisioterapista mentre l'altra ortesi come slave dal lato del paziente. Per realizzare ciò, si sono proposti due sistemi di controllo bilaterale con l'obiettivo di rendere stabile l'interazione tra il master e lo slave, nonostante le condizioni variabili nella rete (i.e. internet).

Mediante l'utilizzo del dispositivo master, il fisioterapista può muovere da remoto la mano del paziente ed allo stesso tempo, percepire la sua resistenza al movimento, consentendo la valutazione di parametri importanti, come ad esempio, il livello residuale di spasticità. In questa maniera, si può valutare da remoto le condizioni del paziente, e conseguentemente proporne un appropriato programma di riabilitazione.

To Ximena

Acknowledgements

I would like to thank all the people that directly or indirectly helped me during my studies. Particularly, my gratitude especially goes to Professor Roberto Oboe for being my advisor and giving me the chance to face and achieve this unique challenge.
I am also very thankful to all members of the Mechatronics Research Group of the University of Trento, the Mechatronics Lab and Industrial Robotics Lab of University of Padova - Vicenza branch. An especial thank also goes to all members of the Ohnishi Lab, that so kindly assisted me when I was a visiting student at Keio University.
Finally, I am so obliged to all my family and, especially, to my mother Barberina. Thank you so much.

Trento, March 2011

Omar A. Daud

Published papers

The main results presented in this thesis have been summarized in the following papers:

1) O. A. Daud, F. Biral, R. Oboe, and L. Piron, "A general framework for a rehabilitative oriented haptic interface," in The 11^{th} IEEE International Workshop on Advanced Motion Control, Nagaoka, Japan, March 2010, pp. 685-690.

2) R. Oboe, O. A. Daud, S. Masiero, F. Oscari, and G. Rosati, "Development of a haptic teleoperation system for remote motor and functional evaluation of hand in patients with neurological impairments," in The 11^{th} IEEE International Workshop on Advanced Motion Control, Nagaoka, Japan, March 2010, pp. 518-523.

3) R. Oboe, O. A. Daud, S. Masiero, F. Oscari, and G. Rosati, "Remote evaluation of muscular capabilities in patients with neurological impairments," in The International Symposium on Application of Biomechanical Control Systems to Precision Engineering - Engineering Review of Biological Evolution of Motion Control, ISAB-2010. Fukushima, Japan, July 2010, pp. 102-106.

4) O. A. Daud, F. Biral, R. Oboe, and L. Piron, "Design of a haptic device for finger and hand rehabilitation," in IECON 2010 - 36^{th} Annual Conference on IEEE Industrial Electronics Society, 2010, pp. 2075-2080.

5) R. Oboe, O. A. Daud, S. Masiero, F. Oscari, and G. Rosati, "A teleoperation system for a remote evaluation of the hand in patients with neurological impairments," in The 8^{th} France-Japan and 6^{th} Europe-Asia Congress

on Mechatronics, MECATRONICS-2010. Yokohama, Japan, November 2010, pp. 17-22.

6) Agostini M., Turolla A., Cocco L., Daud O. A., Oboe R., Piron L., "Haptic interface for hand rehabilitation in persons with a stroke". WCPT 2011, Amsterdam, Netherlands, 20-23 June 2011 (submitted).

7) O. A. Daud, R. Oboe, M. Agostini and A. Turolla, "Performance Evaluation of a VR-based Hand and Finger Rehabilitation Program," in Industrial Electronics, 2011. ISIE 2011. IEEE International Symposium on, 27-30 June 2011 (submitted).

Contents

Chapter 1

Introduction

Hand and finger dexterity is important as it is fundamental for many activities a person needs to perform in order to be independent. Stroke can reduce a person's movement functionality because of the resulting death of associated brain cells. It is not only stroke that damage the motor and sensory system, but also other pathologies, such as musculoskeletal diseases. However, stroke remains the leading cause of serious and long-term disability today. Many studies attest that initiating the physical therapy immediately after a stroke often grants a higher quality of life for the patient. Haptics and virtual environments offer the chance to improve the traditional methods of rehabilitation.

1.1 Motivation

When a person becomes unable to interact physically with its immediate environment, and so unable to achieve their personal goals, because of injury or disease, technology-based solutions are needed for relearning.

Research programs dealing with the development of personal robots, robotic therapy, and tele-rehabilitation services have increased in the past ten years and will continue because the ever-increasing ability of health care to allow people live longer. Rehabilitation robotics is projected to quickly grow in the coming decades (Kassler, 1993).

Nowadays, almost all the activities in physical therapy and training robots has been focused on relearning the movement ability of post stroke patients.

The main reason of such emphasis is the relatively large number of patients in these conditions, for which the associated rehabilitation costs are high.

Because the human neuromuscular system presents use-dependent plasticity, which states that the use modifies the properties of neurons and muscles, including the pattern of their connectivity, and hence their function (Sawaki, 2005), post stroke patients can experience important benefits with intensive rehabilitation.

In fact, the use of haptic interfaces is becoming a common approach for treating the disability induced by stroke or musculoskeletal disorders. For doctors and operational therapists, a computer-based system is highly desired, since it is an efficient measurement system, and can provide intense rehabilitation exercise. Therefore, haptic interfaces not only have the potential to rehabilitate and help stroke survivors in regaining essential skills for their daily living activities, but also give objective information to doctors and therapist about the rehabilitation process.

1.2 Background

The process of neurorehabilitation tries to take advantage of this use-dependent plasticity in order to help people relearn the movements that were lost after neuromuscular injuries or diseases. Neurorehabilitation is typically provided by prepared therapists, that usually are required to give hands-on assistance as well.

Because neurorehabilitation is a hard and time-consuming activity, in recent years health programs have put some limits on the amount of therapy, in an effort to contain health care costs. Nevertheless, at the same time, there has been an increasing scientific evidence that more therapy can increase the movement relearning via use-dependent plasticity. Neurorehabilitation is a logical target for robotics, because the amount of recovery is linked with the amount of repetition. Advanced robotics can deliver either continuous therapy at a lower cost or supporting assistance to physical therapists, giving patients a better chance for an effective rehabilitation.

An important issue in robotic therapy is how to optimize use-dependent plasticity. Researchers must determine what the robot should do in order to match the patient's movement effort, and consequently maximize the recovery in terms of movement ability. Meeting this challenge involves the solution of these two key problems: determining appropriate movement tasks (what movements should patients exercise and what type of feedback should they

receive, taking into account their performance), and determining an appropriate pattern of mechanical input to the patient during these tasks (what forces should the robot apply to the patient's limb to stimulate plasticity). The prescription of movement tasks and the mechanical inputs fundamentally constrain the mechanical and control design of the robotic therapy device.

Nowadays, almost all the activities that deal with physical therapy and training robots have been focused on relearning the movement abilities the patients had before stroke. The main reason for this emphasis is explained by the large amount of stroke patients, the high costs associated with this kind of rehabilitation, and the large gains that can be obtained by stimulating use-dependent plasticity.

1.3 Stroke

A stroke occurs when a blood vessel that carries oxygen and nutrients to the brain is either blocked by a clot or bursts. When that happens, part of the brain cannot get the blood (and oxygen) it needs, so it starts to die. Stroke can be caused either by a clot obstructing the flow of blood to the brain (called an ischemic stroke) or by a blood vessel rupturing and preventing blood flow to the brain (called a hemorrhagic stroke).

1.3.1 The effects of stroke

The brain is an extremely complex organ that controls various body functions. If a stroke occurs and blood flow can't reach the region that controls a particular body function, that part of the body won't work as it should. If the stroke occurs toward the back of the brain, for instance, it's likely that some disability involving vision will result. The effects of a stroke depend primarily on the location of the obstruction and the extent of brain tissue affected. However, because one side of the brain controls the opposite side of the body, a stroke affecting one side will result in neurological complications on the side of the body it affects.

- Right Brain: If the stroke occurs in the brain's right side, the left side of the body (and the right side of the face) will be affected, which could produce any or all of the following:
  - Paralysis on the left side of the body.
  - Vision problems.

  - Quick, inquisitive behavioral style.
  - Memory loss.

- Left Brain: If the stroke occurs in the left side of the brain, the right side of the body will be affected, producing some or all of the following:
  - Paralysis on the right side of the body.
  - Speech/language problems.
  - Slow, cautious behavioral style.
  - Memory loss.

- Brain Stem: When stroke occurs in the brain stem, depending on the severity of the injury, it can affect both sides of the body and may leave someone in a "locked-in" state. When a locked-in state occurs, the patient is generally unable to speak or achieve any movement below the neck.

Therefore, common motor impairments that result from stroke are hemiparesis, which refers to weakness on one side of the body; abnormal tone, which refers to an increase in the felt resistance to passive movement of a limb; impaired coordination, which can manifest itself as an apparent loss in control degrees of freedom and decreased smoothness of movement; and impaired somatosensation, which refers to a decreased ability to sense the movement of body parts. Secondary impairments include muscle atrophy and disuse-related shortening and stiffening of soft tissue, resulting in decreased passive range of motion of joints. Often the ability to open the hand, and to a slightly lesser extent close the hand, is dramatically decreased.

1.3.2 The impact of stroke

Stroke is the third largest cause of death, ranking behind cardiovascular diseases and all forms of cancer. Stroke is the leading cause of serious and long-term disability today. Hence, the socio-economic impact of stroke is considerable world-wide. Stroke is assuming an increasing impact in terms of media attention, patient and carer knowledge, service developments and research. It is estimated that there are 4.5 million deaths a year from stroke in the world and over 9 million stroke survivors. Almost one in four men and nearly one in five women aged 45 years can expect to have a stroke if they live to their 85^{th} year. The overall incidence rate of stroke is around 2-2.5

per thousand population. The risk of recurrence over 5 years is 15-40%. It is estimated that by 2023 there will be an absolute increase in the number of patients experiencing a first ever stroke of about 30% compared with 1983 (Wolfe, 2000).

1.4 Upper-limb robotic therapy

The literature suggests that the degree of initial impairment and recovery of arm paresis play an important role in regaining arm function and reducing the limitations of the movement activity in chronic stroke patients (Harris and Eng, 2007). Therefore, many patients might receive multidisciplinary rehabilitation immediately after a stroke. However, despite intensive rehabilitation efforts, only 5% to 20% approximately reach complete functional recovery (Nakayama et al., 1994); in other words four of five patients leave rehabilitation with restricted arm function. Thus, there still exists an urgent need for new inpatient and outpatient rehabilitation and training strategies that match the specific patients' requirements (Barker and Brauer, 2005).

Another important issue relies on the fact that while the patient uses robotic rehabilitation devices he/she is more motivated, and hence encouraged; experimental evidence states that the number of repetitions were largely increased. Robotic training devices, therefore, allow intensive, frequent, and repetitive exercise, according to the principles of motor learning. Recently, the first results describing an integration of the existing robotic devices in post-stroke rehabilitation where proposed by (Prange et al., 2006). However, contrary to the remarkable number of publications about electromechanical technologies (Jaeger, 2006), scientific evidence of the benefits of these technologies, which could justify costs and effort, is still lacking. There is, therefore, a need for a systematic evaluation in the form of a systematic review of the available literature to assess the effectiveness and acceptability of these robotic training devices.

1.5 Robot-assisted arm training devices

Usually, robotic training devices provide passive movement to the patient's arm. However, some devices include other modalities that supplies a partial assistance of movement up to a fixed resistance. Some other devices may assist active movements of an isolated joint, like in continuous passive motion, while other devices can be able to move multiple segments to per-

form reaching-like movements. Robotic rehabilitation can be achieved by, for example, varying the force, decreasing assistance, increasing resistance, and expanding the movement amplitude. Moreover, some devices, such as the Bi-Manu-Track and the MIME, may be used to provide bimanual exercise: the device simultaneously moves (mirrors) the affected limb passively, steered by the non-paretic limb. Several robotic systems incorporate more than one modality into a single device. In the following, the most famous devices that have been clinically tested for upper-limb rehabilitation are described.

1.5.1 MIT-MANUS

The first robotic system that was clinically tested was the MIT-Manus; a two degree-of-freedom robot manipulator that assist patients by tabletop arm movements. The MIT-MANUS is a planar two-joint arm that uses a selective compliant assembly robot arm (SCARA) configuration. Such configuration allows the MIT-MANUS performing planar movements with a wide range of forces to be applied to the arm. The MIT-MANUS assists the patient by moving the arm across the tabletop as the patient interacts with a virtual environment. As shown in Fig. 1.1a, the patient moves a cursor into a target that changes locations on the computer screen. Additional modules have been developed in order to allow vertical motion, wrist motion, and hand grasping.

The first clinical test consisted in comparing the motor recovery of acute stroke patients that received additional therapy over the ones that received the conventional one (Aisen et al., 1997). These patients received additional robotic therapy for an hour a day, during five days per week; this was conducted for several weeks. This group of patients had a better recovery of the arm movement ability than the other group, according to clinical scales, without any increase in adverse effects, such as shoulder pain. The improvements might subjectively be characterized as small, but somewhat meaningful. These improvements were supported by three-year follow-up. This first study demonstrated that acute stroke patients, who received extra therapy, recovered in a better way. Further studies confirmed that robotic therapy can also give benefits to chronic stroke patients (Fasoli et al., 2003).

1.5.2 MIME

The MIME, that stands for Mirror Image Movement Enabler, uses a six-degree-of-freedom, industrial robot manipulator (PUMA 560) to assist the movement of the patient's arm, as shown in Fig. 1.1b. It applies forces

to the paretic limb through a customized forearm splint. The robot moves the forearm through a large range of positions and orientations in a three-dimensional space. A six-axis sensor measures the forces and torques between the robot and the paretic limb. Several modes of robot-assisted movement have been implemented with MIME, including passive, active-assisted, and active-constrained, as well as a bimanual mode (the unimpaired limb assists the impaired one).

The first clinical tests found out that chronic stroke patients improved their movement ability compared to the patients who received the conventional tabletop exercises with an occupational therapist (Lum et al., 2002).

1.5.3 ARM-Guide

The Assisted Rehabilitation and Measurement (ARM) Guide, illustrated in Fig. 1.1c, measures and applies assisting or resistive forces to linear reaching movements across a wide workspace. The ARM Guide consists of a hand piece that is attached to a linear track, and is actuated by a DC servomotor. The track can be oriented at different yaw and pitch angles to allow reaching-like operations through different parts of the workspace.

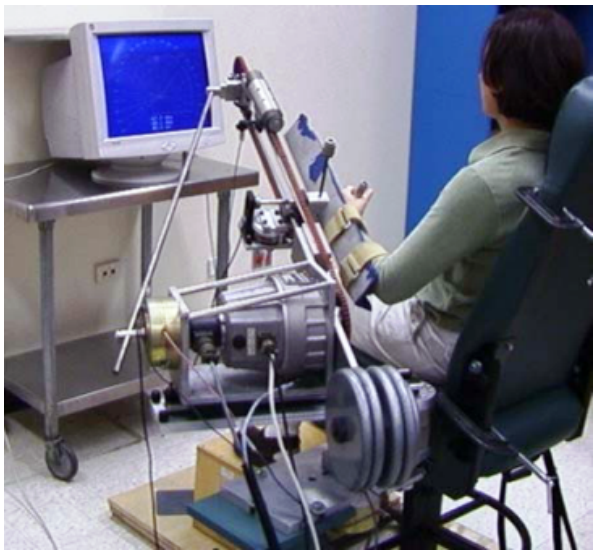
Chronic stroke patients who received assistance with this robot improved their movement ability (Kahn et al., 2006). However, they improved as much as the control group that simply practised a matched number of reaches without robotic assistance. This suggests that movement effort is an important factor for recovery.

1.5.4 Bi-Manu-Track

The Bi-Manu-Track is a 2x1 degree-of-freedom robot that enables hemiparetic patients to bilaterally practice two different movement cycles: forearm pronation/supination and wrist flexion/extension. The handles of the robot perform a rocker-like rotary motion in either a mirror image or parallel fashion. Three different control modes are possible: passive-passive, active-passive (the non affected limb drives the affected one), and active-active (the affected limb has to overcome an initial isometric resistance). Amplitude, speed, and resistances can be set individually. An extensive clinical study confirms the effectiveness of this device (Hesse et al., 2005).

(a)

(b)



(c)

(d)

Figure 1.1: Robot-aided systems: (a) The MIT-MANUS device. (b) The MIME device. (c) The ARM-GUIDE device. (d) The BI-MANU-TRACK device.

1.5.5 Other Devices

Other relevant devices that has been clinically tested are briefly introduced in these paragraphs.

The GENTLE/s system uses a commercial robot, the HapticMaster, to assist patient's movement as the patient interacts with a virtual environment. The HapticMaster allows four degrees of freedom and achieves a high bandwidth of force control using force feedback. Chronic stroke patients who exercised with GENTLE/s improved their movement ability (Amirabdollahian et al., 2007).

The Rutgers Master II-ND (ND stands for New Design) Force-Feedback Glove is a haptic interface designed for dextrous interactions with virtual environments. The glove provides force feedback up to 16 [N] to each thumb, index, middle, and ring fingertips. It uses custom pneumatic actuators arranged in a direct-drive configuration in the palm. This device is used for helping to extend or flex the fingers, and has been shown to improve hand movement ability of chronic stroke patients (Merians et al., 2006). Another device is the CyberGlove which is a fully instrumented glove that provides up to 22 high-accuracy joint-angle measurements. It uses proprietary resistive bend-sensing technology to accurately transform hand and finger motions into real-time digital joint-angle data. By integrating both systems, Rutgers Master II-ND glove and CyberGlove, chronic stroke patients were trained with virtual reality (Adamovich et al., 2005).

The PHANTOM haptic interface permits touch interactions between a human operator and virtual or remote (real) environment. The PHANTOM is a desktop device that provides a force-reflecting interface between a human operator and a virtual environment on a computer screen, in fact, it measures the fingertip position and exerts a force on it. A system that integrates an immersive virtual-reality display with the PHANTOM device for rehabilitation purposes is described in (Patton et al., 2004).

NeReBot is a three-DOF wire-based robot that can slowly move the arm of a post stroke patient. Acute stroke patients who received additional therapy with the NeReBot recovered significantly better comparing to those who received just conventional therapy (Masiero et al., 2007). RehaRob uses an industrial robot arm to move the post-stroke patient's arm along arbitrary trajectories (Fazekas et al., 2006).

Several other robotic therapy devices are currently available, for example, the ARM-In (Nef and Riener, 2005) and Pneu-WREX systems (Wolbrecht et al., 2006), which are exoskeletons that assist nearly naturalistic movement

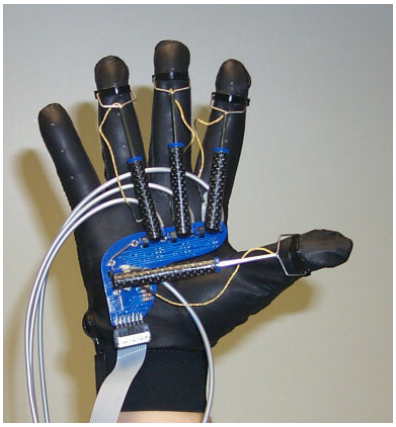
of the arm, achieving a wide range of force control.

1.6 Robotic-assisted telerehabilitation approach

As described before, almost all the robotic devices used for rehabilitation has been focused on robots that might be attach to the patient's upper limb. In fact, these devices are used in a local scenario, where the patient interacts with virtual manipulation experiments, presented on the computer screen.

Another important consideration deals with the fact that as demand soars, rehabilitation facilities must keep pace with both, the standards of care and the unrelenting cost-containment pressures of today's health care environment, in which inpatient post stroke stays have been shortened by two-thirds in the last decade (Ottenbacher et al., 2004). Thus, a health care delivery system is now promoting, to a sicker population, inpatient rehabilitation in less than half the time, and increasingly complementing treatment with outpatient care. As a consequence, an emerging approach towards remote rehabilitation is taking place (Carignan and Krebs, 2006). In fact, in these recent years, remote rehabilitation systems are taking a higher interest due to the multiple applications and advantages it actually has (see Fig. 1.3). Specifically, robotic-assisted telerehabilitation offers innovative, interactive, and precisely reproducible therapies that can be performed for an extended duration, and can be consistently implemented from site to site. Therefore, as the Internet access is already available in almost every home, the idea of doing remote rehabilitation is taking a wide consent because it can grant a high possibility of success. In fact, it can bring medical care program to the patient's house, as well as reducing the patient's hospitalization time and the associated costs. Nowadays, there are some low cost-complexity systems available that use devices such as force feedback joysticks and steering wheels for bringing medical care at the patient's home (Feng and Winters, 2005).

(a)



(b)

(c)

(d)

Figure 1.2: Other robotic devices used for rehabilitation: (a) The GENTLE/s device. (b) The Rutgers Master II-ND Glove. (c) The Cyberglove. (d) The PHANTOM device.

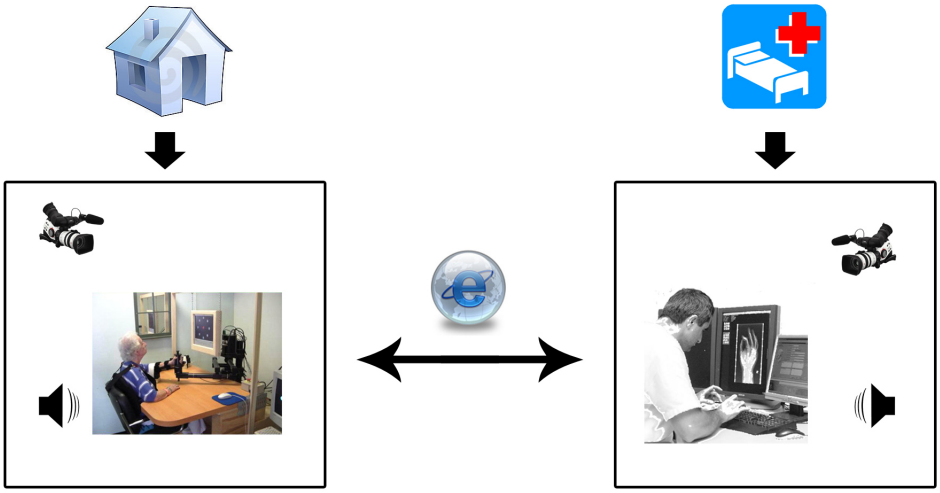


Figure 1.3: Remote rehabilitation representation.

[1]For further information, see H.F. Machiel Van der Loos and David J. Reinkensmeyer, *Rehabilitation and Health Care Robotics*, in Bruno Siciliano and Oussama Khatib, *Springer Handbook of Robotics*, Berlin, Heidelberg, 2008, pp. 1223-1251.

Chapter 2

Haptics

The word haptics, which is believed it comes from the Greek word haptesthai, means "related to the sense of touch". According to the psychology and neuroscience literature, haptics corresponds to the study of human touch by kinesthetic and cutaneous receptors, that are associated with perception and manipulation. In the robotics and virtual reality literature, haptics is largely defined as real and simulated touch interactions between robots, humans, and real, remote, or simulated environments, in the various combinations.

2.1 Overview

Haptic technology aims at providing touch sensations in human operators. In order to improve the human operator performance in simulated and teleoperated environments, haptic interfaces try to generate a compelling sensation as close as possible to the one the operator would experience when directly touching a real environment. Haptic interfaces attempt to replicate touch experience of manipulating or perceiving a virtual or real environment through mechatronic devices and computer control.

A haptic interface consists of a haptic device and a control computer with a software that relates the human operator inputs into haptic information display. While the low-level design of haptic interfaces varies widely depending on the application, their operation generally follows the haptic loop illustrated in Fig. 2.1. First, the haptic device senses an operator input, which may be a position (and its derivatives), force, muscle activity, etc. Second, the

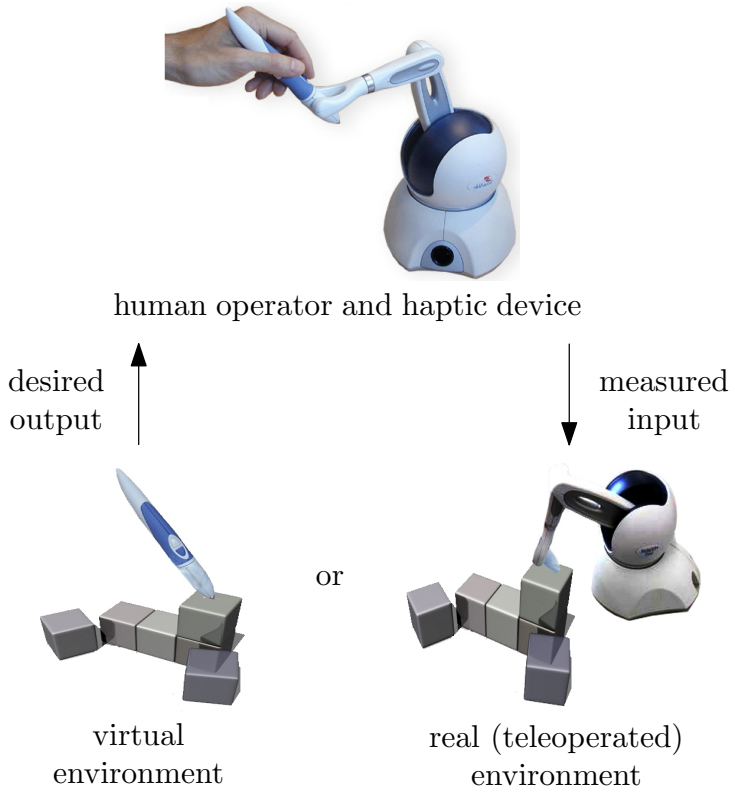


Figure 2.1: The haptic loop of a generic haptic interface.

sensed input is applied to a virtual or teleoperated environment. For a virtual environment, the effect of the operator's input on virtual objects and the subsequent response to be displayed to the operator are computed based on models and a haptic rendering algorithm. In teleoperation, a remote manipulator attempts to track the operator's input. When the remote manipulator interacts with the real environment, haptic information is measured and sent back to the actuators in order to reproduce the haptic sensation in the human operator.

There are many robotic designs that can be used as haptic devices. The most common are exoskeletons, actuated grippers, parallel and serial manipulators, small-workspace mouse-like devices, and large-workspace devices that capture whole arm movement.

As human beings integrate kinesthetic and cutaneous information, with

motion and control cues to form haptic perceptions, haptic devices would ideally include both force and tactile displays. However, this has been rarely done due to size and weight limitations of the actuators.

2.2 Human haptics

2.2.1 Anatomy and physiology

Inside the human nervous system there are two functions that play an essential role in generating haptic perceptions: these are kinesthesia and tactile sensing. By one hand, kinesthesia deals with the internal sensing of forces and displacements inside muscles, tendons, and joints. It is realized by muscle spindles, which transduce stretch of muscles, and Golgi tendon organs, which transduce joint rotation, especially at the extremes of motion. In principle, these and similar receptors could be stimulated directly to produce haptic sensations. By the other hand, the tactile sensing deals with the sensation of deformations of the skin. Haptics incorporates both, and is associated with a manipulating or exploring activity.

2.2.2 Psychophysics

Psychophysics is the field of science that studies the physical capabilities of the senses. It has been an important source of data for the designing and developing of haptic devices. Their major contribution refers to methodologies that haptic researchers have applied in order to determine the required capabilities that haptic devices might have.

Some of the main psychophysical methods that have been successfully applied to haptics, include threshold measurement by the method of limits and adaptive up-down methods.

2.2.3 Psychology: exploratory procedures

The Exploratory Procedures (EP), which are characteristic of human haptic exploration, where defined as stereotyped hand motions (Lederman and Klatzky, 1987). These are eight (see Fig. 2.2), and the property for which are optimal are:

1. lateral motion (texture).
2. pressure (hardness).

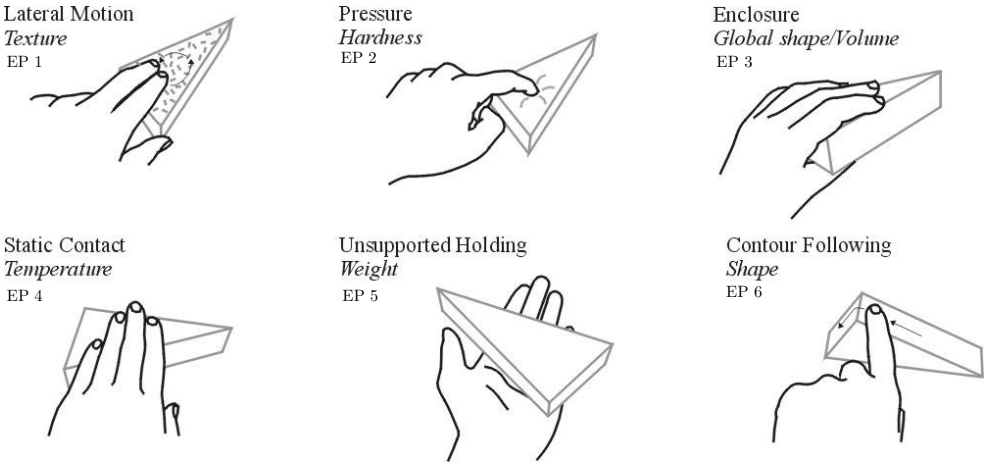


Figure 2.2: Six of the eight human exploratory procedures.

3. static contact (temperature).
4. unsupported holding (weight).
5. enclosure (global shape, volume).
6. contour following (exact shape, volume).
7. part motion test (part motion).
8. function testing (specific function).

Each of these exploratory procedures is a bimanual task involving contact with interior surfaces of the hand, motion of the wrist, various degrees of freedom of the hand, tactile and temperature sensors in the skin (e.g., EPs 1 and 3), and kinesthetic sensors in the arm (EP 4). A haptic device able to reproduce all of these exploratory procedures is far beyond today's state of the art. However, these results are significant for the designing, since they allow to deduce the requirements for the device.

2.3 Haptic devices

A haptic device is a manipulator with sensors and actuators, designed for reproducing the sense of touch in a human operator from a virtual or remote

environment. It can only represent a point-based contact, usually the three X-Y-Z directions, sometimes also the rotational torque. When interfaced with a virtual world, the end-effector position of the device is associated in the virtual environment to the so-called avatar, which is represented by a little ball (or any other virtual object).

2.3.1 Control architecture

Haptic devices can be classified as impedance and admittance devices, depending on their control architecture.

- Impedance control: The operator moves the device, and the device applies a force vector to the operator according to the computed behavior of the simulated object or surface. The paradigm is this: "displacement in - force out".
- Admittance control: The device measures the forces that the operator exerts on it, reacting with the corresponding motion (acceleration, velocity, position). The paradigm is: "force in - displacement out".

Admittance control and impedance control are dual. What is difficult to implement for one, is easy for the other, and viceversa. In fact, a haptic device of impedance type is back-drivable, have low friction and inertia, and have force-source actuators. A commonly used impedance haptic device is the PHANTOM. Instead, a haptic device of admittance type is non-back-drivable and have velocity-source actuators. The velocity is controlled with a high-bandwidth low-level controller, and is assumed to be independent of applied external forces. The HapticMaster device operates under admittance control.

Choosing an admittance or an impedance control architecture determines the design of the device, and consequently the controlling architecture. An important amount of haptic devices implements impedance control today, because of the their low cost.

2.3.2 Mechanisms

An important issue on creating high-fidelity haptic sensation in the operator side is represented by the design of the mechanism to be used by the haptic device. The main requirements of the mechanism for an impedance haptic device are low inertia, high stiffness, and good kinematic conditioning throughout the workspace. Such workspace must be designed to effectively

match the appropriate human limb, basically the finger or arm. The weight of the mechanism should be minimized, as it is perceived by the operator as weight and inertia of the virtual or teleoperated environment. Kinematic singularities should be avoided as they are harmful to haptic interfaces because they create directions in space in which the end-effector cannot be moved; this phenomenon usually generates disturbances on the illusion of haptic contact with virtual objects. High transmission ratios must be avoided as they introduce high levels of friction. This constraint requires haptic interfaces to make high demands on actuator performance.

An ideal haptic device can move freely in any direction while the singular configurations and their operating effects around the neighborhood are avoided. Usually, the kinematic performance of a haptic device is derived from the mechanism's Jacobian matrix. There are three well-known methods that measure such performance:

- Manipulability (Yoshikawa, 1985): which corresponds to the product of the singular values of the mechanism's Jacobian matrix.
- Mechanism isotropy (Salisbury and Craig, 1982): determining the ratio between the minimum and the maximum singular values of the mechanism's Jacobian matrix.
- Minimum force output (Buttolo and Hannaford, 1995): maximizing the force output in the worst direction.

What defines the workspace of a haptic device is the matching between its workspace and the one of the targeted human limb workspace, which can be determined by using anthropometric data. The performance goals, such as low inertia and avoidance of kinematic singularities, must be formalized into a quantitative performance measure that can be computed for any candidate design. In fact, the following aspects must be taken into account:

- Uniformity of kinematic conditioning throughout the target workspace.
- Favoring designs with lower inertia.
- Guaranteeing that the target workspace is reachable.

These aspects operate in a single point in the space and thus must be integrated over the entire workspace to give a qualitative evaluation of a proposed haptic device design.

2.3.3 Sensing

The state of a haptic device can be measured by sensors. This state can be modified by the operator's applied position/force, the haptic control law, and/or device and environment dynamics. The operator's input is sensed as an applied position or an applied force.

Encoders are used as position sensors on the joints of the haptic devices. Many haptic applications, such as the rendering of virtual environments with damping, require velocity measurement. Velocity is typically obtained by numerical differentiation of the sensed position.

Force sensors, such as strain gauges and/or load cells, are used to measure the operator's applied force. This type of sensors are used by admittance-controlled haptic devices. However, these sensors are also used by impedance-controlled devices as a mechanism for canceling undesirable friction or other forms of disturbances.

2.3.4 Actuators and mechanical transmission

The performance of a haptic device depends strongly on the actuator properties and the mechanical transmission between the actuator and the haptic interaction point. In fact, the main requirements for actuators and mechanical transmission for impedance-type haptic devices are: low inertia, low friction, low torque ripple, back-driveability, and low backlash. In addition, if the design is such that the actuator itself moves as the user's position changes, a higher power-to-weight ratio is desired.

Although closed-loop force control has been used for haptic display in impedance devices, most often the mechanism is designed to have sufficiently low friction and inertia so that open-loop force control is accurate enough.

2.4 Haptic rendering in virtual environments

Haptic rendering in impedance devices is the process of computing and generating the forces that are fed back due to the operator's interaction with virtual objects.

An important property of haptic systems is that their timing constraints are quite severe. In fact, tactile receptors in the fingers of human beings are known to respond up to 10 [kHz] (Shimoga, 1993), while audio range of frequencies are up to 20 [kHz]. In order to reproduce either the haptic sensation or the audio response of a hard contact simulation, at least a sampling rate

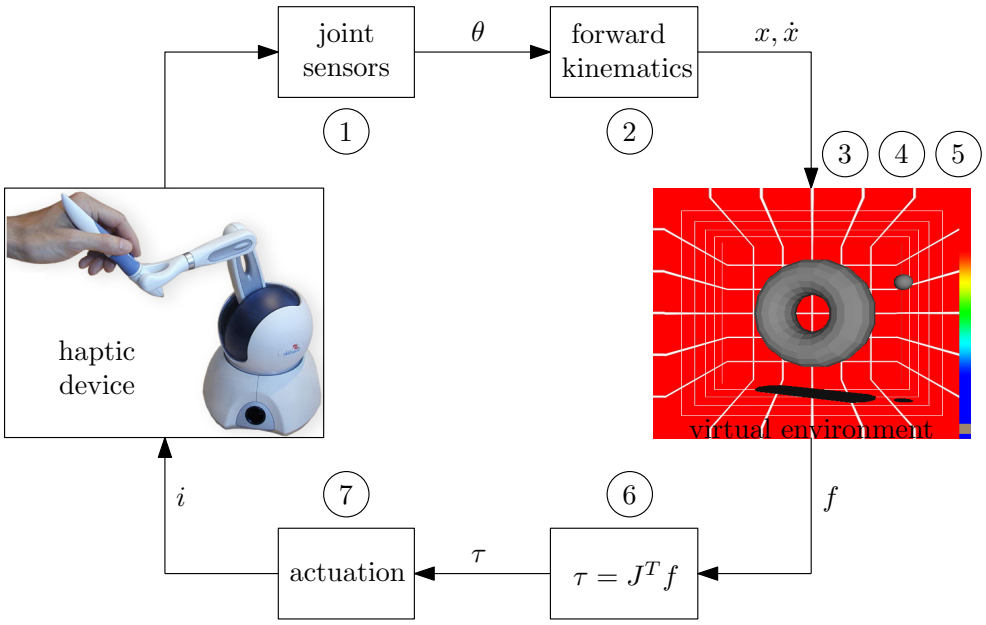


Figure 2.3: Schematic representation of the haptic rendering loop.

of 25 [μs] are needed. Haptic devices do not have this kind of bandwidth, and that such high fidelity is not usually needed. In fact, a common sampling rate in haptic simulation is 1 [kHz]. This sampling rate guarantees stability and realism. The haptic rendering cycle for an impedance-type haptic interface is represented in Fig. 2.3, and it has basically seven steps:

1. Position sensing: Rotary optical quadrature encoders are usually used as position sensors. They are often integrated inside rotary motors, that are used as actuators.

2. Forward kinematics: The measurements of position and velocity are typically represented in the joint space. However, these must be converted into the Cartesian space using the forward kinematics and the Jacobian matrix of the model.

3. Collision detection: In case of contact with virtual objects, the collision detection software detects collisions between a virtual object and the haptic interaction point (HIP). Actually, this interaction determines whether the HIP is penetrating or is inside the object surface. Object

surfaces are usually represented by geometric models such as polygons or splines. Hence, if the HIP is found to be outside all objects, then the force that is returned by the software is zero. The speed of computation of the collision detection software is paramount for haptic applications; however, the worst-case speed, as opposed to the average speed, is what count most in order to recreate a more realistic situation. Generally, solutions that evaluate collision in constant time are preferred.

4. Surface point determination: If the HIP is inside a surface, then the haptic force must be computed. Many researchers have used the idea of a virtual spring connecting the HIP to the nearest point on the surface, usually called intermediate haptic interaction point (IHIP), as a model of interpenetration and force generation (Zilles and Salisbury, 1995). However, all these researchers realized that the closest surface point is not always the most faithful model for representing the contact. Hence, considering the situation illustrated in Fig. 2.5, as the HIP moves laterally below the top surface of the cube, eventually it becomes close enough to the edge that the closest surface point becomes the side of the cube (at position P_4). In this situation, the algorithm needs a memory element in order to keep the IHIP on the top surface and generate an upward force at all times, otherwise the operator is suddenly ejected out the side of the cube.

5. Force response: Force is usually computed by using the spring model, as described in Eq. 2.1,

$$f = kx \tag{2.1}$$

where x is the vector from the HIP to the IHIP, and k the stiffness. When k is sufficiently large, the object surface becomes a virtual wall perpendicular to x, creating an impedance surface. This basic interaction is the basis of most haptic virtual environments. However, the pure stiffness model can be augmented to provide other effects, by using the virtual coupling. Damping effects can be added as well. Coulomb friction and/or other forms of nonlinearities may be displayed parallel to the surface too. In order to provide a more realistic representation of hard surfaces, vibrations can also be added in an open loop at the moment of collision. For example, by adding damping effects, the force rendering model becomes as described in Eq. 2.2, and illustrated in Fig. 2.4

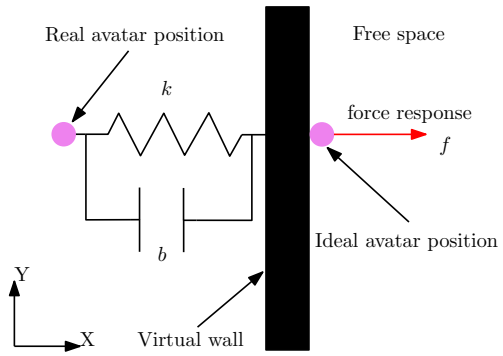


Figure 2.4: Interaction between the avatar and the virtual object surface, generating a reaction force f.

$$f = \begin{cases} 0 & \text{if } x > x_w \\ kx + b\dot{x} & \text{if } x \leq x_w \end{cases} \tag{2.2}$$

where x_w is the position of the virtual wall. Similarly, this interaction can be applied in a two-DOF interaction, three-DOF interaction or more.

6. Kinematics and Jacobian matrix: The force response calculated in the Cartesian space must then be reported into torques in the joint space. This can be done by the transpose of the haptic device Jacobian matrix, as shown in Eq. 2.3,

$$\tau = J^T f \tag{2.3}$$

where τ is the torque command to the actuators, f is the desired force vector, and J^T is the transpose Jacobian matrix of the haptic device.

7. Actuation: Current amplifiers are typically used to create a direct relationship between the voltage output by the computer via a digital-to-analog (D/A) converter and the torque output by the motor. The effect of actuator and amplifier dynamics and D/A resolution on system stability is typically negligible in comparison to position sensor resolution and sampling rate for most haptic devices. Actuator or amplifier saturation can produce undesirable behavior, particularly in multi-degree-of freedom haptic devices where a single saturated motor torque may change the apparent geometry of virtual objects. The force vector, and thus

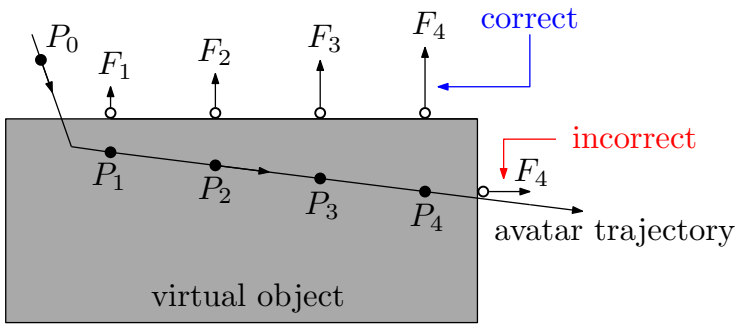


Figure 2.5: Haptic rendering computation of the contact force. HIP are shown at times $1-4$ (solid circles, $P_0 - P_4$). IHIP are shown when the HIP is inside the object (open circles).

the corresponding actuator torques, must be scaled appropriately if any actuator is saturated.

It is important to note that haptic rendering on dynamic objects carries out further complexity in terms of computational load. A real time haptic interaction with 3D objects that translate and rotate becomes computationally intensive as the objects collide among each other, as well as with the avatar. In this case, dynamic equations have to be solved (using a numerical integration method, such as Euler integration) and the state of each object has to be updated at every sampling time (see. Fig. 2.6). The methods that are used to calculate forces and torques, are based on the principles of rigid body dynamics (Baraff, 1994) and impulse mechanics (Mirtich and Canny, 1995) for the graphical simulation of floating multiple objects.

As the position and orientation of the virtual objects change at every sampling time, the updated coordinates (visual and haptics) of the virtual objects should be used to detect collisions with the new coordinates of the HIP. However, it would be computationally too expensive to update the object database at every sampling time in order to detect collisions with the new HIP (see Fig. 2.6b). A better strategy, in terms of computational load, is to compute everything relative to the original coordinates and apply the effects of transformation later, i.e. the HIP is multiplied by the inverse of the transformation matrix and the collisions are detected by the original haptic coordinates. Then, the reaction force is computed related to the original coordinates and then multiplied by the transformation matrix (see Fig. 2.6c). Therefore, only the visual coordinates are updated at every sampling time,

and the haptic coordinates of the object are not required to be updated at each iteration, when the object is transformed into a new state.

For static objects though, this complexity is more relaxed, since there are no dynamics to be calculated at every time step.

2.5 Control and stability in haptic interfaces

The haptic rendering cycle described in Fig. 2.3 is a closed-loop dynamical system. Rendering realistic contact forces, and maintaining a stable behavior of human-environment contact is still very challenging. The evidences of instability in haptic interfaces are buzzing, bouncing, or even diverging behavior. Empirically, instability is frequently encountered when developing haptic interfaces with virtual objects characterized by high stiffness, but this instability can be eliminated by reducing the stiffness of the virtual object or by the operator (by a firmer holding of the haptic device).

Although linear theory is very limited to treat this problem, it can be useful to compute a basic study by analyzing the factors affecting instability. A simplified model of an impedance device is shown in Fig. 2.7.

$G_1(s)$ and $G_2(s)$ represent dynamics of the haptic device for both operator position sensing and force display, respectively. Assume that the virtual environment (VE) and human operator (HO) can each be represented by a linear impedance such as:

$$Z_{VE} = \frac{F_{VE}}{X_{VE}} \tag{2.4}$$

$$Z_{HO} = \frac{F_{HO}}{X_{HO}} \tag{2.5}$$

Then the loop gain of the closed-loop system from the human operator and back again is:

$$G_l(s) = G_1(s)G_2(s)\frac{Z_{VE}}{Z_{HO}} \tag{2.6}$$

Stability in the classical sense is assessed by applying the magnitude and phase criteria of Nyquist to $G_l(s)$. Increasing Z_{VE} (corresponding to stiffer or heavier virtual objects) increases the magnitude of $G_l(s)$ and thus destabilizes the system while a firmer hold by the human operator, which increases the magnitude of Z_{HO}, has a stabilizing effect. Similar arguments apply to phase shifts that might be present in any part of the system.

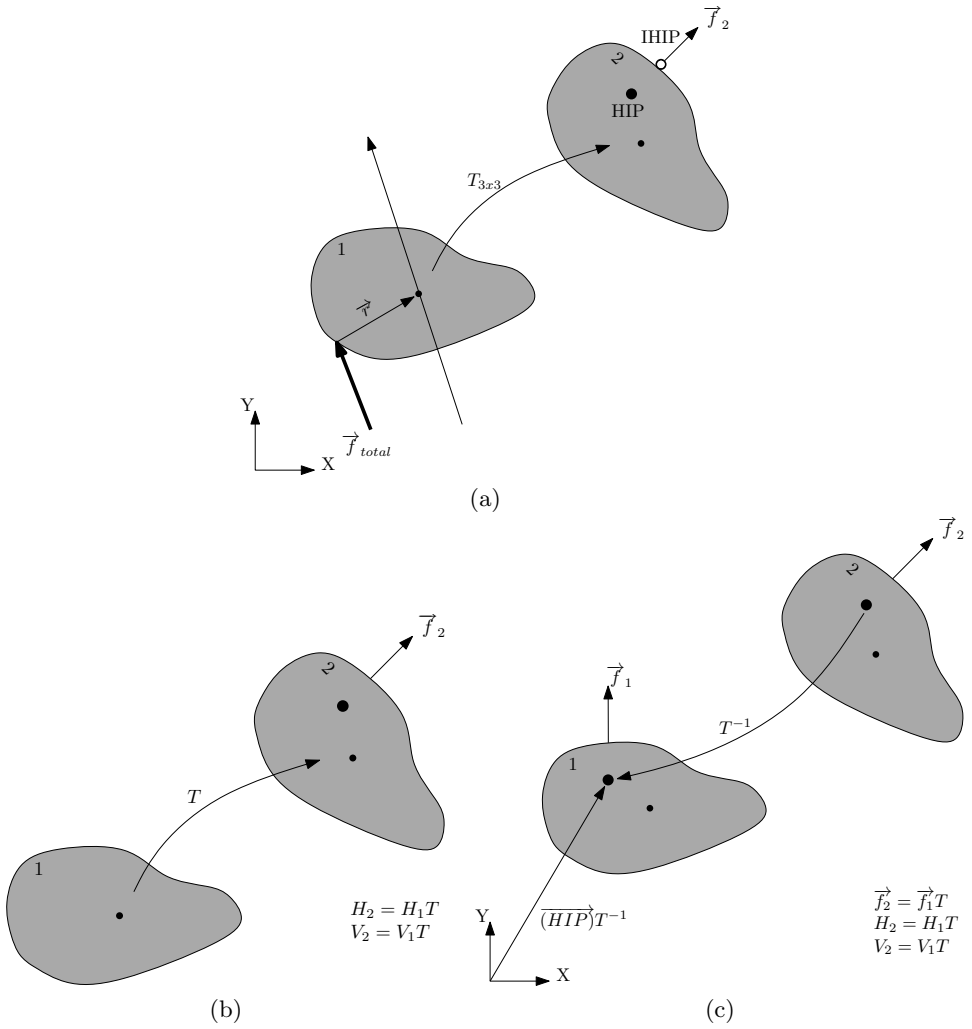


Figure 2.6: Point-based haptic interaction model by floating rigid object in virtual environments. (a) T_{3x3} represents the transformation matrix, f_2 represents the force at the fingertip. H and V represent the haptic and visual coordinates of the object, respectively. (b) Haptic and visual rendering are updated and collisions are detected using the new haptic coordinates. (c) Only visual coordinates are updated. Reaction force is computed related to the original coordinates using the transformation matrix T.

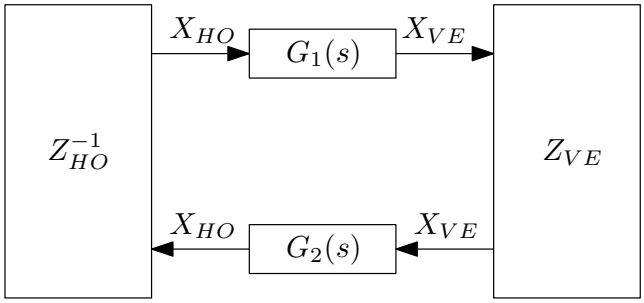


Figure 2.7: Simplified linear model of haptic rendering in an impedance device.

A second feature is the digital implementation, which introduces sampling and quantization, both of which have significant effects on stability. Analyzing a virtual wall of impedance, as represented in Eq. 2.7, (Colgate and Brown, 1994) analytically demonstrated an important relationship among the maximum stiffness a device can render, the level of mechanical damping of the device, the level of damping of the virtual wall, and the sampling time used for controlling the device.

$$H(z) = K + B\frac{z-1}{Tz} \tag{2.7}$$

Specifically, they derived the following condition for passivity of the device, as shown in Eq. 2.8,

$$b > \frac{KT}{2} + B \tag{2.8}$$

Therefore, in order to have a stable interaction, the damping of the haptic device b should always be higher than the sum of the level of damping of the virtual wall B and the product $\frac{KT}{2}$; where K is the stiffness to be rendered by the device and T the sampling period. Stiffer walls tend to become unstable for higher sampling periods, resulting in high-frequency vibrations and possibly elevate (and uncontrollable) forces. Increasing the level of mechanical damping featured by the device can limit instability, even though this limits the device's capabilities of simulating null impedance for free movement. Therefore, sampling time is a key issue in guaranteeing a stable haptic interaction.

[2]For further information see Blake Hannaford and Allison M. Okamura, *Haptics*, in Bruno Siciliano and Oussama Khatib, *Springer Handbook of Robotics*, Berlin, Heidelberg, 2008, pp. 719-739.

Chapter 3

Virtual reality based rehabilitation

For a post-stroke patient, the use of haptic based therapy highly contributes in regaining the mobility that was lost. Meanwhile, for operational therapists and doctors, this type of computer-based system is an efficient measurement tool, in which the most relevant parameters for finger and hand rehabilitation (e.g. position, velocity, acceleration and jerk) are accurately evaluated, determining the progresses achieved by the patient. As a result, the use of haptic interfaces has the potential to rehabilitate and help stroke survivors to relearn skills that were lost when part of the brain was damaged.

3.1 Overview

Virtual reality based rehabilitation is conceived as the interaction between a haptic device, which consist of a specific manipulator, and a virtual environment. Indeed, a haptic interface enables the patient to move and to interact with virtual objects inside a virtual space, hence a correspondence between the end-effecter of the haptic display and a virtual object (avatar) inside the virtual world is verified. This virtual object (avatar), interacts with other virtual objects (only static objects) under a visco-elastic behavior. As a VR-based application for rehabilitation, static objects are enough for representing the virtual environment. The interaction force depends on the "penetration" the avatar performs on the other virtual objects. Due to this interaction, the

post-stroke patient, receives a force feedback as a response. This allows the patient to interact with different types of objects, which may have different kind of properties.

The implementation of a haptic interface for rehabilitative applications often implies the design of custom devices, which requires the development of specific HW interfaces and kinematic and dynamic models. Clearly, this situation (besides of being very time consuming) carries out the problem of the lack of flexibility, which is highly desirable since patients may have different motor learning perspectives, and so they may require different types of exercises, and therefore devices for their rehabilitation.

In order to ease the design and testing of different devices, which may eventually have such differences in kinematics, sensing, and actuating systems, we implemented a general framework completely developed inside Matlab/Simulink, in such a way that any modification of the hardware is accomplished by simply modifications of the corresponding S-functions. In this way, we can guarantee a hardware independent solution for rehabilitative purposes.

We propose here two types of virtual reality haptic interfaces. One haptic device uses a five-bar linkage mechanism, while the other uses two linear ball slides, connected by two flexible transmissions to the linear motors for transmitting the linear motion.

3.2 Description of the general framework

Usually, the haptic interface consists of a computer (e.g. a PC), a virtual reality engine, a data acquisition board, motor drivers and a mechanism that constitutes the manipulated part of the haptic device. As shown in Fig. 3.1, the positions (usually joint angles) of the haptic device are acquired via data acquisition board and converted into real world coordinates of the end effector, using the forward kinematic model. Such coordinates are then passed to the VR engine, which is in charge of displaying the exercises on the computer screen and computing the interaction force between the avatar and the virtual objects. The computed force is then converted into force/torque references to the actuators, using the Jacobian matrix.

Each block of Fig. 3.1, except the haptic/graphic rendering of the virtual world, has an S-function associated with it. Therefore, from a constructive point of view, this application grants great flexibility, since any change in the mechanical or electrical hardware requires only a minor modification of the related S-functions, while the control algorithm remains the same. Indeed,

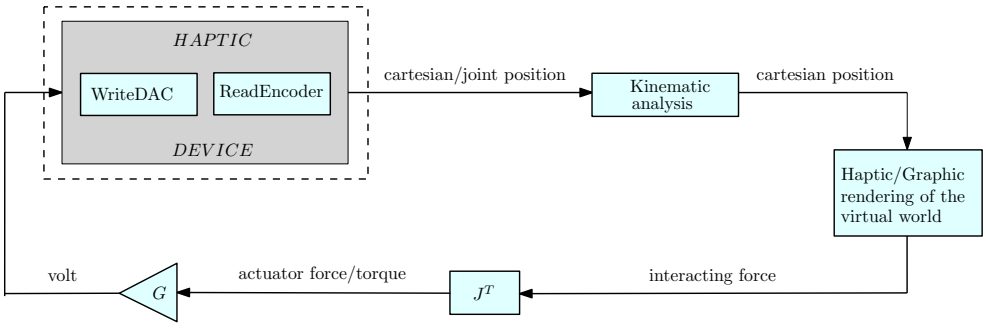


Figure 3.1: General framework architecture.

the only situations in which you need to implement some modifications are when the number of DOFs of the system are changed, or when the type of actuator (linear or rotating) is changed or when there is a modification in the type of sensors (cartesian or joint position). Therefore, the general framework approach is always the same, the only changes are inside the S-function kernels that are function of the mechanical and electrical hardware configuration.

The only invariant part of the general framework is the VR engine, which can be selected among many existing others. For our implementation, we chose Handshake proSENSE Toolbox for haptic and graphic rendering (Handshake VR Inc., 2006). In a nutshell, the Handshake proSENSE Toolbox adds haptic rendering to standard VRML based environments. As a MATLAB/Simulink based product, it works in a drag-and-drop fashion, thus allowing even untrained users (e.g. doctors and therapists) to quickly develop and test models, and to do on-the-fly modifications on both, virtual environment and exercises, depending on the requirements and performances of the patient.

Such general purpose framework guarantees a fast and easy implementation for different types of devices, which must handle the corresponding settings and configurations that depend on the type of therapies, which are personally oriented.

In the following, we describe the proposed architecture, which is represented by the block diagram in Fig. 3.1.

3.2.1 Data acquisition board interface

All data acquisition cards come along with a software development kit (SDK), which includes several source-code modules written in C/C++, as well as dy-

namic link libraries (DLL), import libraries (LIB) and utility files. These set of libraries and executable software modules are designed to help the development of application programs. In standard programming, the application SDK can be used to implement a real-time interface to the data acquisition board within Matlab/Simulink, using the S-function (user-defined function) (The MathWorks Inc., 2005). An S-function describes a dynamical system, in which the characteristic code can be written in C language.

A specific S-function must be created in order to achieve a real-time communication with the acquisition board, reading the signal from the encoders and writing the command signal into the DAC channels. The S-function programming procedure has the same constructive method for any data acquisition board.

3.2.2 Kinematic model and transposed Jacobian

When a haptic display uses rotating motors, the reaction force must be converted into a torque command that is transmitted to the motors. This transformation is possible using the transpose jacobian matrix of the mechanism that defines the haptic device. The kinematic analysis and the transpose jacobian matrix can also be implemented inside an S-function. Then, the procedure to create an S-function that implements the forward kinematic model or the transposed Jacobian is much more simpler than the one used for the I/O board communication.

3.2.3 Virtual world

In the proposed architecture, the Handshake proSENSE Toolbox, which is a Matlab/Simulink based product, was chosen for the graphic and haptic rendering of the virtual world. This toolbox adds several haptic properties, such as stiffness, damping and friction, into standard VRML files, which represents the developed virtual environment. In fact, once a VRML file is loaded, the user can specify and select fields as input ports.

3.3 Five-bar linkage haptic interface

We developed a haptic interface for finger/hand rehabilitation, integrated in a general purpose framework, which provides haptic and graphic rendering, and in which it is also implemented the control algorithm of the five-bar linkage

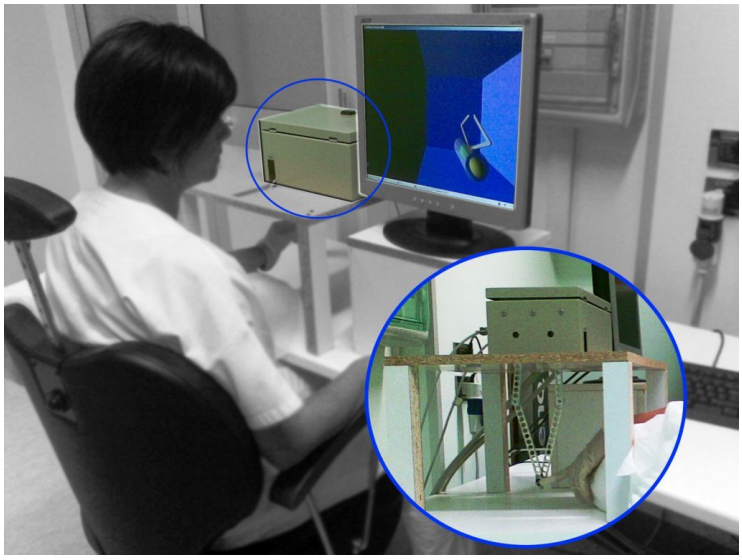


Figure 3.2: Haptic interface setup. The setup considers a PC with a control algorithm, a VR environment with graphic and haptic rendering, and a five-bar linkage mechanism with finger holder.

mechanism, as well as the external communication with the actuators; these features guarantee a real-time system.

The haptic interface consists of a PC running Microsoft Operating System (Windows XP), with a data acquisition card (Sensoray Model 626 PCI Multifunction I/O Board) that provides the interface to a five-bar linkage haptic device, moved by two AC brushless servo motors (Mavilor Motors Model BLS-74) with the relative motor drivers (Infranor CD1-a). The setup and the prototype are shown in Fig. 3.2. The software for controlling the haptic interface was completely developed within Matlab/Simulink as a real-time application. The external communication to the acquisition board was implemented as an S-function, as well as the kinematic analysis for solving the mechanism, and the calculation of the transpose jacobian matrix. The visual and haptic feedback were implemented using the Handshake proSENSE Toolbox (for details, see Appendix A).

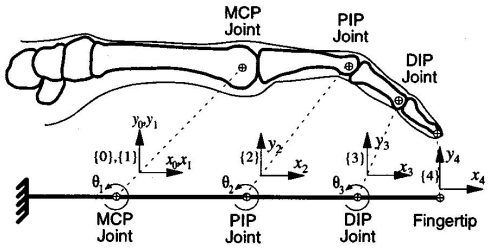


Figure 3.3: Planar skeletal model of the human finger.

Finger Joint	Angular Motion Range (Degrees)	Link Sizes (mm)
MCP	$-90 \leq \theta_1 \leq 45$	48.3
PIP	$-120 \leq \theta_2 \leq 0$	28.2
DIP	$-90 \leq \theta_3 \leq 50$	19.1

Table 3.1: Finger joint motion ranges.

3.3.1 Biomechanical considerations for the design

The considerations for the design of the haptic device are focused on the implementation of a single finger haptic display, in which the force is exerted at the fingertip. Hence, we need to take into account all possible movements of each articulation that belongs to the finger, and of course, the reachable workspace of the finger itself. Consequently, this workspace must fit inside the workspace of the haptic interface.

The design is based on a male index finger (see Fig. 3.3). We have considered the average dimensions of the finger and a set of admissible movements (Table 3.1). This information was derived from literature and based on statistical data (Venema and Hannaford, 2001).

Under normal conditions, which means that the finger's motion is without obstacles and/or any load applied on it, the reachable workspace of a single finger is represented by Fig. 3.4. To cover such workspace, we have designed a five-bar linkage mechanism.

3.3.2 Design of the haptic device

A haptic device, suitable for finger/hand exercise rehabilitation was developed. The chosen mechanism was a five-bar linkage, in which one bar is fixed to the frame, and the two cranks, fixed to the frame, are considered the moving

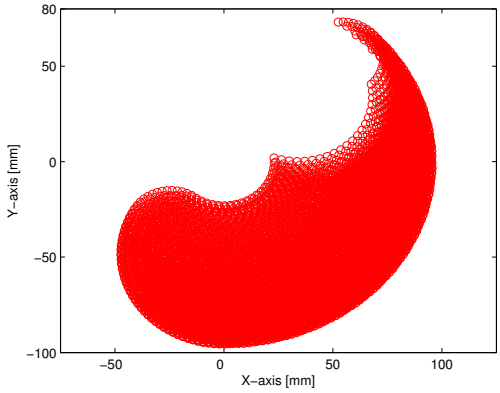


Figure 3.4: Reachable workspace of an average male's index finger.

members, as shown in Fig. 3.5. The proposed mechanism has two active DOF and three passive DOF (rotations) at the fingertip. In this particular case, joint P_1 and joint P_4 are driven by AC brushless motors, in which the maximum torque available is 3.4 [N-m], that corresponds to a force of 20 [N] at the fingertip. Each motor has an incremental encoder with a resolution of 32768 pulses per revolution, for which the manipulator position resolution is 0.022 [mm].

The design of the proposed mechanism was based on these three factors:

- The haptic device workspace must cover the whole reachable workspace of an average male's index finger.
- Low inertia.
- High performance.

The performance analysis is based on the evaluation of the mechanism isotropy (ISO), defined as follows:

$$\mu = \frac{\sigma_{min}(J(\boldsymbol{\theta}))}{\sigma_{max}(J(\boldsymbol{\theta}))} \tag{3.1}$$

where σ_{min} and σ_{max} are the minimum and maximum singular value decomposition values of the jacobian matrix $J(\boldsymbol{\theta})$ that describes the five-bar linkage mechanism, respectively. The mechanism isotropy is a function of the joint angles $\boldsymbol{\theta}$ and the value goes from 0 to 1. An ISO value of 0 means the mechanism is in a singular configuration. A typical singular configuration of the five-bar

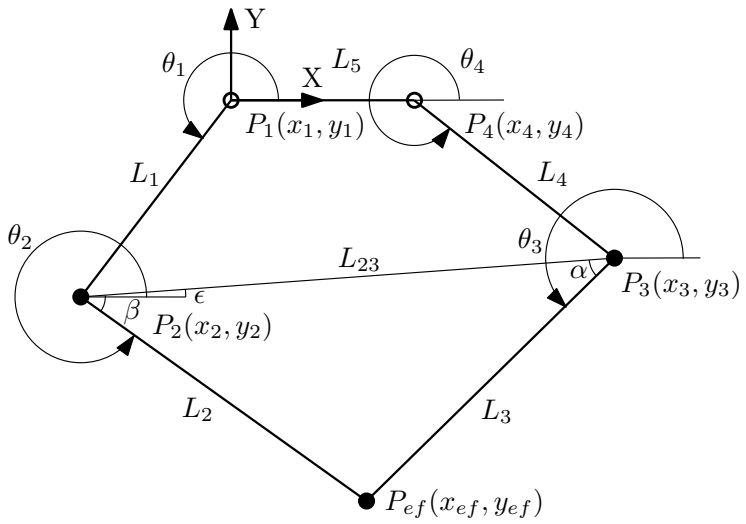


Figure 3.5: Five bar linkage mechanism.

mechanism is obtained when the links are fully stretched. An ISO value of 1 means maximum performance, which also means that the mechanism can move equally well in all directions.

To achieve these requirements we have implemented an optimization procedure, using a non-linear programming algorithm, in which the link lengths were determined by maximizing the mechanism isotropy and by finding the best fitting between the average male's index finger workspace and the haptic device workspace.

In fact, the optimization algorithm searches the best solution in order to fit the workspace of Fig. 3.4 inside the reachable workspace of the mechanism, as shown in Fig. 3.6, maintaining a high level of ISO. These results are reported in Table 3.2. Note that ISO is always high (0.4 minimum) in the whole workspace.

3.3.2.1 Kinematic model and jacobian matrix

Forward kinematic analysis has to be done, in order to determine the end-effector position of the mechanism (Spong and Vidyasagar, 1989). The angular positions, θ_1 and θ_4, are known parameters because they can be read from the encoders that come with the AC brushless motors. The end-effector position of the haptic device is determined with respect to the origin of the XY axis,

Joint	Length [mm]
L_1	140
L_2	180
L_3	180
L_4	140
L_5	70

Table 3.2: Link lengths of the optimized mechanism.

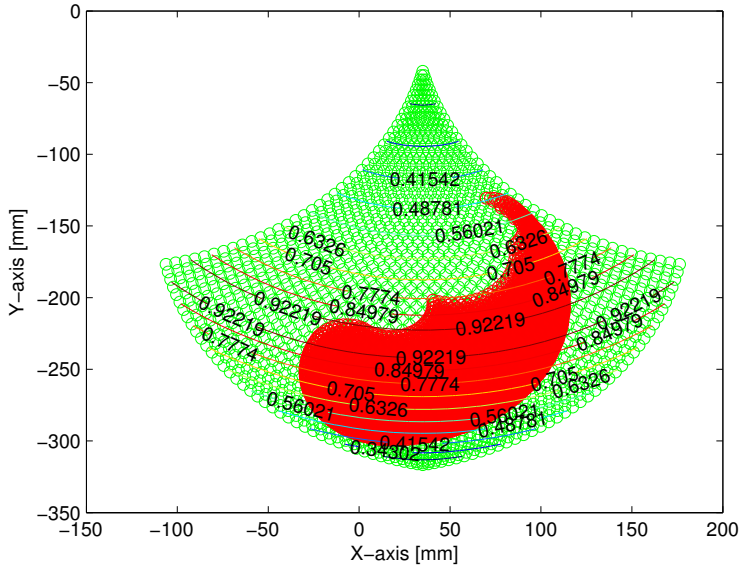


Figure 3.6: Reachable finger workspace, five bar linkage workspace and ISO values.

that is placed in P_1. Forward kinematics, as well as force analysis are based on Fig. 3.5, where L_1, L_2, L_3, L_4, L_5 and θ_1, θ_2, θ_3, θ_4 represent the link lengths of the manipulator and the joint angles, respectively. The cartesian position $P(x_{ef}, y_{ef})$ of the end-effector of the device,

$$x_{ef} = L_1 \cdot \cos(\theta_1) + L_2 \cdot \cos(\theta_2) \tag{3.2}$$

$$y_{ef} = L_1 \cdot \sin(\theta_1) + L_2 \cdot \sin(\theta_2) \tag{3.3}$$

The jacobian matrix of the mechanism is defined:

$$J(\boldsymbol{\theta}) = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \tag{3.4}$$

Notice that $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3, \theta_4]^T$.

$$a_{11} = -\frac{1}{2} \cdot \frac{L_1 \cdot (\cos(-\theta_2 + \theta_1 - \theta_3) - \cos(-\theta_2 + \theta_1 + \theta_3))}{\sin(\theta_2 - \theta_3)} \tag{3.5}$$

$$a_{12} = \frac{1}{2} \cdot \frac{L_4 \cdot (\cos(\theta_2 + \theta_3 - \theta_4) - \cos(\theta_2 - \theta_3 + \theta_4))}{\sin(\theta_2 - \theta_3)} \tag{3.6}$$

$$a_{21} = \frac{1}{2} \cdot \frac{L_1 \cdot (\sin(-\theta_2 + \theta_1 - \theta_3) + \sin(-\theta_2 + \theta_1 + \theta_3))}{\sin(\theta_2 - \theta_3)} \tag{3.7}$$

$$a_{22} = -\frac{1}{2} \cdot \frac{L_4 \cdot (\sin(\theta_2 - \theta_3 + \theta_4) - \sin(\theta_2 + \theta_3 - \theta_4))}{\sin(\theta_2 - \theta_3)} \tag{3.8}$$

Once the mechanism is defined, it is necessary to obtain a relationship that links the applied force at the fingertip and the equivalent torque of the electric motors. The relationship is expressed through the transpose Jacobian of the mechanism:

$$\tau_F = J^T(\boldsymbol{\theta}) \cdot F \tag{3.9}$$

where F represents the generalized forces exerted on the end-effector, and τ_F represents the torques exerted by the actuators in the joints.

If we consider that the mechanism is going to be used in a vertical plane, so it is necessary to determine the equivalent torques in order to compensate the effects of gravity. Therefore, if we take into account both, the generated forces at the fingertip, and the forces due to the weight, we have that the generated torques transmitted to the motors can be expressed as:

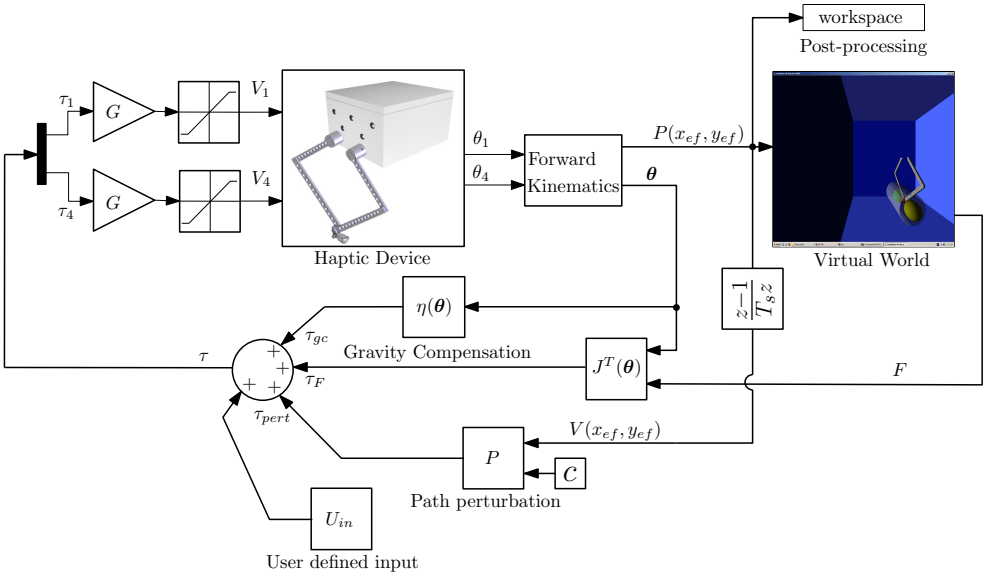


Figure 3.7: Control diagram of the haptic device and the virtual environment.

$$\tau = J^T(\boldsymbol{\theta}) \cdot F + \eta(\boldsymbol{\theta}) \tag{3.10}$$

where $\eta(\boldsymbol{\theta})$ represents the gravity compensation.

3.3.2.2 Control diagram for the haptic interface

An open loop impedance control was used for the proposed five-bar linkage mechanism. The input in the physical model of the virtual world is the end-effector position of the device, expressed as a cartesian position, and the output is the reaction force. The control scheme is described in Fig. 3.7.

As the position of the end-effector is known, the physical model algorithm detects a collision with a virtual object. Depending on the properties of the object, the algorithm determines the reaction force. The calculated force is then converted to motor torque through the transpose Jacobian matrix (Eq. 3.9) and then the contribution of the gravity compensation is added (Eq. 3.10).

The torque gain G is used to convert the signal into volt. A saturation block bounds it, preventing higher exertion forces, and avoiding any possible injury.

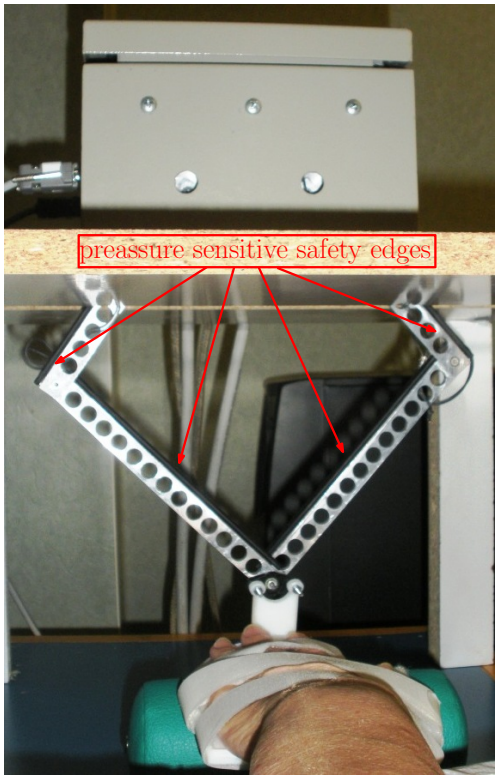


Figure 3.8: Pressure sensitive safety edges.

3.3.2.3 Safety features

Saturation blocks can be used for safety, but only as a software solution. This, however, not always guarantee safety. Hence, a hardware solution is needed in order to achieve safety. That's why we have implemented this haptic device with a pressure sensitive safety edges along the moving bars of the mechanism (see Fig. 3.8). Using this hardware solution we guarantee the safety of the patient when unexpected higher forces occur, and/or when the mechanism may have unexpected behaviors.

3.3.3 Rehabilitation program

This rehabilitation program tries to evaluate the motion performance in two patients, each one of them with different pathologies; one with a stroke and the other one with a musculoskeletal problem. The kinematic evaluation was

performed by using the same haptic device, while the clinical evaluation was conducted by using the following functional assessment scales; Fugl-Meyer (Fugl-Meyer et al., 1975), Box and Block Test (BBT) (Mathiowetz et al., 1985a), and Jebsen-Taylor Test (Jebsen et al., 1969) for the post-stroke patient, while for the patient with musculoskeletal disorders, the Fugl-Meyer and the Nine Hole Peg Test (NHPT) (Mathiowetz et al., 1985b). With this rehabilitation program we have evaluated the motion performance of two patients, one with a stroke and the other with a musculoskeletal problem. Hence, we propose two different exercises, applied on each patient, by using the same haptic device.

The main scope of the device described above is to rehabilitate the functionality of the finger (see Fig. 3.10a). However, this can be adapted into a device for hand rehabilitation, since for a post-stroke patient it is more suitable to rehabilitate first the hand and then the fingers. Then, for hand rehabilitation, instead of using a finger holder as an end-effector, we have replaced it by a hand holder, attached to the end-effector of the five-bar linkage mechanism by a spherical passive joint (see Fig. 3.10b). The latter allows the three rotations around a pivotal point at the end effector and, in turn, the movements of the patient's hand (see Fig. 3.11). As a result, the haptic device with finger holder was used by the patient with a musculoskeletal problem, while the patient with stroke used the same haptic device but with hand holder.

During this rehabilitation program, the two patients were asked to perform two tasks. In the first one, these two patients were asked to compute a reaching operation between two spheres, and in the second one, to follow a 8-shaped predefined trajectory. All these two tasks were carried out inside a certain tolerance, in vertical and horizontal arrangement. For the reaching operation the chosen tolerance was equal to 0.03 [m], and for the 8-shaped predefined trajectory was equal to 0.014 [m]. These patients were under virtual reality based therapy for twenty sessions during three weeks.

The virtual environment used in this rehabilitation program was developed in V-Realm Builder 2.0. The VRML file that represents the virtual world is loaded in the HVR World block (Handshake proSENSE Toolbox). This function provides haptic and graphic rendering to the model. A spring-damper model was used to describe the haptic properties of the virtual objects in the virtual environment. The yellow and green sphere, shown in Fig. 3.12, have haptic properties, as well as the semi-transparent gray tube.

Perturbing the patient's path makes this kind of exercise effective (Shad-

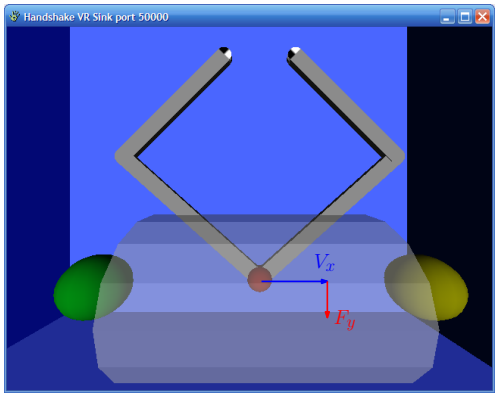


Figure 3.9: Orthogonal perturbation on the patient's path.

mehr and Mussa-ivaldi, 1994). In fact, this perturbation is performed as a force, proportional to the end-effector velocity exerted by the patient. This perturbation is applied orthogonally to the patient's movement, affecting their actual path (see Fig. 3.9).

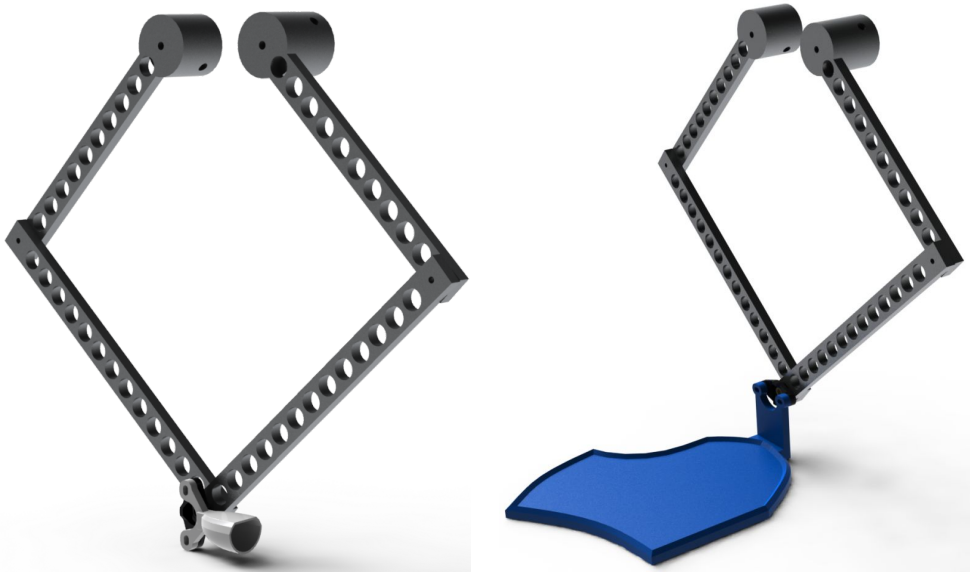
In order to test the motion characteristics and evaluate the progresses of the two patients, we have evaluated the position error and the jerk. Actually, jerk is considered a useful data, as it measures the patient's strain (Rohrer et al., 2002). We report the pre-training and the post-training results.

3.3.4 Kinematic and clinical evaluation

The kinematic evaluation was based on the percent variation of the position error and jerk, while the clinical evaluation was based on functional assessment scales, such as Fugl-Meyer, BBT and Jebsen-Taylor test.

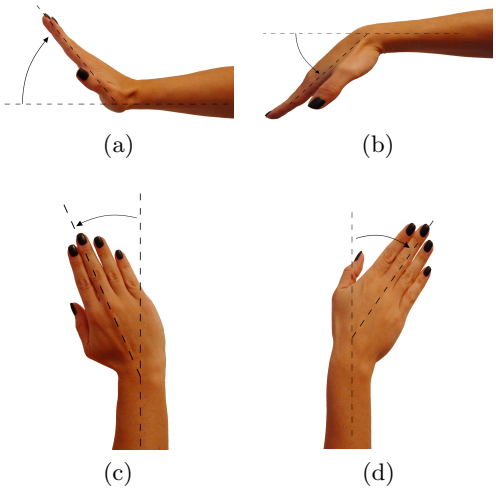
3.3.4.1 Patient with musculoskeletal disease

We report here the kinematic results of the reaching operation and the 8-shaped predefined trajectory in horizontal arrangement. However, these exercises were also computed in vertical and sagittal arrangement. We report in Table 3.5 the clinical results after the complete VR-based rehabilitation program. The values of jerk reported in Table 3.3 and Table 3.4, for pre and post training, are taken by computing the mean value between the root mean square value of jerk in X-axis and Y-axis, respectively.



(a) (b)

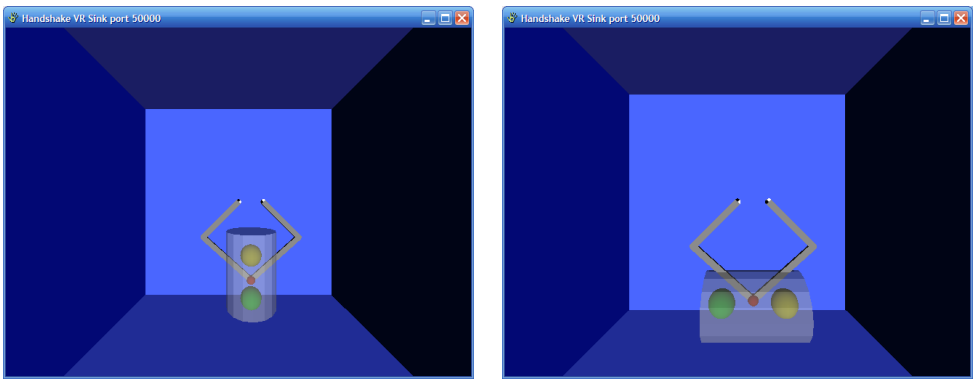
Figure 3.10: Haptic device for finger/hand rehabilitation. (a) Finger rehabilitation. (b) Hand rehabilitation.



(a) (b)

(c) (d)

Figure 3.11: Possible movements for hand rehabilitation: (a) Extension. (b) Flexion. (c) Radial deviation. (d) Ulnar deviation.



(a) (b)

Figure 3.12: Virtual environment for the two proposed exercises: (a) Vertical arrangement. (b) Horizontal arrangement.

Evaluating parameter	Pre	Post	Percent Variation [%]
Position Error - RMS $[m]$	0.0124	0.004	-67.74
Jerk - RMS $[m/s^3]$	0.137	0.132	-3.65

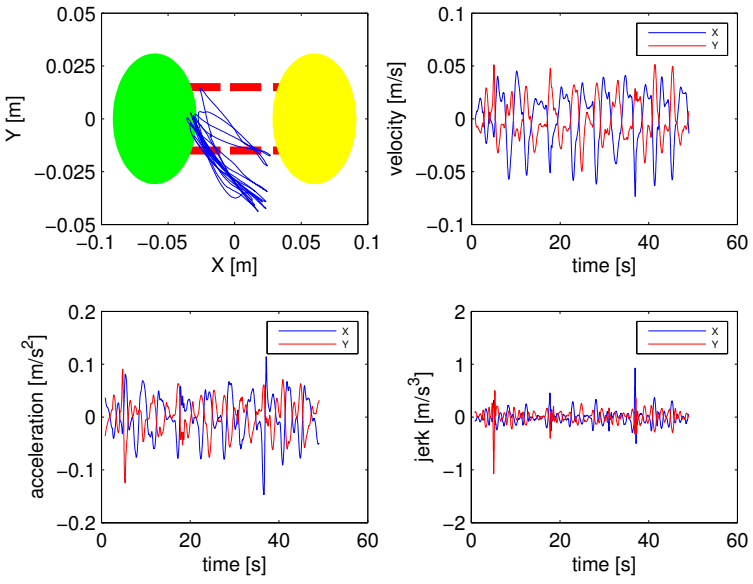
Table 3.3: Reaching operation results in horizontal arrangement.

Evaluating parameter	Pre	Post	Percent Variation [%]
Position Error - RMS $[m]$	0.0027	0.0015	-44.44
Jerk - RMS $[m/s^3]$	0.1013	0.0865	-14.61

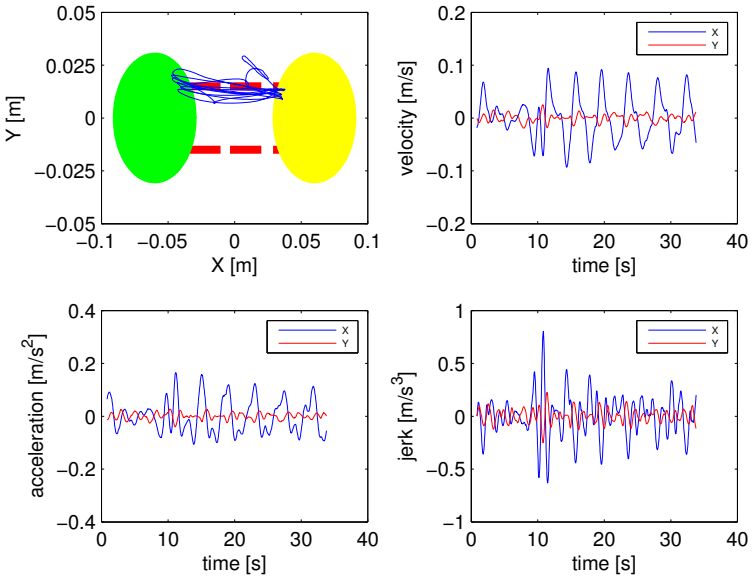
Table 3.4: 8-shaped predefined trajectory results in horizontal arrangement.

Functional Scale	Initial Evaluation	Final Evaluation
Fugl-Meyer	56/66	56/66
Nine hole peg test	18 [s]	18 [s]

Table 3.5: Clinical evaluation of the patient with musculoskeletal disease.

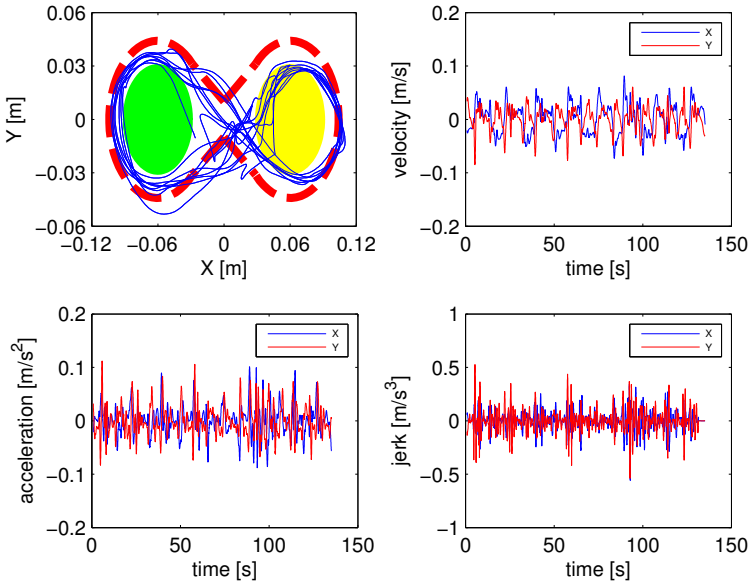


(a)

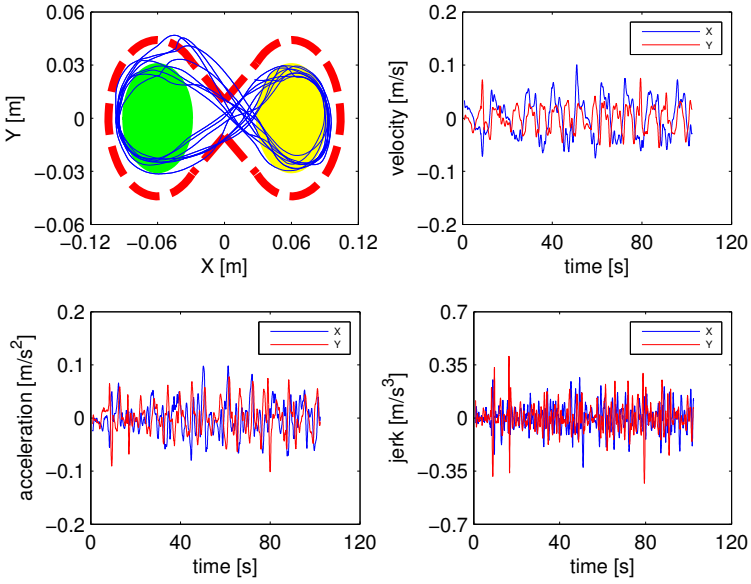


(b)

Figure 3.13: Kinematic results for reaching operation with two spheres in XY plane in horizontal arrangement: (a) Pre-training. (b) Post-training.



(a)



(b)

Figure 3.14: Kinematic results for 8-shaped predefined trajectory along two spheres in XY plane in horizontal arrangement: (a) Pre-training. (b) Post-training.

Evaluating parameter	Pre	Post	Percent Variation [%]
Position Error - RMS $[m]$	6.7e-4	1.02e-5	-98.48
Jerk - RMS $[m/s^3]$	0.0034	0.0024	-29.41

Table 3.6: Reaching operation results in vertical arrangement.

Evaluating parameter	Pre	Post	Percent Variation [%]
Position Error - RMS $[m]$	0.0034	0.0005	-85.29
Jerk - RMS $[m/s^3]$	0.0066	0.0052	-21.21

Table 3.7: 8-shaped predefined trajectory results in vertical arrangement.

3.3.4.2 Patient with stroke

The subject under VR-based therapy is a sub-acute patient. We report here the results of the reaching operation and the 8-shaped predefined trajectory in vertical arrangement. These exercises were also executed in horizontal and sagittal arrangement. We report in Table 3.8 the clinical results after the rehabilitation sessions.

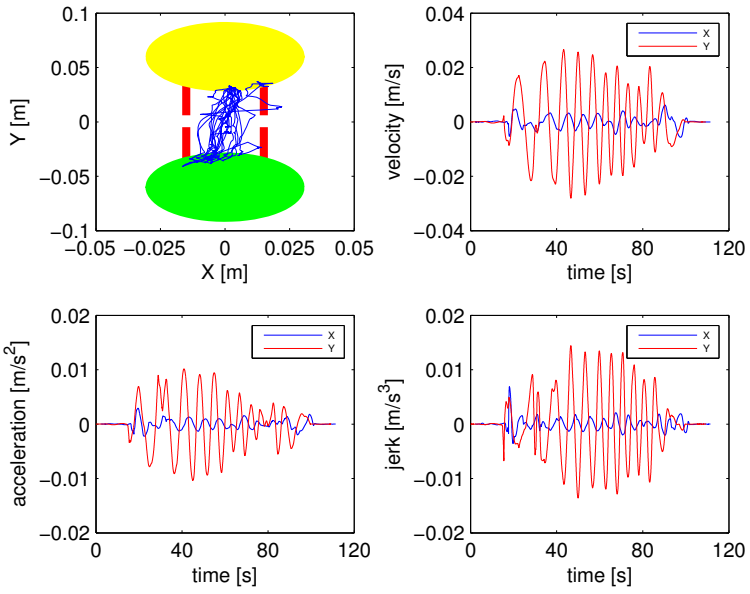
The values of jerk reported in Table 3.6 and Table 3.7, for pre and post training, are taken by computing the mean value between the root mean square value of jerk in X-axis and Y-axis, respectively.

3.3.5 Discussion of the results

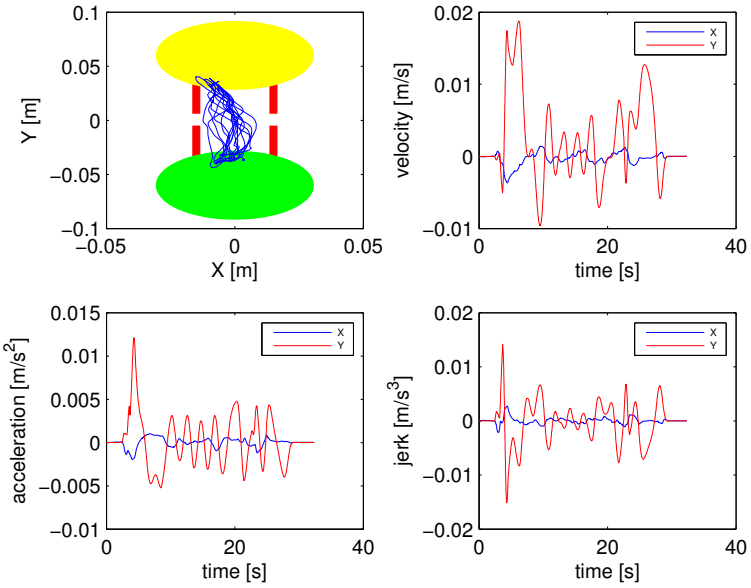
In order to quantify the improvement of the kinematic functionalities of each single patient, we have evaluated their motor behavior by comparing the pre and post training results, based on the percent variation of the position error and jerk. For the patient with musculoskeletal problem, the position error was reduced by 67.74 % and jerk by 3.65 %, during the reaching operation

Functional Scale	Initial Evaluation	Final Evaluation
Fugl-Meyer	40/66	53/66
Box and Block Test	20 [cubes/min]	30 [cubes/min]
Jebsen-Taylor Test	180 [s]	120 [s]

Table 3.8: Clinical evaluation of the sub-acute patient.

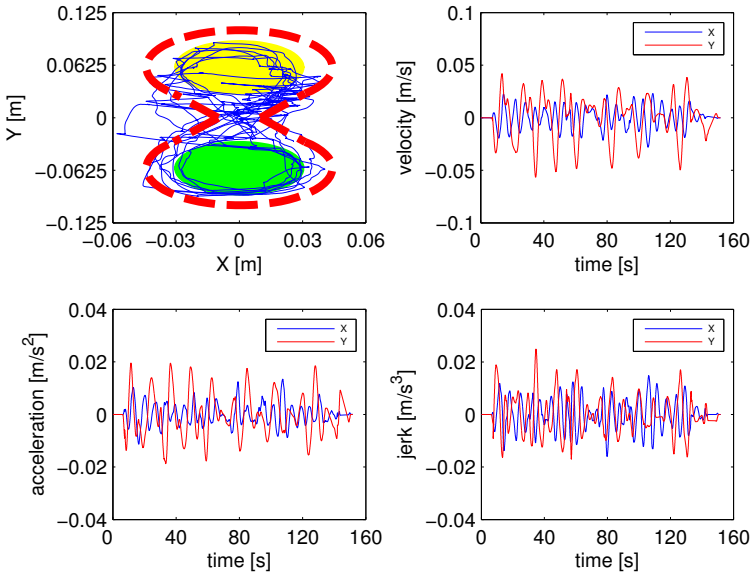


(a)

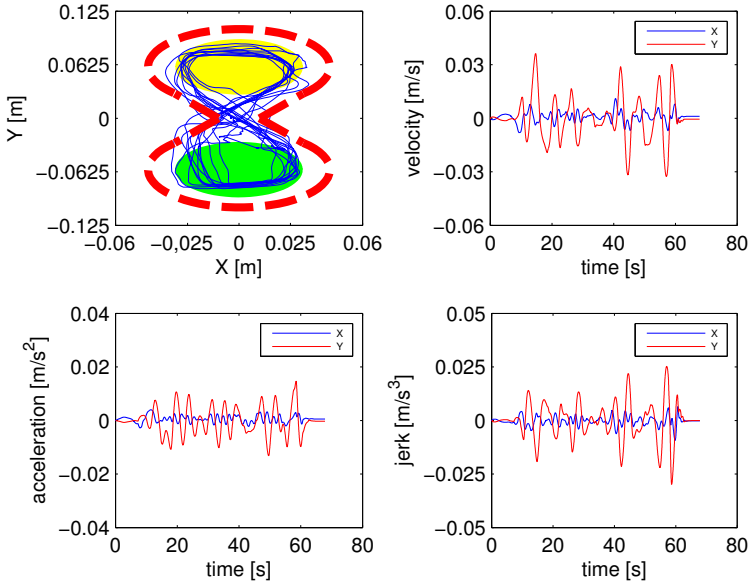


(b)

Figure 3.15: Reaching operation with two spheres in XY plane in vertical arrangement: (a) Pre-training. (b) Post-training.



(a)



(b)

Figure 3.16: 8-shaped predefined trajectory along two spheres in XY plane in vertical arrangement: (a) Pre-training. (b) Post-training.

exercise, while in the 8-shaped predefined trajectory exercise, position error was reduced by 44.44 %, and jerk by 14.61 %. For the patient with stroke, the position error was reduced by 98.48 % and jerk by 29.41 %, during the reaching operation exercise, while in the 8-shaped predefined trajectory exercise, position error was reduced by 85.29 %, and jerk by 21.21 %. According to these results, using a haptic interface grants a relevant improvement on the patient's finger and hand motion functionality. All the percent variation values were negative, which means that the precision error, as well as the jerk, were reduced. This means that the motor behavior of the patient was modified, which deals with a better motor performance.

Based on clinical evaluation results, we can say that the sub-acute patient has improved their condition after the VR-based therapy. In fact, according to the Box and Block test he has improved by 50 %. In the Jebsen-Taylor functional test, as the evaluation deals with the time the patient takes to compute a certain task, it is expected that the execution time at the final evaluation is smaller. As expected, he has improved by 33 %. According to the Fugl-Meyer scale, despite of having an improvement, he still remains with mild impairment.

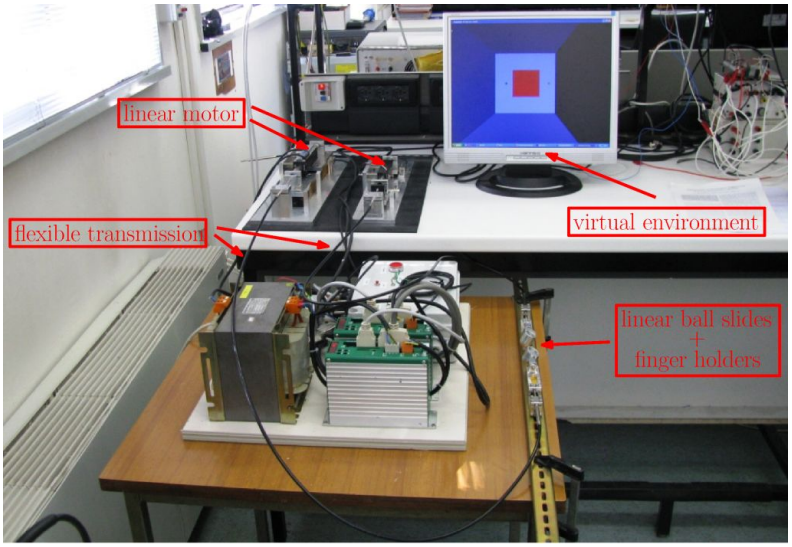
According to the clinical evaluation on the patient with musculoskeletal disease, we can say that there is no improvement, but neither a diminished condition. Anyhow, this was expected as these functional scales and tests are applicable in cases where there are neurological disorders. However, according to the Fugl-Meyer scale, he remains in mild impairment condition.

3.4 Haptic interface for grasping

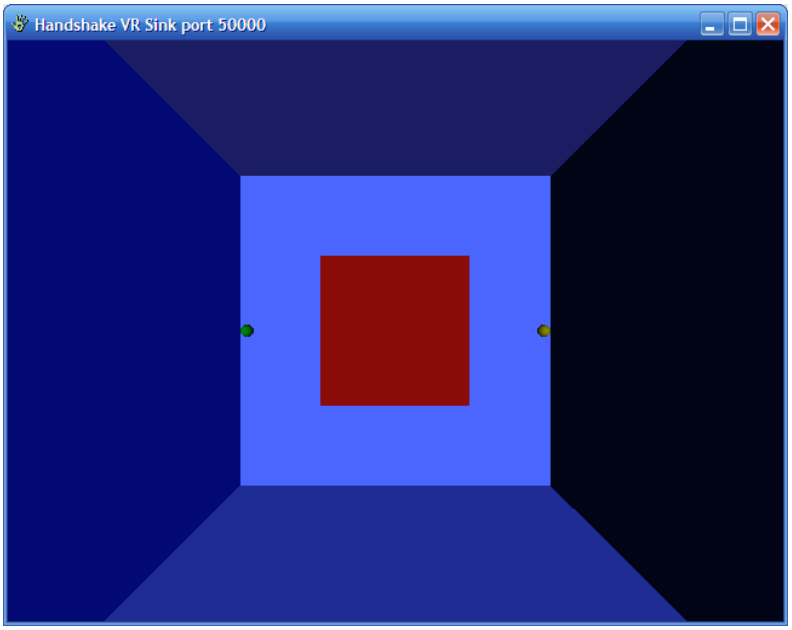
We developed a haptic interface for finger rehabilitation, integrated in the proposed general purpose framework, which provides haptic and graphic rendering, and in which is also implemented the control algorithm of the two linear motor, as well as the external communication, for which a real-time system in achieved. The haptic interface consists of a PC running Microsoft Operating System (Windows XP), with a data acquisition card (Sensoray Model 626 PCI Multifunction I/O Board) that provides the interface to a two linear motor haptic device, actuated by two direct drive linear servo motors (S120Q and S160Q) with the relative motor drivers (Servoland Movo). These motors are, in turn, connected by a flexible transmission with two linear ball slides, in which suitable finger-holders are placed on it.

The software for controlling the haptic interface was completely developed

inside Matlab/Simulink. The external communication to the acquisition board was implemented as an S-function. The position is read directly from the linear encoder. The visual and haptic feedback were implemented by Handshake proSENSE Toolbox. As such rendering must be available for both fingertips, the dual configuration must be enabled. Actually, haptic feedback must be provided for both fingertips (see Fig. 3.17b). As the forces are not scaled, thus sent directly to the actuators, there is no need of using the transposed Jacobian. All these features guarantee a real-time performance of the system. The setup of the prototype is shown in Fig. 3.17. More details of the set-up are shown in Fig. 3.18.

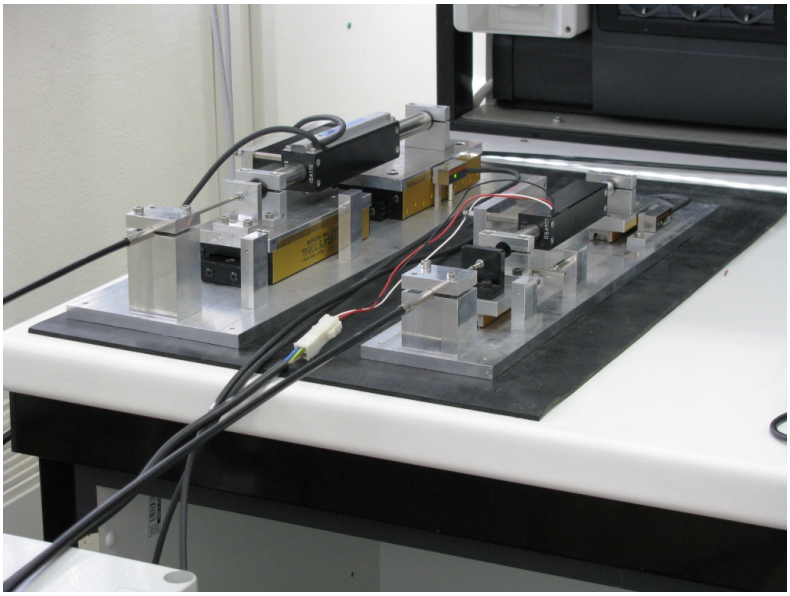


(a)

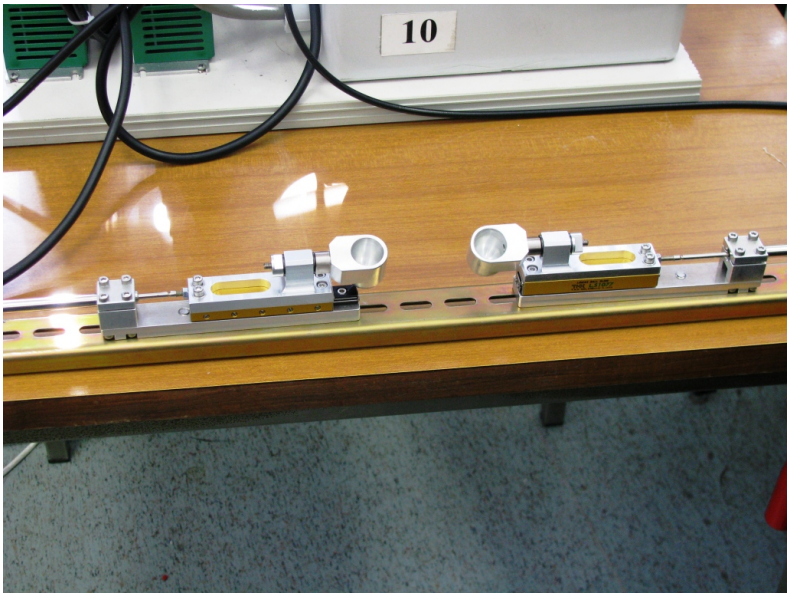


(b)

Figure 3.17: Haptic interface set up. (a) PC with a VR engine, two linear motor connected with two linear sliders by thrust wires. (b) Representation of both fingertips inside the virtual environment.



(a)



(b)

Figure 3.18: Detailed view. (a) Linear motors with linear encoders and flexible transmissions. (b) Finger holders and linear ball slides with the other end of the flexible transmission.

Chapter 4

Teleoperation

The prefix tele comes from Greek and means at a distance, so teleoperation indicates "operating at a distance". Hence, a bilateral teleoperation system enables human interaction with environments that are inaccessible to direct human contact due to their remoteness or the presence of hazards. Haptic feedback improves task performance during teleoperation applications.

4.1 Overview

The purpose of teleoperation is to extend the human capability of manipulating objects to a remote place, providing to the operator the same conditions as those at the remote location, as illustrated in Fig. 4.1. Generally, the human operator imposes a force on the master device which is traduced in a displacement that is transmitted to the slave device that performs that movement. As the slave performs the task, and forces are measured, then the reflecting forces are sent back to the master device.

The main goals in teleoperated systems are stability and transparency. Stability is achieve by maintaining always stable the closed-loop system no matter the behavior of human operator or the environment. Transparency is the ability of a teleoperation system to present the undistorted dynamics of the remote environment to the human operator (Hannaford, 1989a). This ability is affected by the closed-loop system of the master and slave devices, which distort the dynamics of the remote environment perceived by the human operator (Lawrence, 1993; Yokokohji and Yoshikawa, 1994). Hence, these two

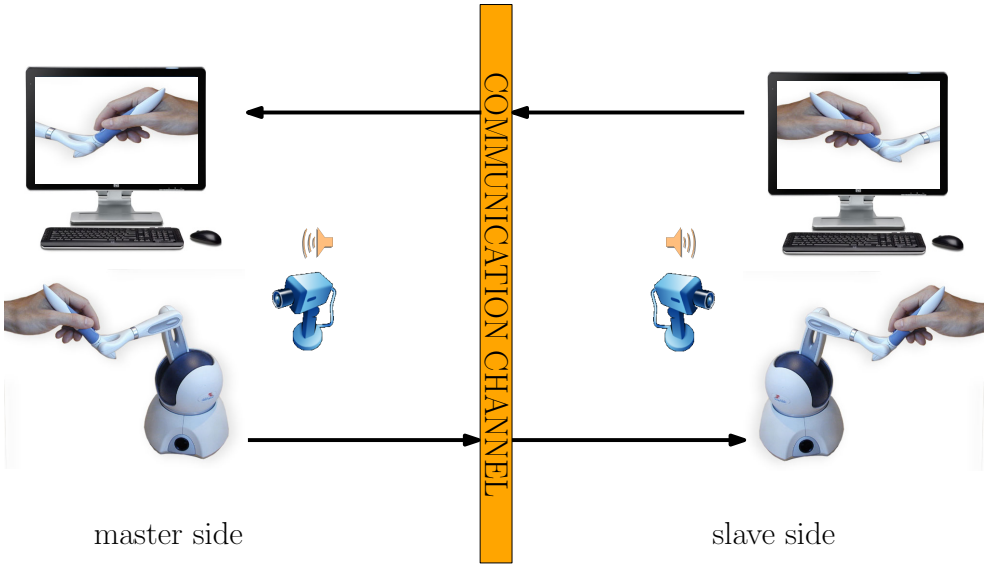


Figure 4.1: Overview of a bilateral teleoperation system.

tasks are generally conflicting.

Several complications arise because of the communication line in teleoperated systems: distortion, delays, and losses that impact stability and performance.

4.2 Stability and performance

In order to increase performance and achieve transparency, many master/slave systems incorporate force feedback. Bilateral interaction provides both forward and feedback information from the operator to the environment and viceversa. The bilateral nature of this setup makes the control architecture particularly challenging: multiple forms of feedback loops even without environment contact or user intervention may conform internal closed loops. The communication line between the two sites are often affected by delays, that destabilize the system. In the following, force feedback in a bilateral control system is discussed. Some basic architectures are proposed.

Considering a block diagram of a teleoperation system, as illustrated in Fig. 4.2, where the master, slave, and communication channel models are lumped into a linear-time-invariant (LTI) master-slave two-port network (MSN)

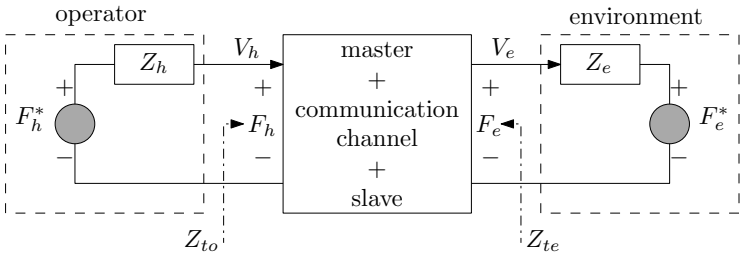


Figure 4.2: A teleoperation network block diagram.

block. The operator and environment are assumed to be in contact with the master and slave, and are modeled around their contact operating point in the Laplace domain by lumped LTI dynamics:

$$F_h = F_h^* - Z_h V_h \tag{4.1}$$
$$F_e = F_e^* + Z_e V_e \tag{4.2}$$

where Z_h, Z_e, V_h, V_e, F_h, F_e, F_h^*, and F_e^* are the master and slave impedances and velocities, the operator force on the master, the slave force on the environment, and the exogenous force inputs generated by the operator and the environment, respectively.

Depending on the choice of the network input and output variables I and O, impedance Z, admittance y, hybrid H, and inverse hybrid g, network matrices are defined as:

$$\begin{bmatrix} F_h \\ F_e \end{bmatrix} = O_Z = ZI_Z = \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix} \begin{bmatrix} V_h \\ -V_e \end{bmatrix} \tag{4.3}$$
$$\begin{bmatrix} V_h \\ -V_e \end{bmatrix} = O_y = yI_y = \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix} \begin{bmatrix} F_h \\ F_e \end{bmatrix} \tag{4.4}$$
$$\begin{bmatrix} F_h \\ -V_e \end{bmatrix} = O_H = HI_H = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} V_h \\ F_e \end{bmatrix} \tag{4.5}$$
$$\begin{bmatrix} V_h \\ F_e \end{bmatrix} = O_g = gI_g = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} F_h \\ -V_e \end{bmatrix} \tag{4.6}$$

where each of the above matrices, if they exist, can be found given any of the other matrices. The above representations fall into the immittance category defined as $P_P = O_P I_P$, $P = [p_{ij}]$ $i, j = 1, 2$, in which $O_P^T I_P = O_Z^T I_Z =$

$O_y^T I_y = O_H^T I_H = O_g^T I_g = F_h V_h - F_e V_e$ is the instantaneous power delivered to the MSN. Hence, the class of immittance representations is of particular interest to the energy-based stability analysis tools such as passivity theory.

Llewellyn's absolute stability condition is expressed in terms of the MSN immittance matrices in the following (Haykin, 1970):

An LTI two-port network is absolutely stable if and only if:

- p_{11} and p_{22} are positive and real, and
- the inequality:

$$\eta_P(\omega) = -\frac{Re(p_{12}p_{21})}{|p_{12}p_{21}|} + 2\frac{Re(p_{11})Re(p_{22})}{|p_{12}p_{21}|} \geq 1 \tag{4.7}$$

  holds on the $j\omega$ axis for all $\omega \geq 0$, where $\eta_P(\omega)$ is called the network stability parameter and $|\cdot|$ and $Re(\cdot)$ denote the absolute and real values of their corresponding arguments.

The positive realness of p_{11} and p_{22} implies passivity of the master and slave when there is no coupling between them, that is, when $p_{12} = p_{21} = 0$. This can also be seen as the passivity of the master and slave when they are free or clamped. On the other hand, the condition represented by Eq. 4.7 incorporates the effect of coupling. After expanding the positive realness condition, Llewellyn's absolute stability conditions are equivalent to the following conditions:

- the immittance parameters p_{11} and p_{22} have no poles in the open right-half-plane (RHP),
- any poles of p_{11} and p_{22} on the imaginary axis are simple and have real and positive residues, and
- the inequalities

$$Re(p_{11}) \geq 0 \tag{4.8}$$

$$\eta_P(\omega) = -\cos(\angle p_{12}p_{21}) + 2\frac{Re(p_{11})Re(p_{22})}{|p_{12}p_{21}|} \geq 1 \tag{4.9}$$

hold, where $\cos(\angle Z) = \frac{Re(Z)}{|Z|}$ for any complex Z. Llewellyn's criterion is valid for any member of the immittance class, and moreover the value of the

stability parameter is independent of the immittance matrix used, that is, $\eta_Z = \eta_y = \eta_H = \eta_g$ (Haykin, 1970).

(Hogan, 1989) has shown that the human arm impedance is highly adaptable and time varying, and although the muscular actuators and the neural feedback driving the arm are active systems, the human hand shows passive characteristics. Therefore, the operator can be modeled by a state independent (exogenous) input force and a passive impedance, as in Eq. 4.1 (Colgate and Hogan, 1988). As for the environment, most of the objects with which we interact are passive and absorb energy. Because the dynamic range of the operator impedance is not as wide as that of the environment, and in some cases the upper bound on the impedance of the object to be manipulated is known a priori, the above absolute stability analysis may provide us with conservative stability conditions.

Besides stability, transparency is the other principal goal in a teleoperation control system. Transparency can be described quantitatively as the transmitted impedance to the operator and to the environment, that is: $Z_{to} = \frac{F_h}{V_h}|_{F_e^*=0} = \frac{F_e}{V_e}|_{F_e^*=0} = Z_e$ and $Z_{te} = \frac{F_e}{-V_e}|_{V_h^*=0} = \frac{F_e}{-V_e}|_{V_h^*=0} = Z_h$, respectively, where Z_{to} and Z_{te} are the transmitted impedances to the operator and to the environment. Using Eqs. 4.1, 4.2 and 4.5, Z_{to} and Z_{te} can be expressed in terms of the MSN hybrid parameters as:

$$Z_{to} = \frac{h_{11} + \Delta h \cdot Z_e}{1 + h_{22} Z_e} \tag{4.10}$$

$$Z_{te} = \frac{h_{11} + Z_h}{\Delta h + h_{22} Z_h} \tag{4.11}$$

where $\Delta h = h_{11}h_{22} - h_{12}h_{21}$. If the hybrid parameters are not functions of Z_h and Z_e, perfect transparency can be achieved if

$$h_{11} = h_{22} = 0 \tag{4.12}$$

$$h_{12} = -h_{21} = 1 \tag{4.13}$$

is satisfied (Hannaford, 1989a). Hence, a perfectly transparent system is marginally absolutely stable, as $Re(h11) = 0$ and $\eta_H = 1$ in Eqs. 4.8 and 4.9, respectively. Therefore, to have higher stability robustness, perfect transparency has to be compromised. In addition, due to the presence of significant transmission delay, there is a trade-off between stability and performance; consequently, perfect transparency is not attainable in practice (Hannaford,

1989b; Lawrence, 1993; Salcudean et al., 1995). Therefore, there must be an examination of Z_{to} and Z_{te} for the infinite spectrum of the environment and operator impedance in order to evaluate the system transparency, which is an involved process. To ease the burden and to quantify transparency, Z_{to} and Z_{te} are examined for extreme values of Z_e and Z_h, respectively: that is, when the master and slave are in free motion ($Z_e = 0$ or $Z_h = 0$), or clamped ($Z_e \to \infty$ or $Z_h \to \infty$). If the network parameters are not functions of Z_h and Z_e, the minimum value and dynamic range of the transmitted impedances can be evaluated as follows:

$$Z_{to,min} = Z_{to}|_{Z_e=0} = h_{11} \tag{4.14}$$

$$Z_{to,width} = Z_{to}|_{Z_e\to\infty} - Z_{to,min} = \frac{-h_{12}h_{21}}{h_{22}} \tag{4.15}$$

$$Z_{te,min} = Z_{te}|_{Z_h=0} = \frac{h_{11}}{\Delta h} \tag{4.16}$$

$$Z_{te,width} = Z_{te}|_{Z_h\to\infty} - Z_{te,min} = \frac{-h_{12}h_{21}}{h_{22}\Delta h} \tag{4.17}$$

Here, the notion of $Z - width$ is borrowed from haptics (Colgate and Brown, 1994) to express the dynamic range of the impedance transmitted to the operator, and viceversa, while maintaining stability. The choice of $Z-width$ is compliant with its original definition in (Colgate and Brown, 1994) as Llewellyn's criterion guarantees passivity of the transmitted impedance. Good performance is then characterized by $|Z_{to,min}| \to 0$ and $|Z_{to,width}| \to \infty$, as well as $|Z_{te,min}| \to 0$ and $|Z_{te,width}| \to \infty$.

The performed analysis and the tools proposed for evaluation are commonly used to assess stability and performance in different types of bilateral teleoperation systems.

4.3 Types of teleoperation systems

The LTI dynamic models, represented in Eqs. 4.18, 4.19, 4.20 and 4.21, are used for impedance/admittance types of master and slave manipulators, where Z_m, Z_s, Y_m, Y_s and F_{cm}, F_{cs}, V_{cm}, V_{cs} denote the master and slave dynamics and their control inputs. Z_m, Z_s and Y_m, Y_s are typically low impedance and admittance dynamics, respectively.

$$Z_m V_h = F_h + F_{cm} \quad Impedance\ Master \tag{4.18}$$
$$Z_s V_e = -F_e + F_{cs} \quad Impedance\ Slave \tag{4.19}$$
$$Y_m F_h = V_h + V_{cm} \quad Admittance\ Master \tag{4.20}$$
$$Y_s V_e = -V_e + V_{cs} \quad Admittance\ Slave \tag{4.21}$$

Based on the above manipulator categories, there are four different types of teleoperation systems: impedance/impedance, impedance/admittance, admittance/impedance, and admittance/admittance. Fig. 4.3 and Fig. 4.4 show the block diagram of the four types of teleoperation systems controlled by general 4ch bilateral controllers. T_d denotes the communication channel time delay, and the C and E blocks denote the control compensator transfer functions.

In all four bilateral controllers, there are generally two types of control signals applied to the master and slave actuators. One type is from local controllers, that is, C_5, C_6, C_m, C_s and E_5, E_6, E_m, E_s, built around the master and slave. The other is from feedforward controllers, that is, C_1, C_2, C_3 and C_4, and E_1, E_2, E_3 and E_4, sending signals to the remote site. The feedforward control signals applied to the master or slave can be of either position or force type, which is determined by the type of manipulator.

4.4 Control architecture

The four-channel control architectures presented in Fig. 4.3 and Fig. 4.4 are simplified if only one signal - position or force - is transmitted from the master or slave. Four two-channel control architectures are possible for each configuration, named for the variable measured and sent to the remote site: force-force (F-F), position-position (P-P), force-position (F-P), and position-force (P-F).

An impedance-admittance position-force (P-F) two-channel control architecture along with the related control parameters leading to perfect transparency under ideal conditions is presented. Besides, an impedance-impedance four-channel bilateral control architecture with time delay compensation is presented too.

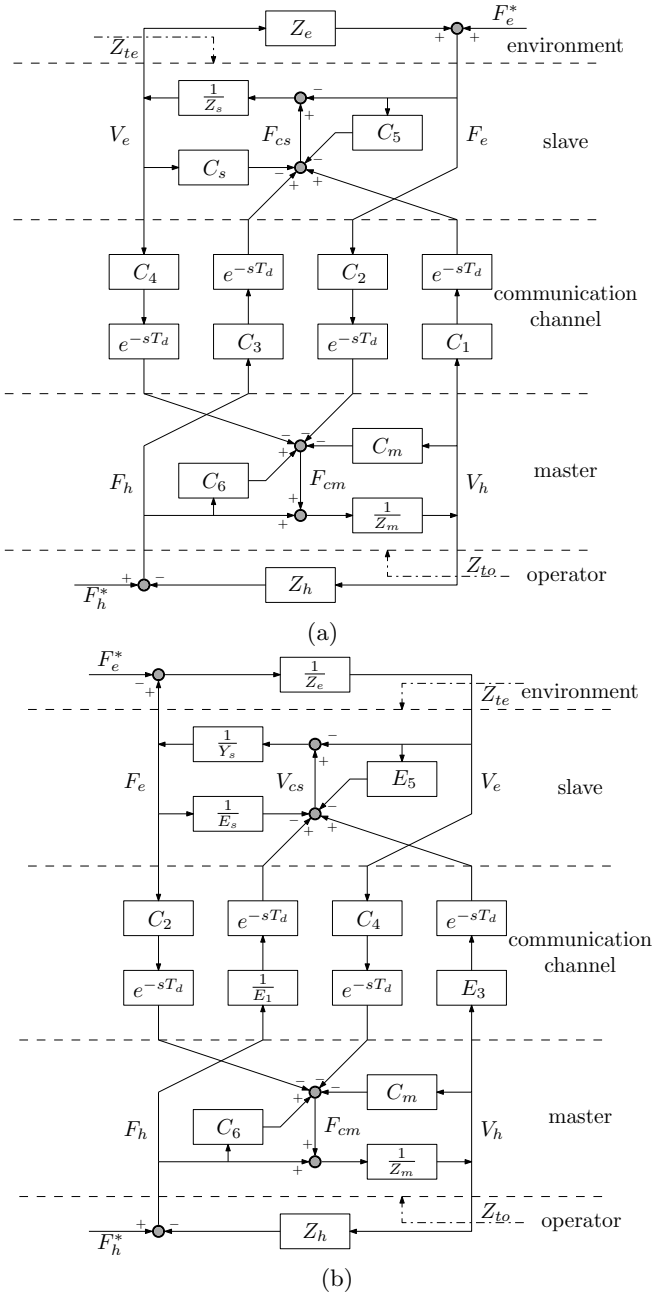


Figure 4.3: Block diagrams of the four types teleoperation systems: (a) Impedance-Impedance. (b) Impedance-Admittance.

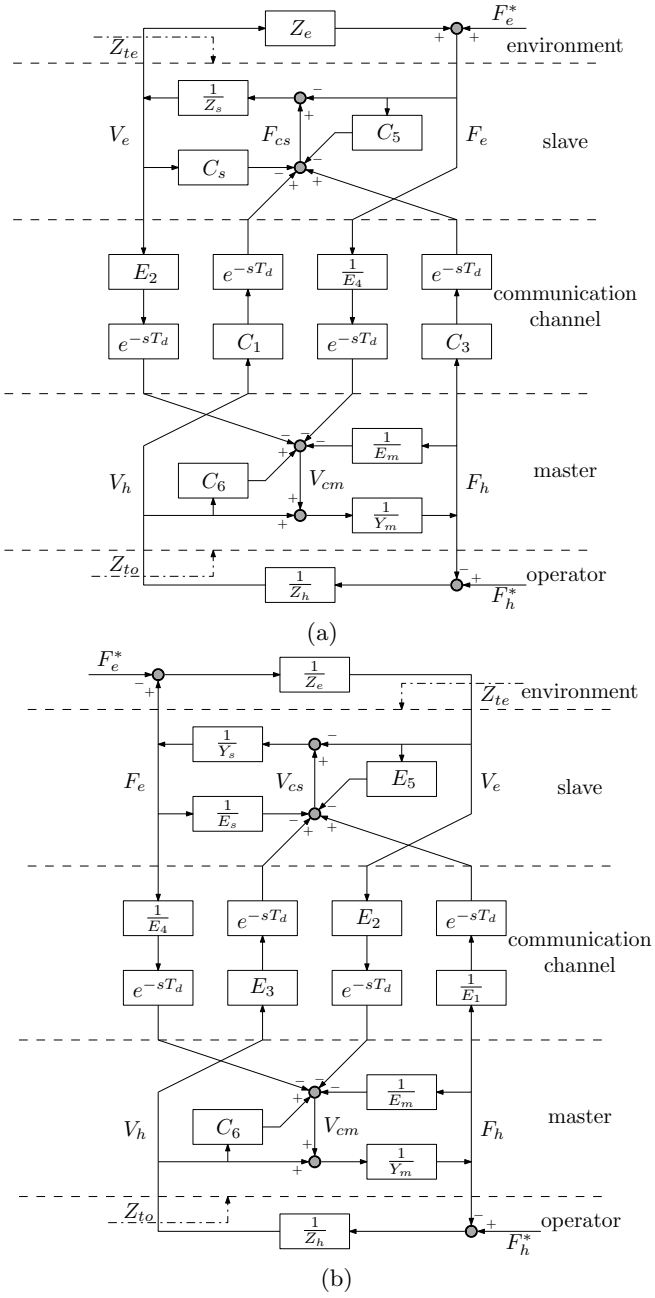


Figure 4.4: Block diagrams of the four types teleoperation systems: (a) Admittance-Impedance. (b) Admittance-Admittance.

4.4.1 Two-channel impedance-admittance type

Applying the control commands F_{cm} and V_{cs} to the impedance-admittance system, shown in Fig. 4.3b, the dynamics of the closed-loop system are expressed as,

$$\begin{aligned} Z_{cm}V_h + C_4 e^{-sT_d}V_e &= (1+C_6)F_h - C_2 e^{-sT_d}F_e && (4.22)\\ (1+E_5)V_e - E_3 e^{-sT_d}V_h &= E_1^{-1}e^{-sT_d}F_h - Y_{es}F_e && (4.23)\end{aligned}$$

where $Z_{cm} = Z_m + C_m$ and $Y_{es} = Y_s + E_s^{-1}$, with Y_s admittance of the slave. Using the Eqs. 4.5, 4.22 and 4.23 the hybrid parameters are:

$$h_{11} = \frac{Z_{cm}(1+E_5) + E_3C_4e^{-2sT_d}}{(1+C_6)(1+E_5) - E_1^{-1}C_4e^{-2sT_d}} \quad (4.24)$$

$$h_{12} = \frac{C_2(1+E_5)e^{-sT_d} - C_4Y_{es}e^{-sT_d}}{(1+C_6)(1+E_5) - E_1^{-1}C_4e^{-2sT_d}} \quad (4.25)$$

$$h_{21} = -\frac{E_1^{-1}Z_{cm}e^{-sT_d} + E_3(1+C_6)e^{-sT_d}}{(1+C_6)(1+E_5) - E_1^{-1}C_4e^{-2sT_d}} \quad (4.26)$$

$$h_{22} = \frac{Y_{es}(1+C_6) - C_2E_1^{-1}e^{-2sT_d}}{(1+C_6)(1+E_5) - E_1^{-1}C_4e^{-2sT_d}} \quad (4.27)$$

Using Eqs. 4.10 and 4.11, along with 4.24, 4.25, 4.26 and 4.27, the operator-transmitted and the environment-transmitted impedance are obtained as:

$$Z_{to} = \frac{[Z_{cm}(1+E_5) + E_3C_4e^{-2sT_d}] + [Y_{es}Z_{cm} + E_3C_2e^{-2sT_d}]Z_e}{[(1+C_6)(1+E_5) - E_1^{-1}C_4e^{-2sT_d}] + [Y_{es}(1+C_6) - C_2E_1^{-1}e^{-2sT_d}]Z_e} \quad (4.28)$$

$$Z_{te} = \frac{[Z_{cm}(1+E_5) + E_3C_4e^{-2sT_d}] + [(1+E_5)(1+C_6) - E_1^{-1}C_4e^{-2sT_d}]Z_h}{[Z_{cm}Y_{es} + E_3C_2e^{-2sT_d}] + [Y_{es}(1+C_6) - C_2E_1^{-1}e^{-2sT_d}]Z_h} \quad (4.29)$$

If time delay T_d is negligible, using the impedance-admittance transparency-optimized control law:

$$\begin{cases} E_1^{-1} = Y_{es} \\ C_2 = 1+C_6 \neq 0 \\ E_3 = 1+E_5 \neq 0 \\ C_4 = -Z_{cm} \end{cases} \quad (4.30)$$

satisfies Eqs. 4.12 and 4.13, providing perfect transparency.

In a two-channel impedance-admittance control architecture, the direct force feedforward from the slave to the master and the coordinating force feedforward from the master to the slave are removed; that is, $C_2 = E_3 = 0$. Hence, Eqs. 4.24, 4.25, 4.26 and 4.27 become,

$$h_{11} = \frac{Z_{cm}(1+E_5)}{(1+C_6)(1+E_5) - E_1^{-1}C_4e^{-2sT_d}} \tag{4.31}$$

$$h_{12} = \frac{-C_4Y_{es}e^{-sT_d}}{(1+C_6)(1+E_5) - E_1^{-1}C_4e^{-2sT_d}} \tag{4.32}$$

$$h_{21} = -\frac{E_1^{-1}Z_{cm}e^{-sT_d}}{(1+C_6)(1+E_5) - E_1^{-1}C_4e^{-2sT_d}} \tag{4.33}$$

$$h_{22} = \frac{Y_{es}(1+C_6)}{(1+C_6)(1+E_5) - E_1^{-1}C_4e^{-2sT_d}} \tag{4.34}$$

therefore, the inverse hybrid parameters are:

$$g_{11} = \frac{h_{22}}{\Delta h} = \frac{1+C_6}{Z_{cm}} \tag{4.35}$$

$$g_{12} = \frac{-h_{12}}{\Delta h} = \frac{C_4e^{-sT_d}}{Z_{cm}} \tag{4.36}$$

$$g_{21} = \frac{-h_{21}}{\Delta h} = \frac{E_1^{-1}e^{-sT_d}}{Y_{es}} \tag{4.37}$$

$$g_{22} = \frac{h_{11}}{\Delta h} = \frac{1+E_5}{Y_{es}} \tag{4.38}$$

which leads to the following inverse hybrid matrix:

$$g = \begin{bmatrix} \frac{1+C_6}{Z_{cm}} & \frac{C_4e^{-sT_d}}{Z_{cm}} \\ \frac{E_1^{-1}e^{-sT_d}}{Y_{es}} & \frac{1+E_5}{Y_{es}} \end{bmatrix}$$

In the same way, Eqs. 4.28 and 4.29 become,

$$Z_{to} = \frac{[Z_{cm}(1+E_5)] + [Y_{es}Z_{cm}]Z_e}{[(1+C_6)(1+E_5) - E_1^{-1}C_4e^{-2sT_d}] + [Y_{es}(1+C_6)]Z_e} \tag{4.39}$$

$$Z_{te} = \frac{[Z_{cm}(1+E_5)] + [(1+C_5)(1+C_6) - E_1^{-1}C_4e^{-2sT_d}]Z_h}{[Z_{cm}Y_{es}] + [Y_{es}(1+C_6)]Z_h} \tag{4.40}$$

as well as Eq. 4.30 that becomes,

$$\begin{cases} E_1^{-1} &= Y_{es} \\ C_4 &= -Z_{cm} \end{cases} \tag{4.41}$$

for which there is no possibility of canceling the dynamics of master and slave, as shown in the resulting inverse hybrid matrix:

$$g = \begin{bmatrix} \frac{1+C_6}{Z_{cm}} & 1 \\ -1 & \frac{1+E_5}{Y_{es}} \end{bmatrix} \tag{4.42}$$

In order to study the stability of this control architecture, the inverse hybrid parameters can be used to easily evaluate the parameter η_{pf}, derived from Eq. 4.9, as follows:

$$\eta_{pf}(\omega) = \eta_{pf1} + \eta_{pf2} = -\cos(\angle \frac{E_1^{-1} C_4 e^{-j2\omega T_d}}{Y_{es} Z_{cm}}) + 2 \frac{Re(\frac{1+C_6}{Z_{cm}}) Re(\frac{1+E_5}{Y_{es}})}{|\frac{E_1^{-1} C_4 e^{-j2\omega T_d}}{Y_{es} Z_{cm}}|} \tag{4.43}$$

The first term η_{pf1} is defined by a cosine function with changed sign, then it can assume the minimum value as -1. To satisfy the Llewellyn's absolute stability conditions, the second term η_{pf2} must be greater than or equal to 2, that is $Re(\frac{1+C_6}{Z_{cm}}) Re(\frac{1+E_5}{Y_{es}}) \geq |\frac{E_1^{-1} C_4 e^{-j2\omega T_d}}{Y_{es} Z_{cm}}|$. Hence, for increasing the robustness of the stability, the local force feedback of the master C_6 can be increased, as well as the damping of the local position feedback E_5, or the value of the feedforward parameters E_1^{-1} and C_4 can be reduced as well. A tight local position feedback of the master C_m, or a tight local force feedback of the slave E_s^{-1}, may have negative effects on the absolute stability. Using Eq. 4.41, η_{pf} defined by Eq. 4.43, can be simplified as follows:

$$\eta_{pf}(\omega) = \cos(\angle - j2\omega T_d) + 2Re(\frac{1+E_5}{Y_{es}}) Re(\frac{1+C_6}{Z_{cm}}) \tag{4.44}$$

In order to achieve the absolute stability: $Re(\frac{1+E_5}{Y_{es}}) Re(\frac{1+C_6}{Z_{cm}}) \geq 1$. If C_6 and E_5 are scalar gains, the absolute stability may be guaranteed in only a certain range of frequencies, because at low and high frequencies $Re(\frac{1}{Z_{cm}}) \to 0$, by substituting Z_m and assuming a position PD controller for C_m. Hence, the result depends on $Re(\frac{1}{Y_{es}})$ which, by using the admittance slave Y_s defined in Eq. 4.21, can counteract the former by means of E_s, if there is no scalar gain. Finally, if time delay is negligible, then the absolute stability of the system is always achieved, with $\eta_{pf1} = 1, \forall \omega \geq 0$, and thus $\eta_{pf} \geq 1, \forall \omega \geq 0$.

To investigate performance, the behavior of the system for an infinite spectrum of transmitted impedance, either from the operator or the environment site, must be carried out. Hence, the minimum value and dynamic range of the operator-transmitted impedance can be evaluated by using Eqs. 4.14 and 4.15 along with Eq. 4.39, as follows:

$$Z_{to,min} = \frac{Z_{cm}(1+E_5)}{(1+C_6)(1+E_5) - E_1^{-1}C_4e^{-2sT_d}} \tag{4.45}$$

$$Z_{to,width} = \frac{-E_1^{-1}C_4Z_{cm}e^{-2sT_d}}{(1+C_6)[(1+C_6)(1+E_5) - E_1^{-1}C_4e^{-2sT_d}]} \tag{4.46}$$

Similarly, the environment-transmitted impedance can be evaluated by using Eqs. 4.16 and 4.17 along with Eq. 4.40, as follows:

$$Z_{te,min} = \frac{(1+E_5)}{Y_{es}} \tag{4.47}$$

$$Z_{te,width} = \frac{-E_1^{-1}C_4e^{-2sT_d}}{(1+C_6)Y_{es}} \tag{4.48}$$

4.4.2 Four-channel impedance-impedance type

Applying the control commands F_{cm} and F_{cs} to the impedance-impedance system, shown in Fig. 4.3a, the dynamics of the closed-loop system are expressed as,

$$Z_{cm}V_h + C_4e^{-sT_d}V_e = (1+C_6)F_h - C_2e^{-sT_d}F_e \tag{4.49}$$

$$Z_{cs}V_e + C_1e^{-sT_d}V_h = C_3e^{-sT_d}F_h + (1+C_5)F_e \tag{4.50}$$

where $Z_{cm} = Z_m + C_m$ and $Z_{cs} = Z_s + C_s$. To analyze the system stability and performance, the MSN hybrid parameters are needed. These parameters can be derived in terms of the system and control parameters from Eqs. 4.5, 4.49 and 4.50 as,

$$h_{11} = \frac{Z_{cm}Z_{cs} + C_1C_4e^{-2sT_d}}{(1+C_6)Z_{cs} - C_3C_4e^{-2sT_d}} \tag{4.51}$$

$$h_{12} = \frac{C_2Z_{cs}e^{-sT_d} - C_4(1+C_5)e^{-sT_d}}{(1+C_6)Z_{cs} - C_3C_4e^{-2sT_d}} \tag{4.52}$$

$$h_{21} = -\frac{C_3Z_{cm}e^{-sT_d} + C_1(1+C_6)e^{-sT_d}}{(1+C_6)Z_{cs} - C_3C_4e^{-2sT_d}} \tag{4.53}$$

$$h_{22} = \frac{(1+C_5)(1+C_6) - C_2C_3e^{-2sT_d}}{(1+C_6)Z_{cs} - C_3C_4e^{-2sT_d}} \tag{4.54}$$

Using Eqs. 4.10 and 4.11, along with 4.51, 4.52, 4.53 and 4.54, the operator-transmitted and the environment-transmitted impedance can be expressed as:

$$Z_{to} = \frac{(Z_{cm}Z_{cs} + C_1C_4e^{-2sT_d}) + [(1+C_5)Z_{cm} + C_1C_2e^{-2sT_d}]Z_e}{[(1+C_6)Z_{cs} - C_3C_4e^{-2sT_d}] + [(1+C_5)(1+C_6) - C_2C_3e^{-2sT_d}]Z_e} \tag{4.55}$$

$$Z_{te} = \frac{(Z_{cm}Z_{cs} + C_1C_4e^{-2sT_d}) + [(1+C_6)Z_{cs} - C_3C_4e^{-2sT_d}]Z_h}{[(1+C_5)Z_{cm} - C_1C_2e^{-2sT_d}] + [(1+C_5)(1+C_6) - C_2C_3e^{-2sT_d}]Z_h} \tag{4.56}$$

If time delay T_d is negligible, using the impedance-impedance transparency-optimized control law,

$$\begin{cases} C_1 = Z_{cs} \\ C_2 = 1 + C_6 \\ C_3 = 1 + C_5 \\ C_4 = -Z_{cm} \end{cases} \tag{4.57}$$

with $C_2 \neq 0$ and $C_3 \neq 0$, satisfies 4.12 and 4.13 providing perfect transparency. Hence, the master and slave are effectively removed and the operator and the environment are virtually connected. The position and velocity of master and slave are transmitted, for which: $C_1 = C_s = B_s + \frac{K_s}{s}$ and $C_4 = -C_m = -(B_m + \frac{K_m}{s})$. The impedance model of master and slave are $Z_m = M_m s$ and $Z_s = M_s s$, respectively.

Stability of the transparency-optimized four-channel controller leads to a characteristic equation of the system, that under ideal conditions, is:

$$\Delta_0 = (C_2Z_{cs} + C_3Z_{cm})(Z_h + Z_e) = 0 \tag{4.58}$$

This indicates that if C_2 and C_3 are single gains and the operator and environment are passive, the system remains stable if $C_2 > -min(\frac{M_m}{M_s}, \frac{B_m}{B_s}, \frac{K_m}{K_s})C_3$ holds. In terms of stability robustness, although this system is stable for $C_2, C_3 > 0$, because $h_{11} = h_{22} = 0$ and $h_{12} = -h_{12} = 1$, the system absolute stability is marginal as $\eta_H = 1$. Note that because the term $C_2 Z_{cs} + C_3 Z_{cm}$ is present in the denominator of h_{11} and h_{22} for $T_d \approx 0$, it cannot have roots in the RHP to be used in Llewellyn's conditions, represented by Eq. 4.8 and 4.9.

In the absence of time delay in the communication line, the analysis of stability and performance is straightforward and stable perfect transparency is achievable. However, when significant delays are present, as in teleoperation over the Internet, both stability and transparency are compromised.

4.5 Time delay compensation

When there is significant delay in the communication line between the master and slave, control architectures can suffer instability problems. This can be traced to the communication block in Fig. 4.2, where the power entering the left side and exiting the right side do not add up. Rather energy may be generated inside the block, which induces the instability (Anderson and Spong, 1989a).

Several approaches to operate under delay have been studied (Eusebi and Melchiorri, 1998), in particular shared compliant control (Kim et al., 1992) and the addition of local force loops (Hashtrudi-Zaad and Salcudean, 2002); however, using Internet as the communication channel adds variability to the delay (Oboe and Fiorini, 1998).

4.5.1 Scattering approach

(Anderson and Spong, 1989b,a, 1988) introduced the notion of scattering variables in order to overcome the problem of time-delay in bilateral teleoperation. Such approach renders the teleoperator system passive.

The bilateral teleoperator considered is depicted in Fig. 4.5. The relationship between the forces and velocities at all ports can be represented, in the LTI case by the hybrid matrix (Eq. 4.5) which enters into the definition of the scattering operator.

The scattering operator is defined in terms of an incident wave $(F(t)+V(t))$ and a reflected wave $(F(t) - V(t))$ as: $F(t) - V(t) = S(F(t) + V(t))$. The

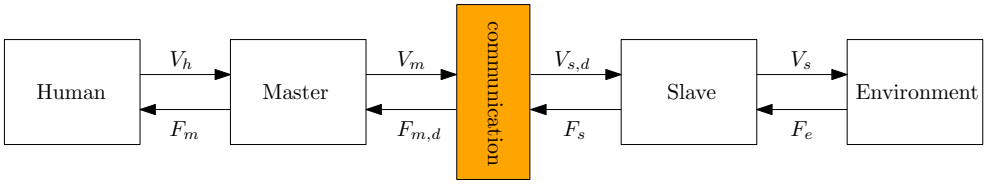


Figure 4.5: Teleoperator scheme.

scattering matrix in the frequency domain can be represented in terms of the hybrid matrix by simple loop transformation,

$$S(s) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} (H(s) - I)(H(s) + I)^{-1} \tag{4.59}$$

In order to guarantee passivity, the scattered wave cannot have energetic content greater than the incident wave, hence with respect to the scattering operator S. In fact, an n-port system is passive if and only if $||S(j\omega)||_\infty \leq 1$ of the corresponding scattering matrix (Anderson and Spong, 1989a).

Let F and V be the force and velocity vectors, respectively, and define P_{in} and P_{out} to be the input and output power, respectively, then the power difference is given by

$$\begin{aligned} \Delta P &= P_{in} - P_{out} = F^T V \\ &= (\tfrac{F+V}{2})^T(\tfrac{F+V}{2}) - (\tfrac{F-V}{2})^T(\tfrac{F-V}{2}) \end{aligned} \tag{4.60}$$

Defining the scattering variables $s_+ = (\frac{F+V}{2})^T$ and $s_- = (\frac{F-V}{2})$, then ΔP can be rewritten in terms of the scattering variables as:

$$\Delta P = s_+^T s_+ - s_-^T s_- = s_+^T (I - S^T S) s_+ \tag{4.61}$$

As the power difference must be nonnegative, the condition on the maximum singular value of the scattering matrix must be: $||S||_\infty = \bar{\sigma}(S^T(j\omega)S(j\omega)) \leq 1$.

4.5.1.1 Constant time delay

The scattering formulation developed in Section 4.5.1, can be used to develop control laws in order to achieve stability for systems with constant time delay. Consider the system (L) transmitting signals though constant delay T with the following control law:

$$\begin{array}{rcl} F_m(t) & = & -F_{m,d}(t) = -F_s(t-T) \\ F_s(t) & = & K_s \int (V_{s,d} - V_s)dt + B_s(V_{s,d} - V_s) \end{array} \tag{4.62}$$

where $F_s(t)$ is called the coordinating torque, and $V_{s,d}(t) = V_m(t-T)$.

The above signals result in a hybrid matrix composed of pure delay elements that render the norm of the scattering operator as $||S||_\infty = \infty$, which is not passive. Therefore, the pure delay communication channel generates energy which possibly destabilizes the teleoperator. This problem can be solved by emulating the behavior of transmission lines by adopting the scattering formulation which gives passivity to the communication channel.

$$\begin{array}{rcl} F_{m,d}(t) & = & F_s(t-T) + Z_0(V_m(t) - V_{s,d}(t-T)) \\ V_{s,d}(t) & = & V_m(t-T) + Z_0^{-1}(F_{m,d}(t) - F_s(t-T)) \end{array} \tag{4.63}$$

where Z_0 is the characteristic impedance inherent to transmission lines theory, that we can view as the ratio of a transformer placed between forces and velocities, that makes their values comparable. Note that all the signals in Eq. 4.63 could be vector-valued in which case Z_0 would be a matrix (Anderson and Spong, 1989b). This control scheme results in the scattering matrix:

$$S(s) = \begin{bmatrix} 0 & e^{-sT} \\ e^{-sT} & 0 \end{bmatrix} \tag{4.64}$$

which has norm $||S||_\infty = 1$. Hence, a passive communication channel is achieved. Moreover, as we discussed earlier, we can infer the stability of the teleoperator for all passive environments and a passive behavior by the human operator. Such formulation achieves velocity and force matching. However, position mismatch or drift between the master the slave positions remains still a problem.

4.5.2 Wave variables

A conceptually similar formulation to the scattering formulation appeared in (Niemeyer and Slotine, 1991a), the so called wave variables formulation. As illustrated in Fig. 4.6, instead of exchanging as reference signals the power variables V_m and F_s, the wave variables are transmitted u_m and u_s, which are given by:

$$\begin{array}{rcl} u_m(t) & = & \frac{1}{\sqrt{2b}}(F_{m,d}(t) + bV_m(t)) \\ u_s(t) & = & \frac{1}{\sqrt{2b}}(F_s(t) - bV_{s,d}(t)) \end{array} \tag{4.65}$$

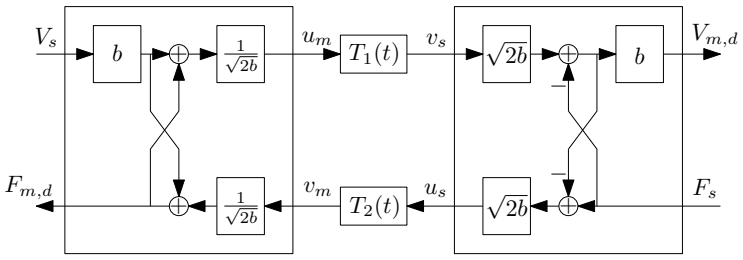


Figure 4.6: Wave variables.

where $F_{m,d}$ and $V_{s,d}$ are the received power signals on the master and slave side, respectively. This formulation is identical to the scattering formulation, with b being the characteristic impedance of the transmission line. As in Eq. 4.60, the total power flow in the communication channel can be written equivalently in terms of the wave variables:

$$\begin{aligned} P(t) &= F_{m,d}^T(t)V_m(t) - F_s^T(t)V_{s,d}(t) \\ &= \tfrac{1}{2}(u_m^T(t)u_m(t) - v_m^T(t)v_m(t)) + \tfrac{1}{2}(u_s^T(t)u_s(t) - v_s^T(t)v_s(t)) \end{aligned} \tag{4.66}$$

where v_m and v_s are the received wave signals, and the reference signals on both sides of the channel are derived as

$$\begin{aligned} F_{m,d} &= bV_m(t) + \sqrt{2b}v_m \\ V_{s,d} &= \tfrac{1}{b}(\sqrt{2b}v_s(t) - F_s(t)) \end{aligned} \tag{4.67}$$

Note that in the last equation we have the symmetry in defining the wave variables (Eq. 4.65) in the sense that given any two of the power and wave variables, the remaining variables can be easily derived. Moreover, the master and slave can be put under force or velocity reference control. When there is a constant time delay in the communication channel ($T_i(t) = T_i, i = 1, 2$), the wave formulation gives the same transformation that in Eq. 4.63, and the passivity analysis can be alternatively performed in time domain. From Eq. 4.66 we can see that:

$$E(t) = \frac{1}{2}\left\{\int_{t-T}^{t}(u_m^T(\zeta)u_m(\zeta) + u_s^T(\zeta)u_s(\zeta))d\zeta\right\} \geq 0 \tag{4.68}$$

and hence, the channel is passive.

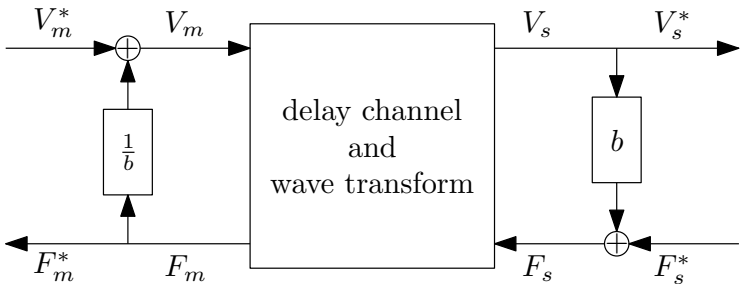


Figure 4.7: Impedance matching.

Due to the intrinsic passivity of the wave formulation, several control strategies can be possible in the wave domain that otherwise cause the loss of passivity when performed directly in the power variables domain.

4.5.2.1 Matching impedances

In the context of transmission line theory, it is well known that if the load that terminates the line has a different impedance than the characteristic impedance of the transmission line then wave reflections occur. In the case of bilateral teleoperation, such reflections degrade the performance of the system. This led to the introduction of impedance-matching elements b at each end of the communication channel (Fig. 4.7). Note that this is the basic setting that can be adapted to impedance matching when the slave manipulator is placed under impedance or force control as discussed in (Niemeyer and Slotine, 1991b).

Arguing that impedance matching elements at both sides of the communication block affects position tracking, (Benedetti et al., 2001) removed the matching element on the master's side, which results in a smaller position drift given by:

$$x_m^*(t-T) - x_s(t) = \frac{1}{2b}\int_{t-2T}^{t} F_s^*(\zeta)d\zeta \tag{4.69}$$

as compared to the drift that results from using the matching elements on both side of the channel given by:

$$x_m^*(t-T) - x_s(t) = \frac{1}{b}\int_{0}^{t} F_s^*(\zeta)d\zeta + x_s(t) \tag{4.70}$$

It is also argued that an increase in the value of b results in a smaller position drift. Of course this comes at the expense of extra damping which could affect negatively the performance of the system.

4.5.3 Geometric scattering

(Stramigioli et al., 2002) raises the approach of the scattering operator in Subsection 4.5.1 into a geometric setting, proposing the results of Subsections 4.5.1 and 4.5.2 in a compact form.

Consider the vector space $\mathcal{V}$ of flows (voltages, forces) and its dual $\mathcal{V}^*$ the space of efforts (currents, velocities) that form a Cartesian product space given by:

$$\mathcal{D} = \mathcal{V} \times \mathcal{V}^* \tag{4.71}$$

where $(f, e) \in \mathcal{D}$. Then, on the space D, there exists a unique way (for each given impedance Z) to decompose it into two eigensubspaces as:

$$\mathcal{D} = \mathcal{S}_Z^+ \oplus \mathcal{S}_Z^- \tag{4.72}$$

where $\mathcal{S}_Z^+$ and $\mathcal{S}_Z^-$ are subspaces of the incident and reflected scattered variables, respectively. This leads to the power decomposition theorem (notice that this gives the geometric counterpart of the result in Eq. 4.61) proposed by (Stramigioli et al., 2002), and states that $\forall (f, e) \in \mathcal{D}$ and any $Z = Z^T > 0$, the following holds:

$$\langle e, f \rangle = \frac{1}{2}||s_Z^+||_+^2 - \frac{1}{2}||s_Z^-||_-^2 \tag{4.73}$$

where $s_Z^+ \in \mathcal{S}_Z^+$, $s_Z^- \in \mathcal{S}_Z^-$, $(f, e) = s_Z^+ + s_Z^-$, and $||.||_+$ and $||.||_-$ are induced inner products on $\mathcal{S}_Z^+$ and $\mathcal{S}_Z^-$, respectively.

More general conditions for the cases of perfect and imperfect impedance matching can also be found in (Stramigioli et al., 2002).

4.5.4 H_∞ and μ-synthesis design

H_∞ and μ-synthesis design procedures can be used to derive compensators for delayed teleoperators, that take into account worst case upper bound on the delay values in the forward (master-to-slave) and backward (slave-to-master) communication (Leung et al., 1995). Consider the linear system (L), from which the models for the master and slave can be written in the frequency

domain as $P_m(s)$ and $P_s(s)$, respectively. A two-step design procedure can be performed in order to design compensators for the free motion case (using H_∞) and then for the delayed constrained motion (using μ-synthesis).

4.5.4.1 Free motion controllers

The controllers C_m and C_s for the master and slave, respectively, can be designed separately within the context of H_∞ as follows:

$$\begin{aligned} C_m: \quad & z = \begin{bmatrix} W_{m1}(F_h - V_m) \\ W_{m2}F_{m1} \end{bmatrix} & w = \begin{bmatrix} F_h \\ d_{m1} \end{bmatrix} \\ & y = V_m + W_{m3}d_{m1} & u = F_{m1} \\ C_s: \quad & z = \begin{bmatrix} W_{s1}(V_m - V_s) \\ W_{s2}F_{s1} \end{bmatrix} & w = \begin{bmatrix} F_h \\ d_{s1} \\ d_{s2} \end{bmatrix} \\ & y = \begin{bmatrix} V_m + W_{s3}d_{s1} \\ V_s + W_{s4}d_{s2} \end{bmatrix} & u = F_{s1} \end{aligned} \tag{4.74}$$

where in both cases z, w, y, u, d_* and W_* are the performance output, exogenous signals, measured outputs, control inputs, disturbances and weighting matrices, respectively.

4.5.4.2 Constrained motion controller

Having designed optimal controllers for the free motion, the delay can be considered as a perturbation to the constrained motion case, i.e. when the slave is in contact with the environment of known impedance Z_e, and the extra controller C_c can be utilized to account for both delays and constrained motion. The delays in the forward and backward channels can be combined into a single element T which appears as perturbation to the system of the form:

$$\Delta_T(s) = e^{-sT} - 1 \tag{4.75}$$

for which $||\Delta_T(j\omega)||_\infty = 2$. This perturbation can be filtered to render its norm < 1, and the μ-synthesis design procedure can be applied to the system shown in Fig. 4.8, to design C_s, where:

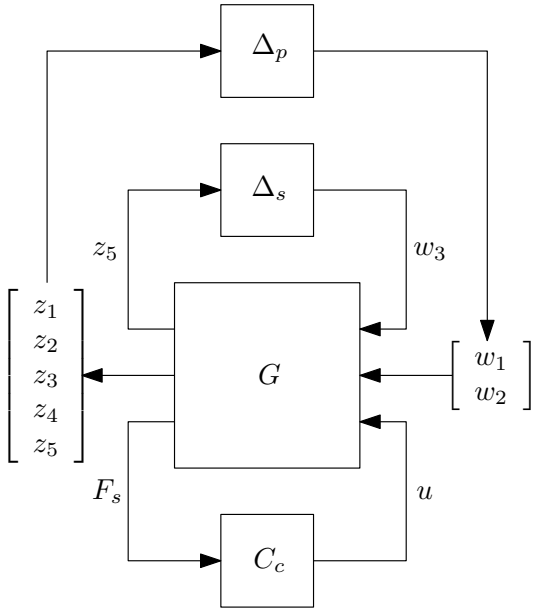


Figure 4.8: μ-synthesis.

$$C_s: \quad z = \begin{bmatrix} W_1(V_m - V_s) \\ W_2(F_s - (\tau_{m1} + F_{m2})) \\ W_3 F_{m2} \\ W_4 F_{s2} \\ z_5 \end{bmatrix} \quad w = \begin{bmatrix} F_h \\ F_b \\ w_3 \end{bmatrix} \tag{4.76}$$

$$y = F_s \qquad u = \begin{bmatrix} \tau_{m2} \\ \tau_{s2} \end{bmatrix}$$

and Δ_p is a fictitious performance perturbation.

The above method can be applied for all possible delays, within a certain range, which in many cases, is too conservative. Thus, the use of gain scheduling was suggested by (Sano et al., 1998), in which the design of the controller corresponds to the encountered delay. The controller is then varied according to updated measurements of the delay, which is suitable for teleoperation through Internet.

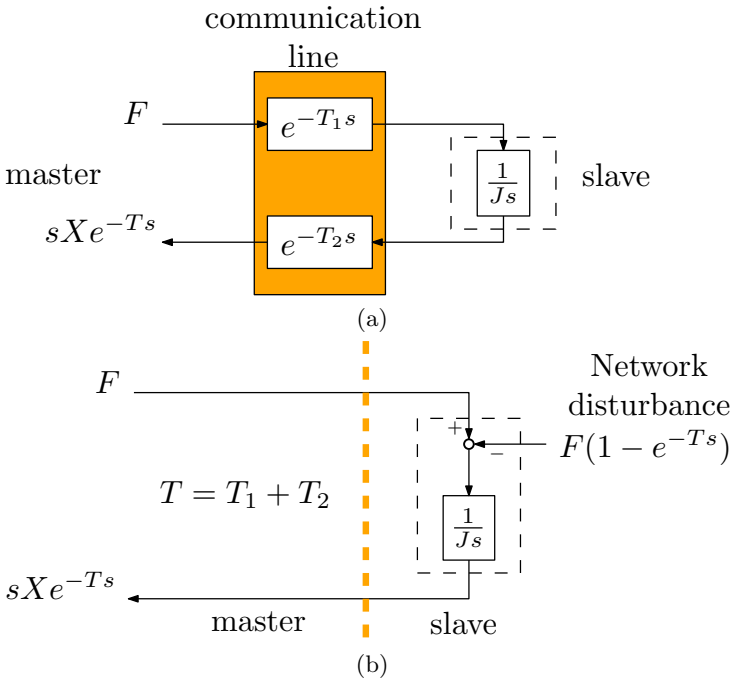


Figure 4.9: Representation of the Network Disturbance concept. (a) Bilateral system with time delay. (b) Bilateral system with Network Disturbance.

4.5.5 Communication disturbance observer (CDOB)

This time delay compensation method is based on the concept of network disturbance (Natori et al., 2008). In this particular case, time delay is considered as a disturbance that is affecting the system on the slave side (Natori et al., 2010).

This method is introduced using a simple example of a bilateral teleoperation system with time delay, as shown in Fig. 4.9a. The control input for the slave (force or torque dimension) is F and sXe^{Ts} is the slave output or the feedback signal for the master (velocity or angular velocity dimension). J is the moment of inertia of the slave. T_1 is the time delay from master to slave and T_2 is the time delay from slave to master ($T = T_1 + T_2$). As illustrated in Fig. 4.9a, the output of slave or the feedback signal that goes to the master sXe^{-Ts} is delayed with respect to the input F of the slave.

Taking into account the concept of network disturbance, the situation that is represented in Fig. 4.9a can be transformed to Fig. 4.9b, in which there

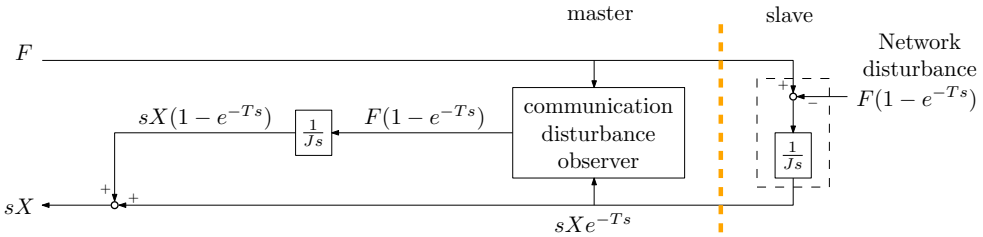


Figure 4.10: Time delay compensation by CDOB in a bilateral teleoperation system.

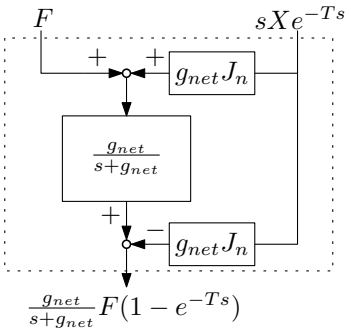


Figure 4.11: Internal structure of CDOB.

is no time delay element; however, there remains a network disturbance, that has this form:

$$D_{net}(s) = F(1 - e^{-Ts}) \tag{4.77}$$

where the ND is estimated by CDOB and used for time delay compensation, as shown in Fig. 4.10.

The internal structure of CDOB, based on DOB, is shown in Fig. 4.11. If $g_{net} \to \infty$ (ideal ND estimation), time delay in the communication line has no effect in the position controller of the master device, since it's compensated by CDOB. Therefore, this method cancels the time delay effect of sXe^{-Ts}, as the feedback signal sX is not affected anymore by time delay.

This time delay compensation method works without delay time model (the internal structure of CDOB does not include any delay time model, as shown Fig. 4.11) contrary to model-based or predictive control approaches.

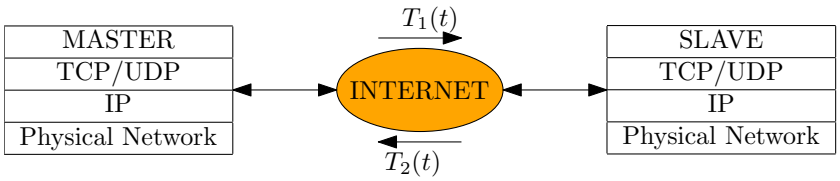


Figure 4.12: Teleoperation through Internet.

4.6 Web-based teleoperation

Teleoperation over the internet began in the middle of the nighties (Goldberg et al., 1995) and has been an active research area since then (Oboe, 2001, 2003). Communicating information across a packet-switched network results in random, time-varying delays that can reach very high values and eventually lead to loss of packets. As a result, the performance of the teleoperated system can be deteriorated dramatically, and the system may become unstable.

As illustrated in Fig. 4.12, the master and slave have to transport their discrete-time information down the software layers until the physical layer after which the data packets undergo random time-varying delays $T_1(t)$ and $T_2(t)$, the forward and backward delays, respectively, distorting the transmitted signals. A choice between using transmission control protocol (TCP) or user datagram protocol (UDP) has to be done based on their performance; both residing at the transport layer in the ISO 7-layer reference model. On one hand, TCP provides reliable two-way communication and guarantees data delivery at the cost of retransmissions and long timeouts that are detrimental in real-time applications such as teleoperation. On the other hand, UDP does not require packet reception confirmation (at the expense of unrecoverable data loss) eliminating unnecessary waiting time, which makes it suitable for real-time applications, such as teleoperation (Oboe, 2001). Results concerning fixed time delays had to be reexamined under the effects of the newly emerging communication medium, the Internet. For example, some earlier problems such as position drift between the master and slave that result from passivity-based control methods are increased by time-varying delays.

[3]For further information, see Hashtrudi-Zaad, K. and Salcudean, S. E. (2001). Analysis of control architectures for teleoperation systems with impedance/admittance master and slave manipulators. *The International Journal of Robotics Research*, 20(6):419-445. Also see Hokayem, P. F. and Spong, M. W. (2006). Bilateral teleoperation: An historical survey. *Automatica*, 42(12):2035-2057.

Chapter 5

Remote rehabilitation

For a post-stroke patient, the virtual reality based therapy contributes not only in regaining some motor skills but also, and fundamentally, to recover the mobility that was lost. However, this type of approach can't reproduce on the operational therapist a direct and immediate force feedback, which is a key issue, in order to test and analyze the degree of effectiveness of the proposed exercises. The developed activity presents an application of remote rehabilitation. Achieving a bilateral interaction between the therapist and the patient, this application may allow a reliable evaluation about the conditions of the patient. The system is based on a hand orthosis which is in bilateral interaction with the master device at the therapist's side.

5.1 Overview

In recent years, remote rehabilitation systems are taking a higher interest due to the multiple applications and advantages it actually has. As the internet access is already available in almost every home, the idea of doing remote rehabilitation is taking a wide consent because it can grant a high possibility of success. In fact, it can bring a medical care program to the patient's house, as well as reducing the patient hospitalization time (Outpatient Service Trialists, 2003).

When a rehabilitation program is carried out with the support of haptic interfaces it can help post-stroke patients to relearn the essential movements that were lost when part of the brain was damaged. It has been proven that

a post-stroke rehabilitation treatment has a higher possibility of success when it's implemented immediately after the critical phase (Oujamaa et al., 2009).

This kind of application grants a reliable evaluation of the conditions of the patient. We propose here a system that performs a remote evaluation of the characteristics of motion and functionality of the hand in patients with neurological impairments. The proposed system is based on a hand orthosis which is in bilateral interaction with the master device at the therapist's side. We propose two bilateral control architectures in order to guarantee an stable interaction between the master and the slave device, even in case of variable network conditions (i.e. Internet).

5.2 Master/Slave system

5.2.1 Master device

The master device was properly design to reproduce a realistic sensation during its manipulation. Because this is a clue issue for the therapist, the master device was implemented in two parts. One part consists on four rigidly connected fingers (the thumb is not taken into account) and the palm of a prosthetic hand. The second part considers the former, that is fixed around the reference frame, and the latter all grouped together. The two parts are directly connected to a low voltage, 3 [Nm] brushless motor, driven by a PWM current amplifier with 20 [A] of maximum current. The bandwidth of the current amplifier is around 5 [kHz]. The angular position is read from an incremental encoder with a 20000 [ppr] resolution. A feedforward gravity compensation was implemented in order to compensate the weight of the prosthetic hand. The maximum force at the fingertip is around 30 [N], which is in the range of the maximum force that can be applied by the orthosis onto the patient's hand. As shown in Fig. 5.1, the palm and four fingers of the master device can move in a range of about 40 degrees. The master device is controlled by a PC (OS: Windows XP) and Matlab/Simulink, using a data acquisition board (Sensoray Model 626 PCI Multifunction I/O Board). The fundamental sample time is 1 [ms].

5.2.2 Slave device

The first design of the slave device was proposed in (Rosati et al., 2009). It consists in a stainless steel plate over which the patient's forearm is fixed.

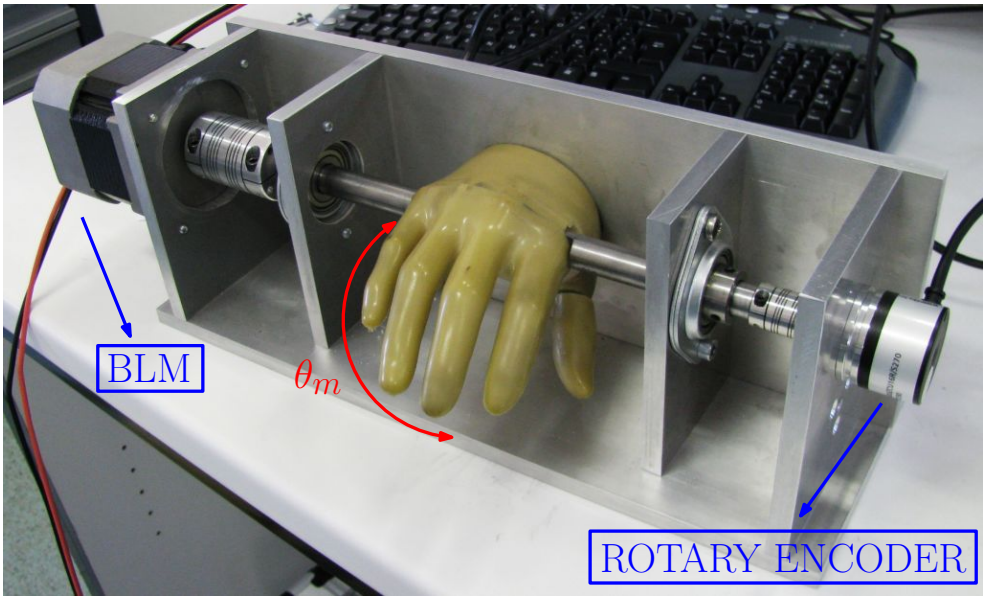


Figure 5.1: Master device.

This plate also carries out a series linear elastic actuator, a mechanism for moving the fingers, the control electronics and the power supply.

The length of the last member of the mechanism, in which four fingers can be fastened together, is conceived for fitting different hand sizes. The actuator is fixed under the base plate of the device, and two push-pull tendons are used to connect the actuator to the mechanism. The tendons are actuated by two flexible transmission cables enclosed in bent aluminum pipes. With this configuration, the maximum force at the fingertip is 30 [N], and can be reached over a maximum metacarpophalangeal (MPC) joint rotation of about 60 degrees.

The flexible transmission cables that drive the mechanism are actuated through a series elastic actuator. The main characteristic of this kind of actuator is the spring element connected in series between the transmission and the output of the actuator. The stiffness of the spring was chosen to guarantee a high compliant behavior of the actuator. Compliance has been considered a desirable feature in robotic therapy, aiding a human-robot safety interaction and preserving causal relationship between patient effort and arm movement, even if robotic assistance is provided (Krebs et al., 1998).

Since the applied force can be estimated by measuring the displacement of the spring, a good force control and a minimum impedance can be obtained, even if the transmission (which generates backlash and friction) is between the motor and the spring (Robinson et al., 1999).

Furthermore, this kind of actuators have a good tolerance to impact (for avoiding patient injuries) and high force/mass ratio. The drawback deals with the presence of the spring, that generates a lower bandwidth of the actuator (due to motor speed limitations). However, this limitation is acceptable since in robotic rehabilitation, only slowly and smoothly movements are required.

The prototype of the slave active hand orthosis is shown in Fig. 5.2a. As shown in Fig. 5.2b, a DC motor with 2000 [ppr] resolution encoder drives a miniature ball screw by a transmission belt. The ball screw nut, with 12 [mm] lead, is connected to one end of the spring series, whose opposite end moves the tendons that drives the mechanism. A linear encoder with 200 [ppmm] resolution measures the displacement of the springs as an indirect measurement of the tendons force. The DC motor is controlled by an EPOS drive in velocity mode, while the controller of the whole device is implemented using a microchip PIC microcontroller that executes the control loop at 2 [kHz]. The controller is physically interfaced to the PC by a serial port at 115.2 [kbit/s].

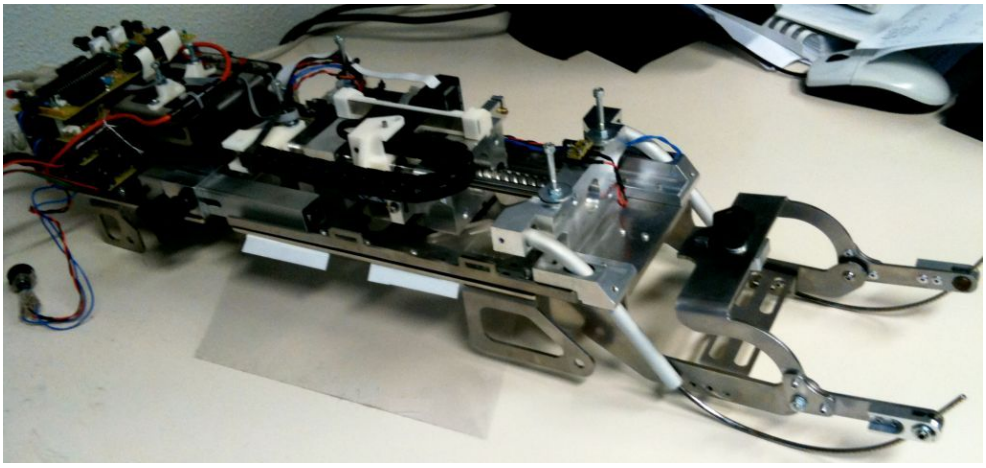
5.3 Remote rehabilitation system

5.3.1 Two-channel bilateral control system

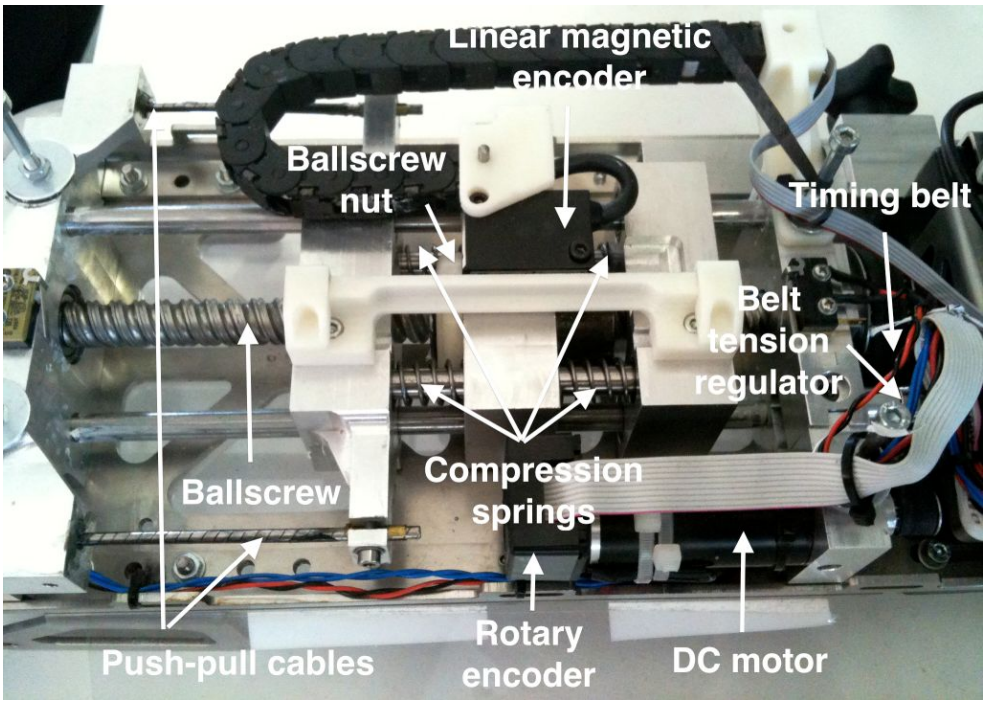
The first control architecture tested here is a two-channel bilateral master-slave scheme, position/force type (P-F), for an impedance-admittance teleoperation system, described above in Section 4.4.1. Such control architecture is illustrated in Fig. 5.3.

The master, in contact with the therapist, is an impedance device controlled by a position regulator; its reference is the delayed position of the slave. The slave, instead, in contact with the patient, is a Velocity Sourced Series Elastic Actuator (VS-SEA) that uses, as a reference, the delayed command force of the master.

The two channel control architecture achieves an stable bilateral interaction only for a limited distance between the slave station (patient's site) and the diagnostic center, as some measurements in short range connections (within 100 km) conducted in Italy, show that even a standard ADSL connection achieves 25 [ms] round-trip time (RTT), with a 400 [kbit/s] sustained



(a) Prototype.



(b) Linear flexible actuator in detail.

Figure 5.2: Hand orthosis prototype and a detailed description of the components.

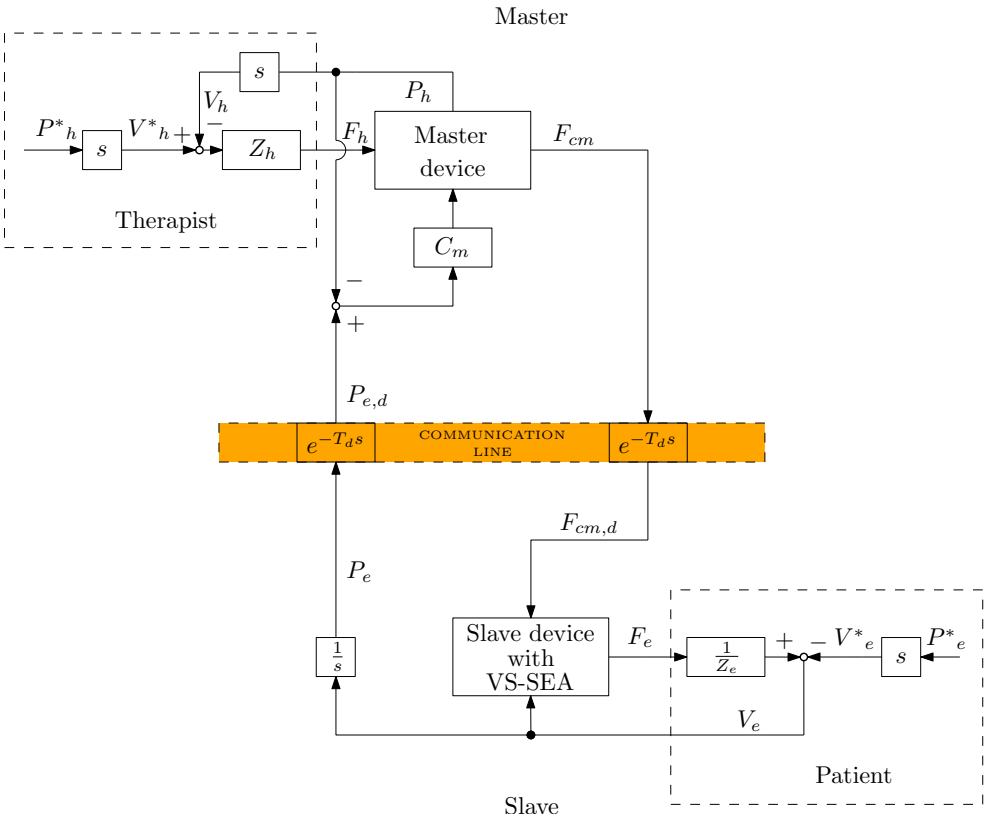


Figure 5.3: Two-channel teleoperation control system.

upload rate, which allows data sampling at several hundred hertz.

5.3.1.1 Experimental results

The proposed teleoperation system was used to test the bilateral interaction between an operator and a healthy subject (as this is still a preliminary prototype) providing a simulation of remote rehabilitation. The experimental tests were based on flexion/extension movements to evaluate the state of flaccidity/stiffness of the hand. The following tests were conducted by the proposed teleoperation system:

- Passive range of motion (ROM) test: at the beginning of the evaluating session the therapist slowly moves the patient's hand (to minimize the - potential - reflected response).
- Active ROM test: the patient is asked to move the hand until its moving limits.
- Muscular resistance test: the patient is asked to maintain the hand blocked while the the therapist tries to flexes it or extends it.
- Muscular force test: the patient is asked to flex or extend the hand while the therapist tries to keep it blocked.

With the presented teleoperation system, exercises can be reproduced by switching the patient to the master at the therapist side as well as switching the therapist to the orthosis (slave side). A possible clinical remote evaluation of the patient's muscular resistance can be performed by the master operator, who grasps the prosthetic hand as the real hand, while the slave operator extends the forearm on the upper frame of the orthosis and puts his/her fingers and his/her hand in the orthosis adaptable support, where the patient can exercise and compute the required movements, similarly to the one illustrated in the Fig. 5.4. The results of some flexo-extension tests are proposed in Fig. 5.5, Fig. 5.6, and Fig. 5.7.

5.3.1.2 Discussion of the results

In the first exercise, shown in Fig. 5.6 and Fig. 5.8a, the therapist tries to open the patient's hand. The therapist pushes up while the patient is asked to maintain his/her hand in the current position. This exercise simulates a possible remote evaluation of the muscular resistance or the stiffness of the

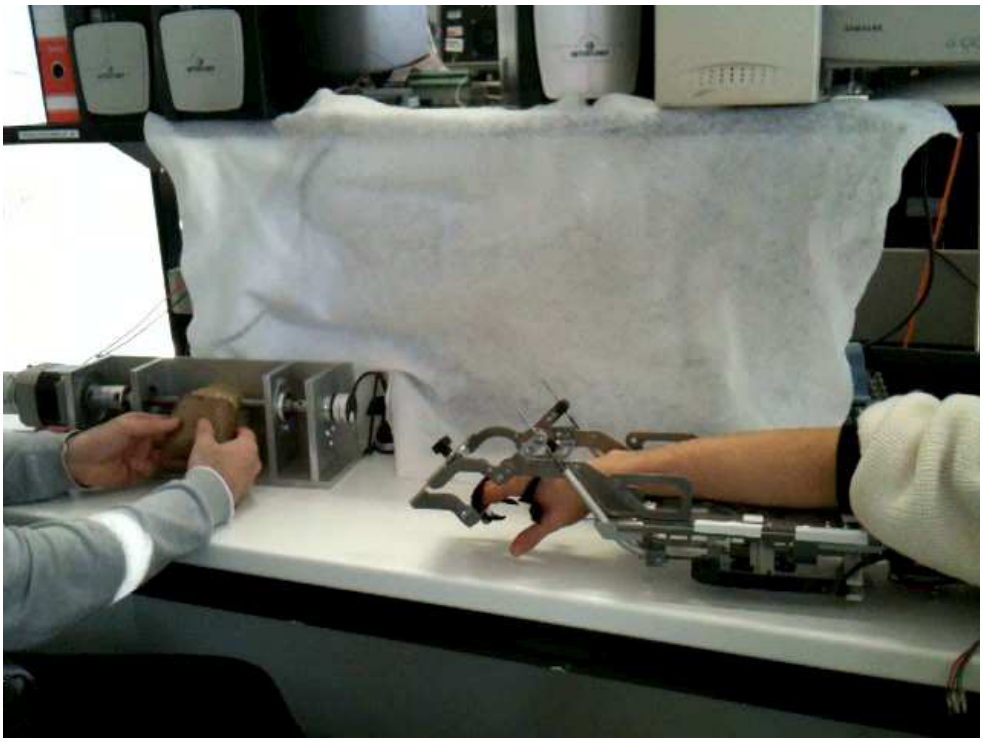
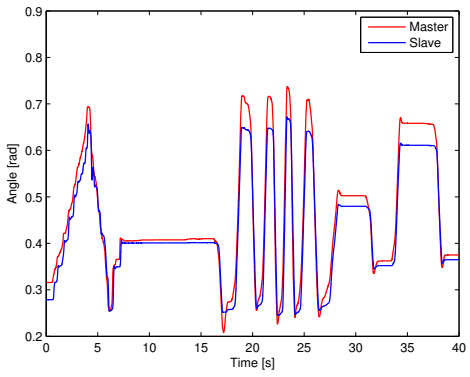
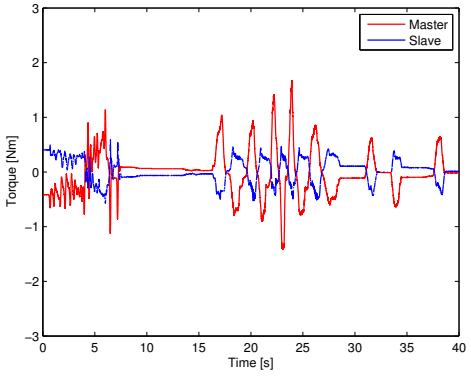


Figure 5.4: Simulation scenario in which the therapist extends the patient's hand

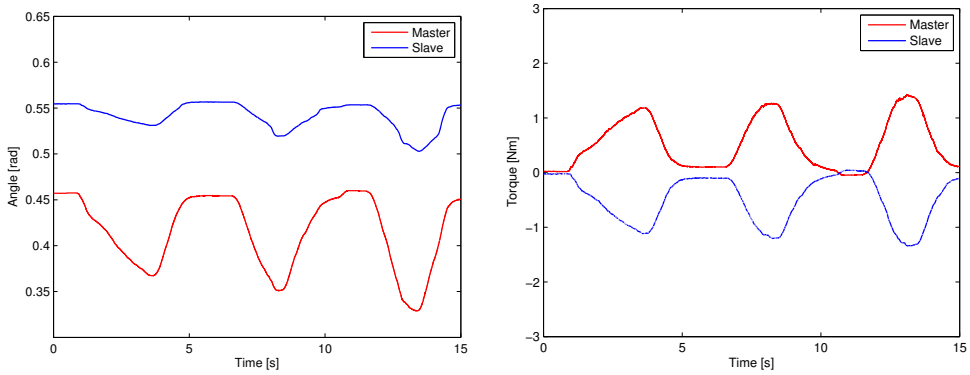


(a) Position response.



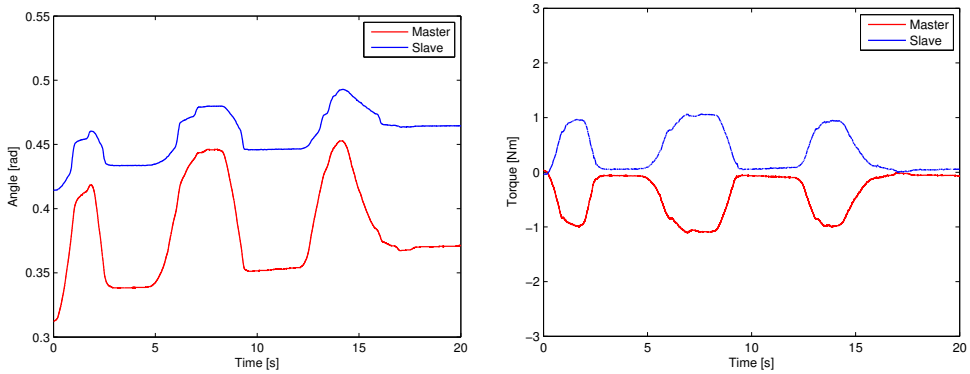
(b) Force response.

Figure 5.5: Free motion.



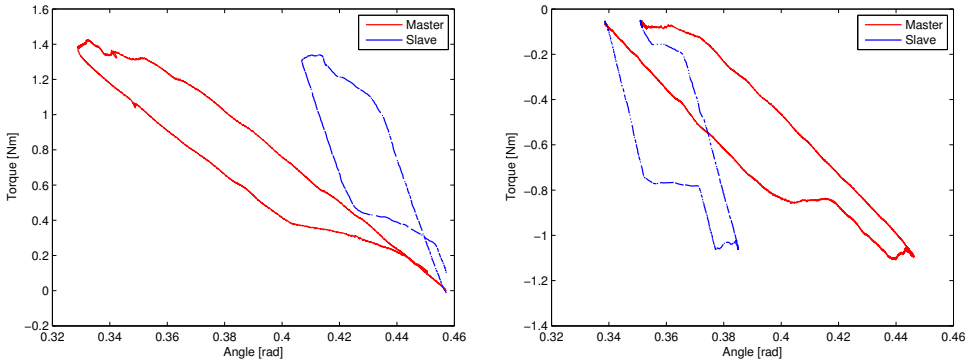
(a) Position response. (b) Force response.

Figure 5.6: The therapist extending and feeling the impedance of patient's hand.



(a) Position response. (b) Force response.

Figure 5.7: The therapist flexing and feeling the impedance of patient's hand.



(a) Extension. (b) Flexion.

Figure 5.8: Transmitted impedance at therapist side when extending and flexing the patient's hand.

patient's hand. This means that while the therapist is trying to extend and release the patient's hand, he/she needs a fair reproduction of the force. The seconde exercise, shown in Fig. 5.7 and Fig. 5.8b, propose a similar test in which the therapist performs the opposite movement, i.e. tries to flex the patient's hand. Notice that there's a visible impedance (stiffness) that is different either in master side or slave side, one scaled from the other; this because the operators are performing different tasks in both sides of the teleoperation system.

5.3.2 Four-channel bilateral control system

The four-channel bilateral control system, based on robust acceleration control (see Appendix B.1), is implemented in virtual modal space. Force control is implemented in common mode f_c and position control is implemented in differential mode x_d, described as follows:

$$f_c = \hat{f}_m + \hat{f}_s \tag{5.1}$$

$$x_d = x_m - x_s \tag{5.2}$$

These two controls then, are implemented in acceleration dimension, as follows:

$$\begin{bmatrix} \ddot{x}_c \\ \ddot{x}_d \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \ddot{x}_m \\ \ddot{x}_s \end{bmatrix} \tag{5.3}$$

In order to achieve robustness in the bilateral system, these controls are set as follows:

$$\ddot{x}_c = 0 \tag{5.4}$$
$$\ddot{x}_d \to 0 \tag{5.5}$$

The common mode is controlled by force controller $C_f(s)$ and the differential mode is controlled by position controller $C_p(s)$, as follows:

$$\ddot{x}_c = -C_f(s)(\hat{f}_m + \hat{f}_s) \tag{5.6}$$
$$\ddot{x}_d = -C_p(s)(x_m - x_s) \tag{5.7}$$

Because these controls are orthogonally spaced, they can be achieved independently and simultaneously. Then, acceleration reference of master and slave are given by these expressions:

$$\ddot{x}_m^{ref} = -C_p(s)(x_m - x_s) - C_f(s)(\hat{f}_m + \hat{f}_s) \tag{5.8}$$
$$\ddot{x}_s^{ref} = C_p(s)(x_m - x_s) - C_f(s)(\hat{f}_m + \hat{f}_s) \tag{5.9}$$

If there's a time delay in the communication line between the master and the slave, $e^{-T_1 s}$ represents the time delay from master to slave and $e^{-T_2 s}$ represents the time delay from slave to master. In this case, acceleration references of master (5.8) and slave (5.9) change as follows:

$$\ddot{x}_m^{ref} = -C_p(s)(x_m - x_s e^{-T_2 s}) - C_f(s)(\hat{f}_m + \hat{f}_s e^{-T_2 s}) \tag{5.10}$$
$$\ddot{x}_s^{ref} = C_p(s)(x_m e^{-T_1 s} - x_s) - C_f(s)(\hat{f}_m e^{-T_1 s} + \hat{f}_s) \tag{5.11}$$

The block diagram of the bilateral teleoperation system based on robust acceleration control is depicted in Fig. 5.9.

5.3.2.1 Time delay compensation

In presence of time delay in a bilaterally controlled teleoperation system, stability and transparency are strongly affected. Time delay seriously deteriorates the performance and makes the system unstable. As time delay is always present over the Internet and is highly variable and unpredictable, a time delay

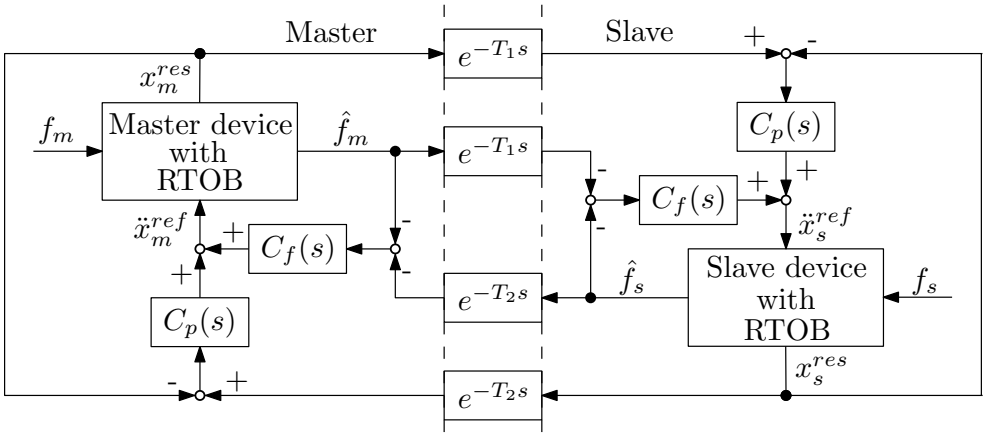


Figure 5.9: Four-channel bilateral system with time delay.

compensation method based on the concept of network disturbance (ND) and communication disturbance observer (CDOB) was implemented (see Section 4.5.5).

There are two evaluating indices based on the concept of transparency. These are reproducibility and operationality (Iida and Ohnishi, 2004). Time delay compensation by CDOB affects the reproducibility and the operationality in such a way that while operationality is improved, reproducibility is deteriorated.

When CDOB is implemented, the control diagram depicted in Fig. 5.9 becomes as Fig. 5.10. The local slave model shown in Fig. 5.10 generates a modeled acceleration reference value of the slave, which is used as an input in the CDOB. Acceleration references of master and slave change as follows:

$$\ddot{x}_m^{ref} = -C_p(s)(x_m - x_s e^{-T_2 s} - \hat{x}_{cdob}) - C_f(s)(\hat{f}_m + \hat{f}_s e^{-T_2 s}) \tag{5.12}$$

$$\ddot{x}_s^{ref} = C_p(s)(x_m e^{-T_1 s} - x_s) - C_f(s)(\hat{f}_m e^{-T_1 s} + \hat{f}_s) \tag{5.13}$$

The compensating value $\hat{x}_{cdob}$, shown in Eq. 5.12, is derived from the estimated ND by CDOB:

$$\hat{x}_{cdob} = \frac{g_{net}}{s + g_{net}} x_s e^{T_1 s}(1 - e^{-Ts}) \tag{5.14}$$

If $g_{net} \to \infty$ (ideal ND estimation), (5.12) becomes:

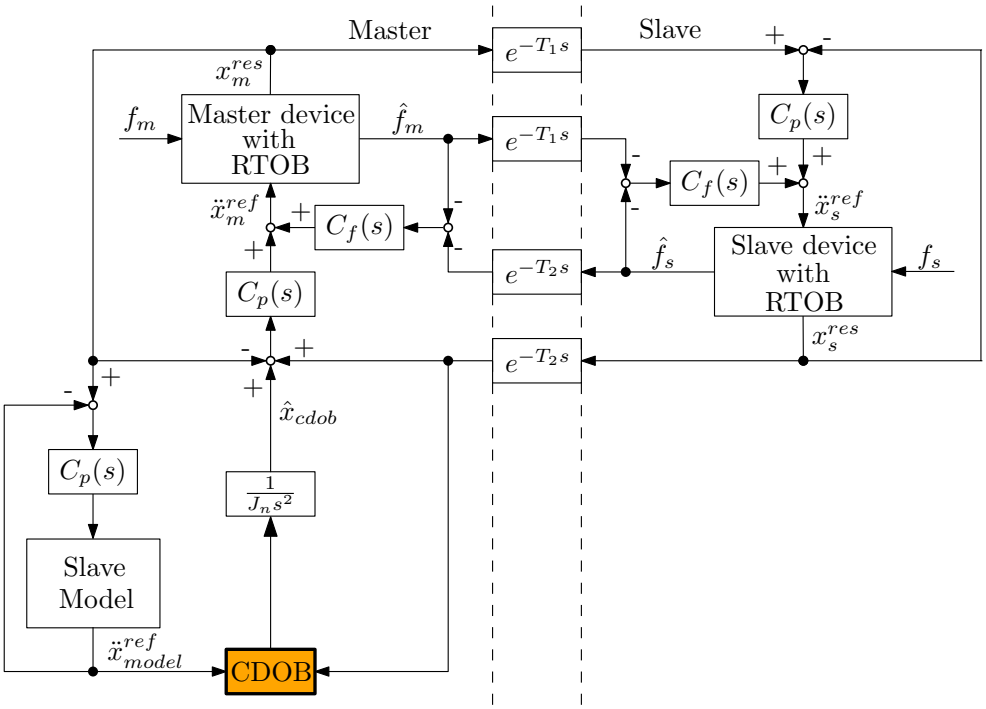


Figure 5.10: Four-channel bilateral system with CDOB.

$$\ddot{x}_m^{ref} = -C_p(s)(x_m - x_s e^{T_1 s}) - C_f(s)(\hat{f}_m + \hat{f}_s e^{-T_2 s}) \tag{5.15}$$

Position signals of master and slave can be rewritten as follows:

$$x_m e^{-T_1 s} = x_s \tag{5.16}$$

Then, Eq. 5.15 demonstrates that time delay has no effect in the position controller of the master device, since it's compensated by CDOB.

5.3.2.2 Scaling down compensation value

In order to guarantee both good operationality and reproducibility, a scaling down compensation value of CDOB is implemented only when there's a contact motion with the environment (Suzuki and Ohnishi, 2010). Therefore, the scaling factor α depends on the estimated reaction torque of the slave device $\hat{f}_s$ [Nm], as follows:

$$\alpha(\hat{f}_s) = \begin{cases} 1 & \text{if} \quad 0 \leq |\hat{f}_s| \leq f_1 \\ \frac{-1}{f_2 - f_1}\hat{f}_s + \frac{f_2}{f_2 - f_1} & \text{if} \quad f_1 < |\hat{f}_s| < f_2 \\ 0 & \text{if} \quad f_2 \leq |\hat{f}_s| \end{cases} \tag{5.17}$$

The value of f_1 is the highest value of friction in the corresponding slave device in free motion conditions, while the value of f_2 is determined by the impedance of the environment. For the presented application, this case deals with the degree of spasticity of the patient's hand. Therefore, f_2 has to be estimated depending on the conditions of each single patient.

The new bilateral control system, with the implementation of the scaling down compensation value, is shown in Fig. 5.11.

5.3.2.3 Experimental results and discussion

The experimental set-up consists in a bilateral teleoperation system based on master and slave device described in Section 5.2.1 and Section 5.2.2, respectively. As with the two channel bilateral control system, the tests were conducted between an operator (therapist) and a healthy subject (patient).

The proposed bilateral control system achieved stability, allowing a safety interaction between the therapist and the patient. Four-channel controller with CDOB is compared with the performance of 4ch controller with CDOB and SDCV under contact motion.

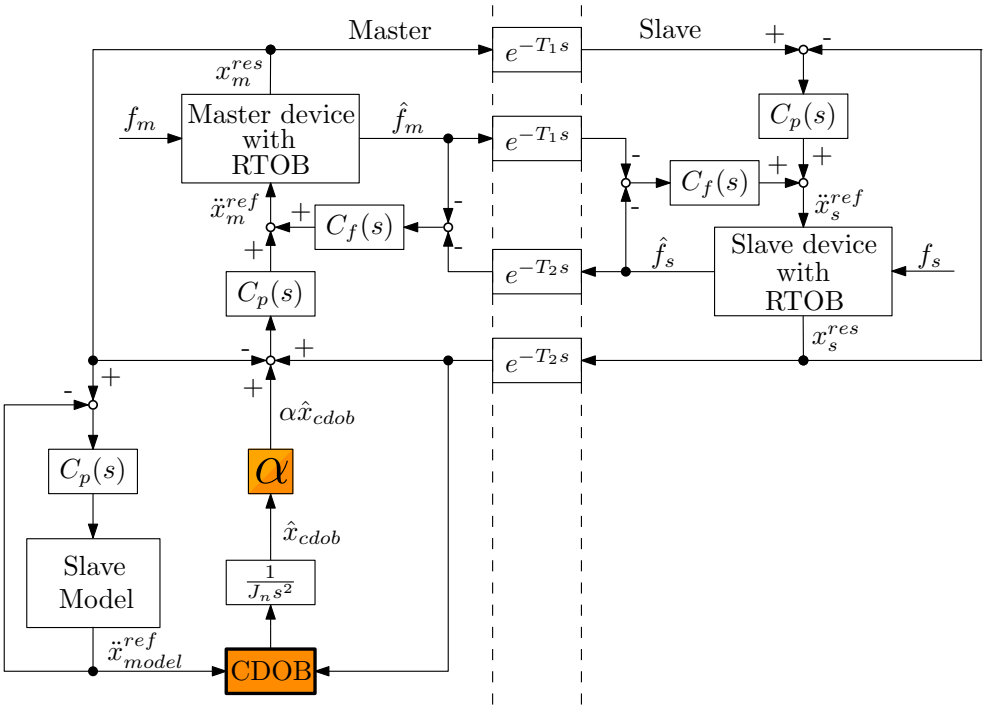
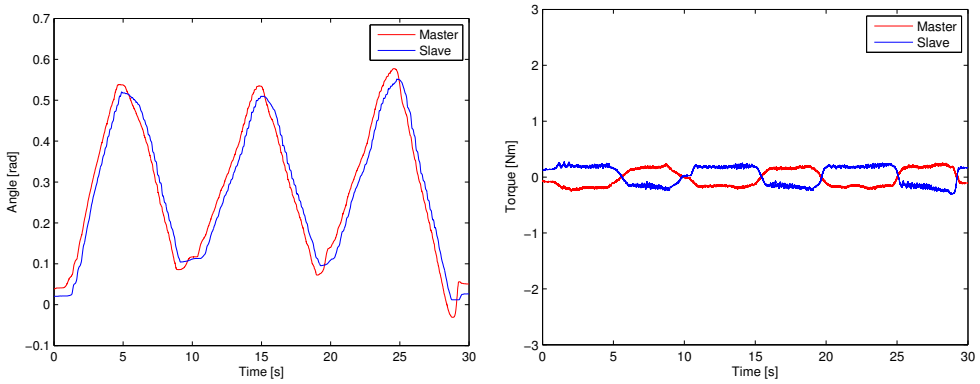


Figure 5.11: Bilateral teleoperation system with time delay based on robust acceleration control with CDOB and SDCV.



(a) Position response.

(b) Force response.

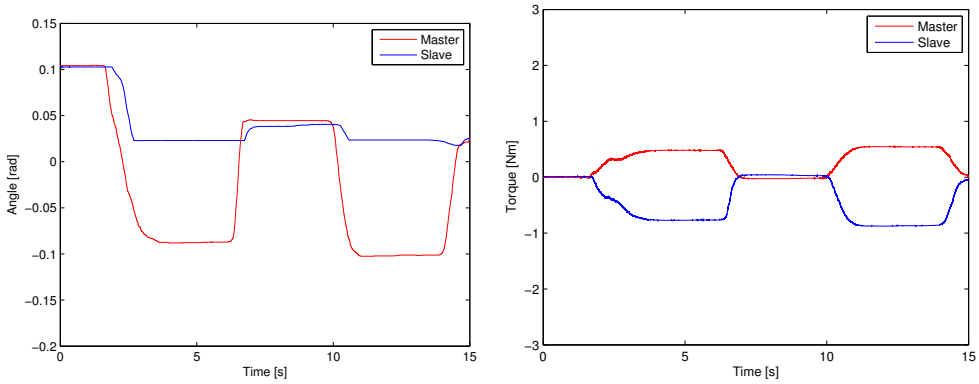
Figure 5.12: Free motion (4ch + CDOB).

The 4ch bilateral controller was designed with a position controller $C_p(s) = K_d s + K_p$, and a force controller $C_f(s) = K_f$, where K_p, K_d, and K_f are position, velocity, and force gain, respectively. Velocity information is obtained by the derivation of the position responses. The force responses are obtained by reaction torque observer based on the disturbance observer (see Appendix B.2). The bandwidth of force sensing by disturbance observer is wider than the one of force sensors. A wider bandwidth guarantees the reproduction of the contact force in high impedance environments.

In Fig. 5.12 we report the experimental results in free motion. There is a good positioning tracking between master and slave device, even in presence of these two phenomena: friction and time delay in the communication line between the master device and the slave device. Hence, it's demonstrated the efficacy of the CDOB.

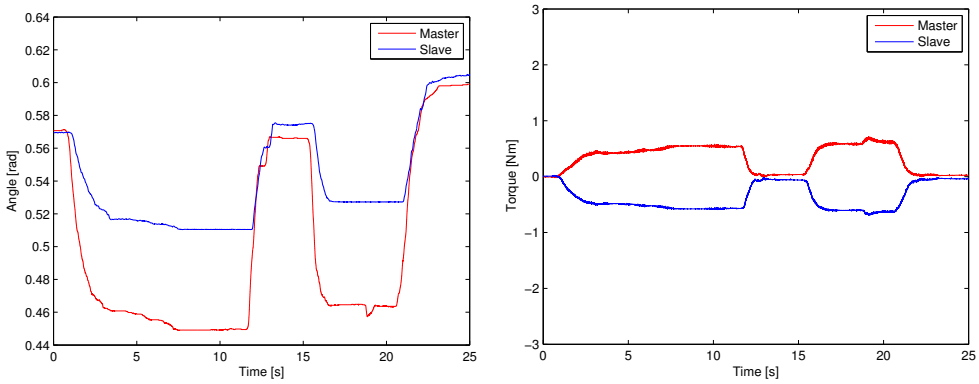
It has been demonstrated that friction compensation is a key issue in order to improve the haptic experience. However, such compensation has not been implemented in the proposed system, due to the difficulty in canceling highly nonlinear friction, mainly caused by the push-pull cables in the hand orthosis. This leads to a hysteresis phenomenon that can be observed in Fig. 5.15 and Fig. 5.18.

We report some results in which the therapist tries to open (Fig. 5.13 and Fig. 5.14) and close (Fig. 5.16 and Fig. 5.17) the patient's hand. The experimental results demonstrate the efficiency of the proposed method, in which the compensation value of CDOB is scaled down, as proposed in Eq. 5.17.



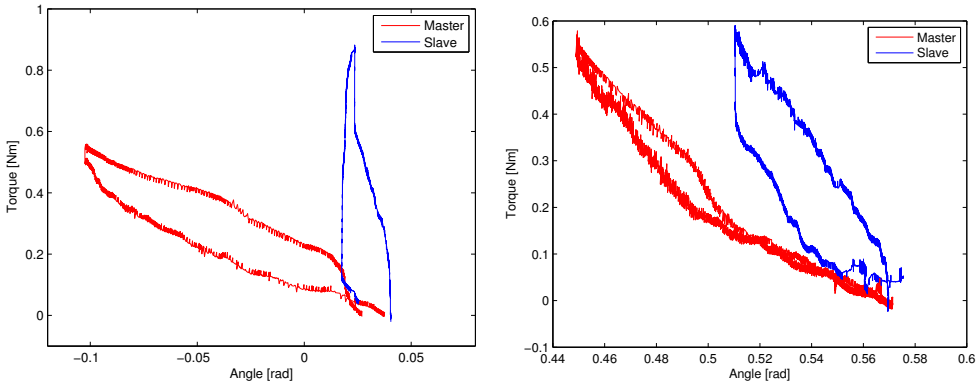
(a) Position response. (b) Force response.

Figure 5.13: Contact motion (4ch + CDOB) - Therapist opening and feeling the impedance of the patient's hand.



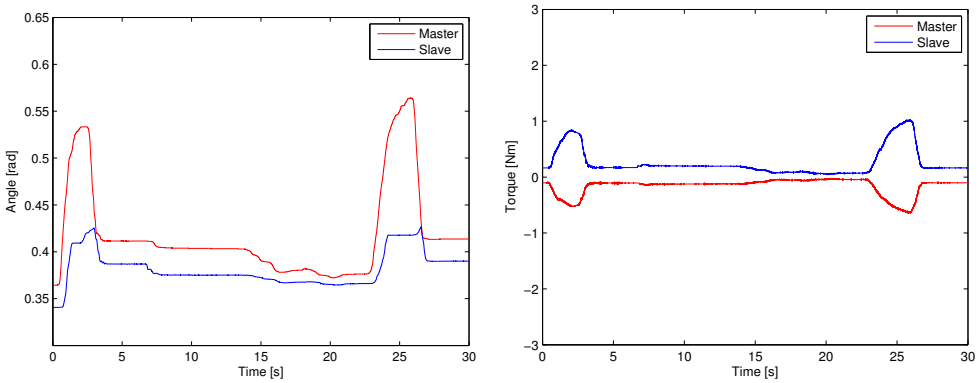
(a) Position response. (b) Force response.

Figure 5.14: Contact motion (4ch + CDOB and SDCV) - Therapist opening and feeling the impedance of the patient's hand.



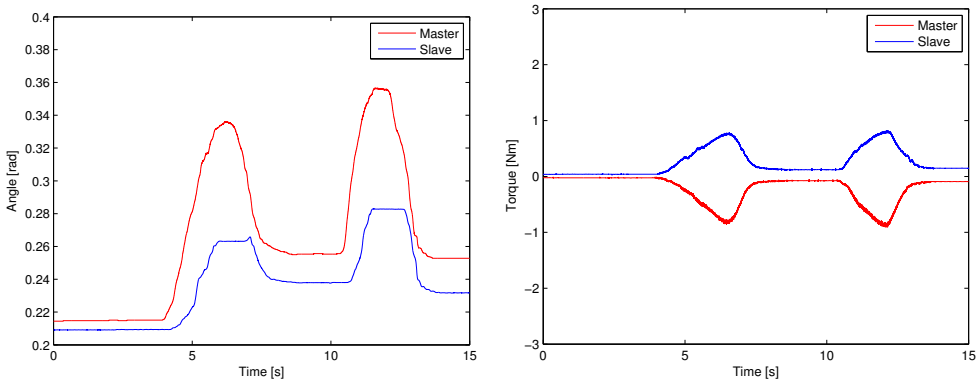
(a) (4ch + CDOB). (b) (4ch + CDOB and SDCV).

Figure 5.15: Comparing (4ch + CDOB) and (4ch + CDOB and SDCV) - Therapist opening and feeling the impedance of the patient's hand.



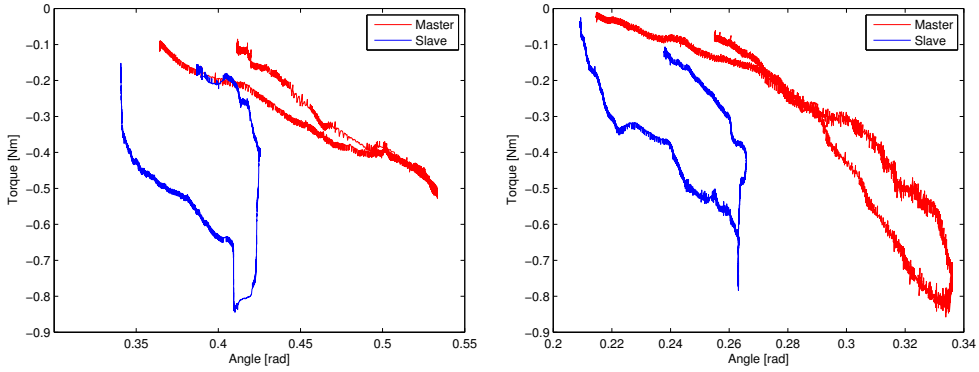
(a) Position response. (b) Force response.

Figure 5.16: Contact motion (4ch + CDOB) - Therapist closing and feeling the impedance of the patient's hand.



(a) Position response. (b) Force response.

Figure 5.17: Contact motion (4ch + CDOB and SDCV) - Therapist closing and feeling the impedance of the patient's hand.



(a) (4ch + CDOB). (b) (4ch + CDOB and SDCV).

Figure 5.18: Comparing (4ch + CDOB) and (4ch + CDOB and SDCV) - Therapist closing and feeling the impedance of the patient's hand.

It improves the performance of the bilateral interaction, while the sense of touch at the master side is clearly better. Indeed, comparing Fig. 5.15a with Fig. 5.15b, and Fig. 5.18a with Fig. 5.18b, it can be said that transparency was dramatically improved, as the force perceived in master and slave is essentially the same, although the impedance perceived at the master side is a scaled version of the actual one. Hence, it's shown that CDOB with scaling down compensation value improves the reproduced force at the therapist's side.

Some experimental parameters are available in Table 5.1.

Parameter	Description	Value
T	Time delay (one way)	100 [ms]
g_{dis}	Cutoff frequency of RTOB	1000 [rad/s]
g_{net}	Cutoff frequency of CDOB	1000 [rad/s]
f_1	Starting to scale down	0.35 [Nm]
f_2	Finishing to scale down	0.75 [Nm]
st	Sample time	1 [ms]

Table 5.1: Experimental parameters.

Chapter 6

Conclusions and future work

6.1 Conclusions

A general purpose framework, completely developed in Matlab/Simulink using the Handshake proSENSE Toolbox, was successfully implemented for two haptic interfaces. One for finger/hand rehabilitation exercises, and the other for grasping operations. However, several other data acquisition boards and devices have been successfully integrated in the proposed general purpose framework.

A five bar linkage haptic interface was used in a rehabilitation program for post-stroke patients and patients with musculoskeletal disorders for finger and hand exercises. This haptic interface, besides giving haptic sensations and kinesthetic feedback to the patients, was a powerful tool to precisely measure important parameters (e.g. position, velocity, acceleration and jerk) that are used by doctors and therapist to fairly evaluate the progresses of the finger/hand functionality of their patients.

As a Matlab/Simulink based prototype, it can support online modifications during the exercises. This property allows the therapist to exercise with a patient in an easy way, as he/she can change the visual and haptic properties of the virtual objects the patient is interacting with. Besides, the developed framework supports online data storage and retrieval during the rehabilitation sessions.

Therefore, the actual application can highly contribute for better opportunities in improving the motor relearning of, not only post-stroke patients but also patients with musculoskeletal diseases, granting an excellent motor learning perspective. Hence, this type of application may grant new oppor-

tunities for creating more efficient treatments and different kind of therapies, increasing the possibilities of full or partial recovery of post-stroke patients.

Remote rehabilitation experiments were proposed by using and comparing two control architectures in order to guarantee a bilateral and real-time haptic interaction. The system consisted in an Internet connected pair of orthosis, one applied to the patient's hand, the other moved by the therapist.

As an initial approach, a two channel bilateral control system architecture has been implemented, in order to achieve transparency and stability, even in presence of small network delays. The proposed control system was successfully tested in a bilateral haptic interaction for remote motor evaluation; hence, it can to be a reliable framework for an effective tele-rehabilitation and tele-evaluation system.

A more complete analysis was carried out by proposing a four-channel bilateral control system architecture based on robust acceleration control by disturbance observer, with CDOB for time delay compensation and SDCV of CDOB in order to achieve both good opertionality and reproducibility. The proposed control system was tested successfully in a bilateral haptic interaction, simulating a remote motor and functional evaluation of hand in patients with neurological impairments. The system was designed in order to guarantee a safe and a stable interaction, even in presence of important network delays, as the ones of Internet. The proposed control architecture consistently improved the performance of the system, conceiving a reliable framework for a more realistic and effective remote rehabilitation system.

6.2 Future work

The five bar linkage haptic interface can be further improved by performing the following modifications and updates:

- Spherical magnetic joint: Implement the device with a spherical magnetic joint at the fingertip, in order to increase safety, especially when unexpected and higher forces can take place.

- VR-engine: The actual VR-engine is based on Handshake proSENSE Toolbox, and the virtual environments are based on VRML files. This can be undesirable, as there can be other potentialities using other types of programs, besides being tied up always to the same program. Hence, the system will be integrated by another VR-engine.

- GUI: The update of the GUI will allow greater flexibility, interactivity and modularity in implementing and interacting with the virtual environments. In addition, the GUI will provide a better support in the management of the data, with higher capabilities of storing and reporting.

- Passive movements: It is considered important to assist with passive motion, by which the therapist can guide the patient at the beginning of the rehabilitation exercises, by then letting him/her gradually operate in a more autonomous way.

Another activity to be done is the implementation of a multi-finger haptic interface, in which the device will be developed with thrust wires (proposed in Section 3.4) for transmitting the motion, and with linear and/or rotary actuators. The specific choice of using one type of motor over the other, will be analyzed as well.

With respect to remote rehabilitation, the developed application was implemented only as a preliminary evaluation of the system, in order to exercise (in the future) with real patients, performing a remote assessment of their conditions. Such master/slave system will be further improved, especially the hardware.

Appendix A

General framework implementation

For the five-bar linkage haptic interface, the communication with the board as well as the forward kinematics and the calculation of the jacobian matrix were programmed in C code and implemented as S-functions.

The virtual environment was developed in V-Realm Builder 2.0. The resulting VRML file that describes the virtual world is loaded inside the HVR World block (Handshake proSENSE Toolbox) which provides haptic and graphic rendering. A spring-damper model was used to describe the haptic properties of the objects inside the virtual environment.

A.1 Data acquisition board interface

The communication with the Sensoray S626 board was implemented by an S-function as follows:

1. Set the configuration parameters of the Simulink environment. We set the communication in external mode, using a fixed step discrete solver, with a fundamental sampling time of 0.001 [sec] to achieve a real-time application.

2. S-function builder block (Fig. A.1): Set three input ports and two output ports, declaring the data type attributes and dimensions. One input port is for enable or disable the block. The other two inputs are the command signals that goes to the DAC channel. The two output

ports correspond to the encoder signal that the board reads. Once you run the application, it generates the following source files:

(a) sfun.c: This file contains the C source code representation of the standard portions of the generated S-function. The resulting function is reported in Subsection A.1.1.

(b) sfun_wrapper.c: This file contains the custom code entered in the S-function builder.

(c) sfun.tlc: This allows Real-Time Workshop to include this S-function in the code it generates (see Subsection A.1.3).

3. Modify the sfun_wrapper.c file: Enter the object file that has all the necessary dll functions that are needed to implement the real-time communication, as well as the header files. Enter the C code that retrieves the encoder signals and sends the command signals to the DAC. The resulting file is reported in Subsection A.1.2.

4. Under Real-Time Workshop incorporate the source files, object files and header files that are shipped with the product into the custom code pane (see Fig. A.2). Header files, as well as the source files, which in this case correspond to the generated S-function wrapper file must be entered as well (see Fig. A.3). We used the initialize function to initialize the board, in which the DLL library must be open, and so the board, as well as setting the acquisition parameters of the counter and DAC channels (Fig. A.4). To terminate the operations once the simulation is finished, we used the terminate function to reset the DAC channels, unlink the DLL and break off the driver's communication between the application program and board hardware (Fig. A.5).

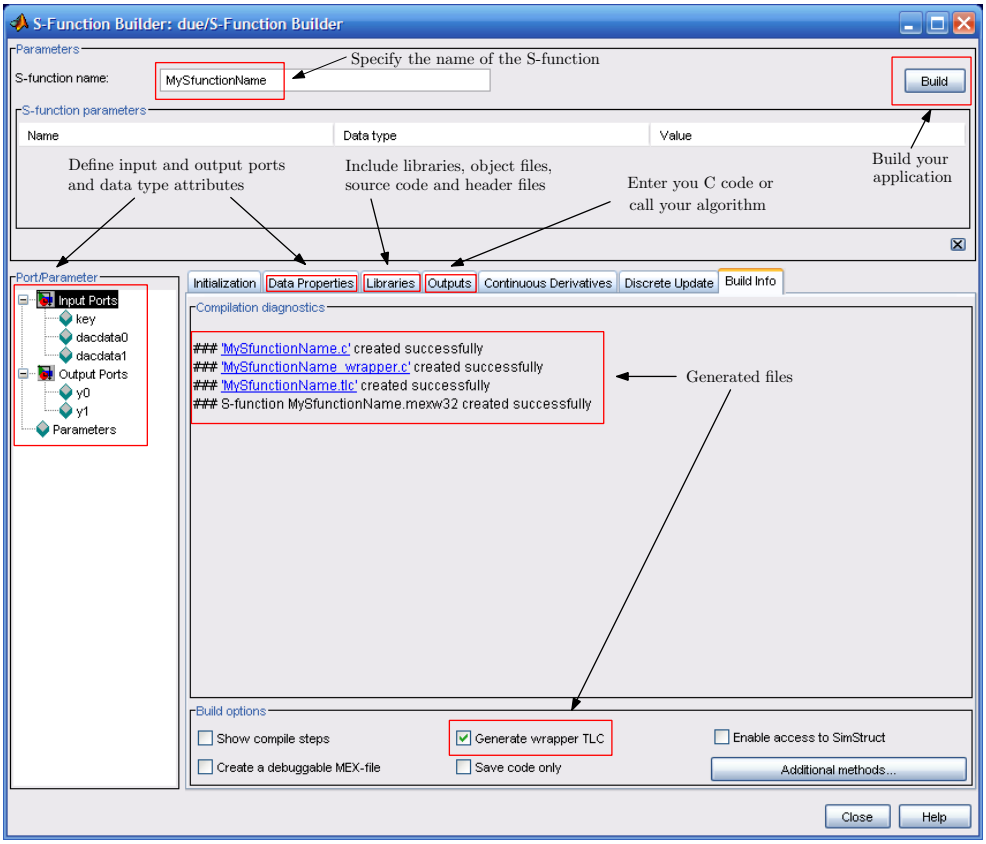


Figure A.1: How to create an S-function. These are the fields you need to fill in order to obtain the necessary files to run a real-time application.

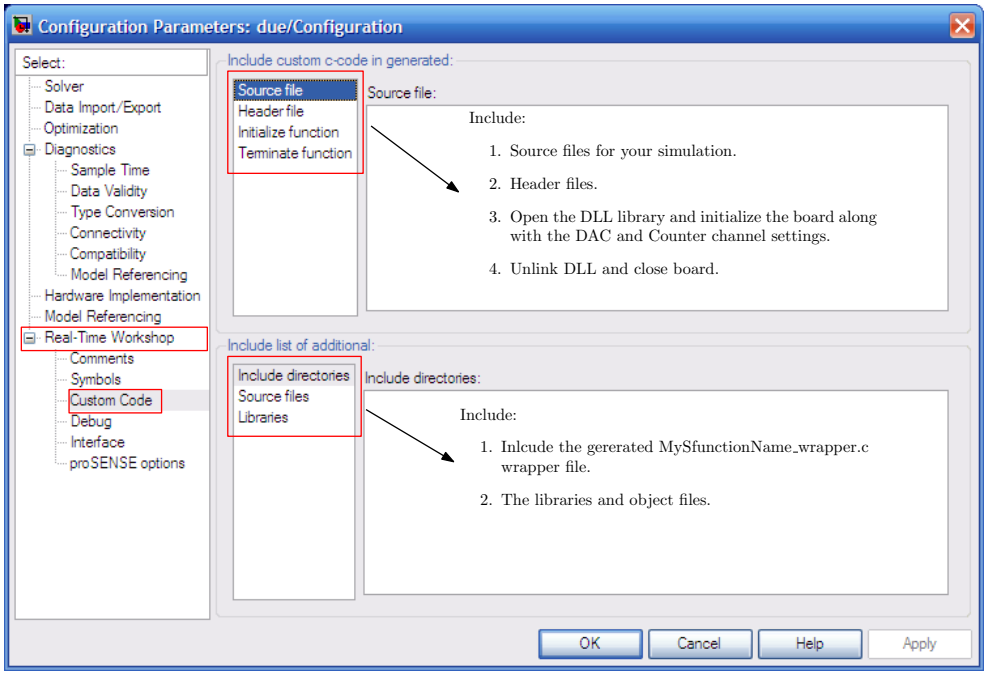


Figure A.2: How to configure an S-function inside Simulink. These are the fields you need to fill in order to compile the code.

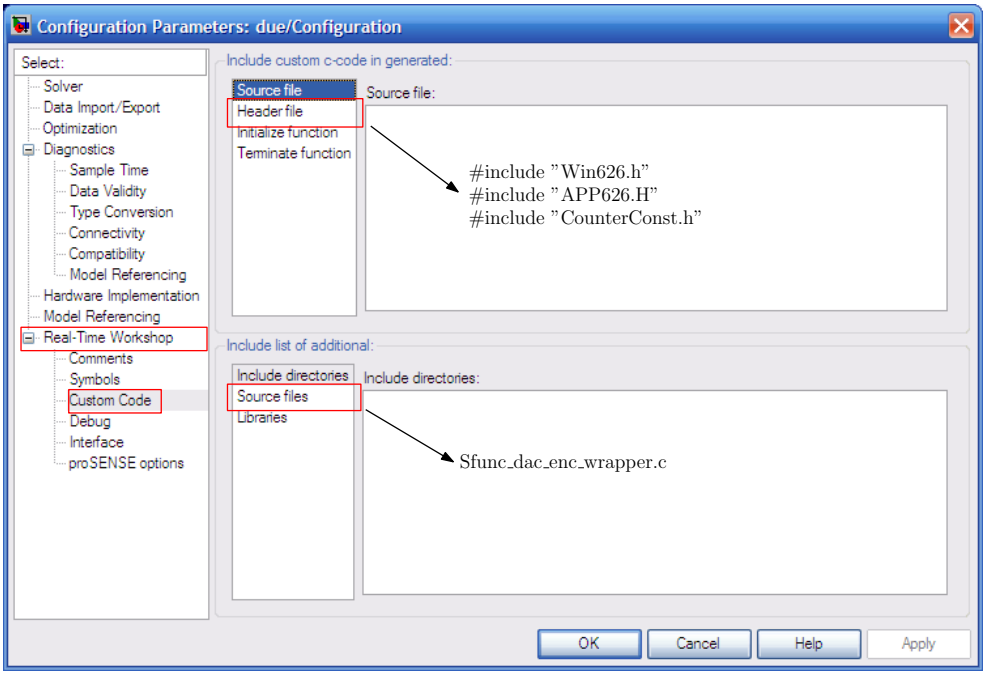
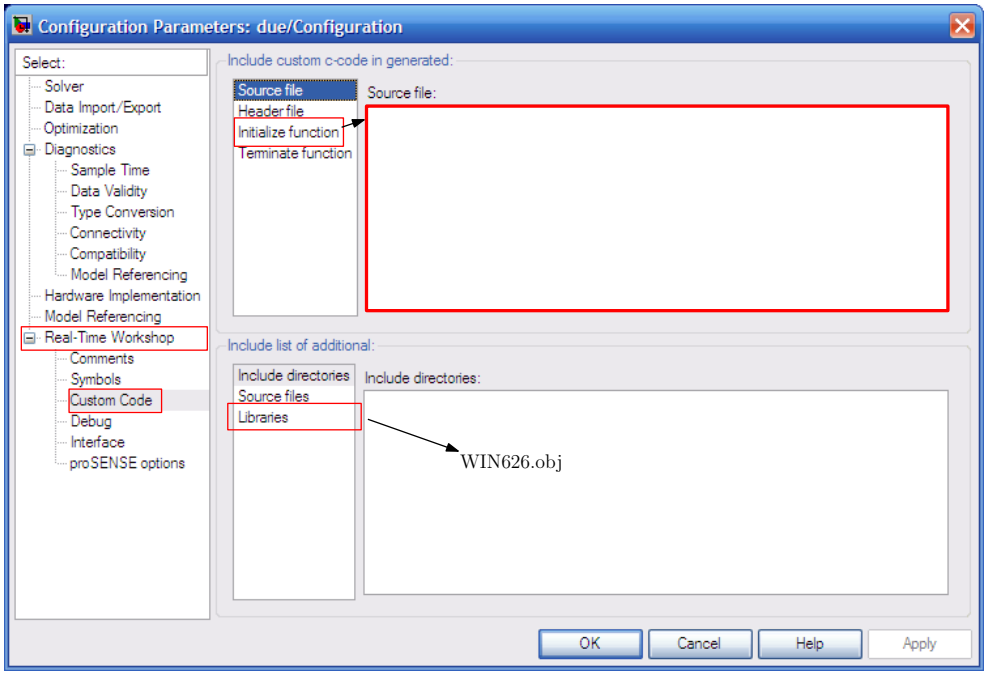


Figure A.3: Include header and source files in the custom code pane.




```
S626_DLLOpen();
S626_OpenBoard(0, 0, 0, 0);

S626_CounterModeSet( 0, CNTR_0A,
( LOADSRC_INDX ≪ BF_LOADSRC ) |
( INDXSRC_HARD ≪ BF_INDXSRC ) |
( INDXPOL_POS ≪ BF_INDXPOL ) |
( CLKSRC_COUNTER ≪ BF_CLKSRC ) |
( CLKPOL_POS ≪ BF_CLKPOL ) |
( CLKMULT_4X ≪ BF_CLKMULT ) |
( CLKENAB_ALWAYS ≪ BF_CLKENAB ) );

S626_CounterPreload( 0, CNTR_0A, 0 );

S626_CounterSoftIndex(0, CNTR_0A);

S626_CounterLatchSourceSet( 0, CNTR_0A,
LATCHSRC_AB_READ );

S626_CounterIntSourceSet( 0, CNTR_0A,
INTSRC_NONE );

S626_CounterModeSet( 0, CNTR_1A,
( LOADSRC_INDX ≪ BF_LOADSRC ) |
( INDXSRC_HARD ≪ BF_INDXSRC ) |
( INDXPOL_POS ≪ BF_INDXPOL ) |
( CLKSRC_COUNTER ≪ BF_CLKSRC ) |
( CLKPOL_POS ≪ BF_CLKPOL ) |
( CLKMULT_4X ≪ BF_CLKMULT ) |
( CLKENAB_ALWAYS ≪ BF_CLKENAB ) );

S626_CounterPreload( 0, CNTR_1A, 0 );

S626_CounterSoftIndex(0, CNTR_1A);

S626_CounterLatchSourceSet( 0, CNTR_1A,
LATCHSRC_AB_READ );

S626_CounterIntSourceSet( 0, CNTR_1A,
INTSRC_NONE );
```

Figure A.4: Board initialization with counters and DAC channels settings.

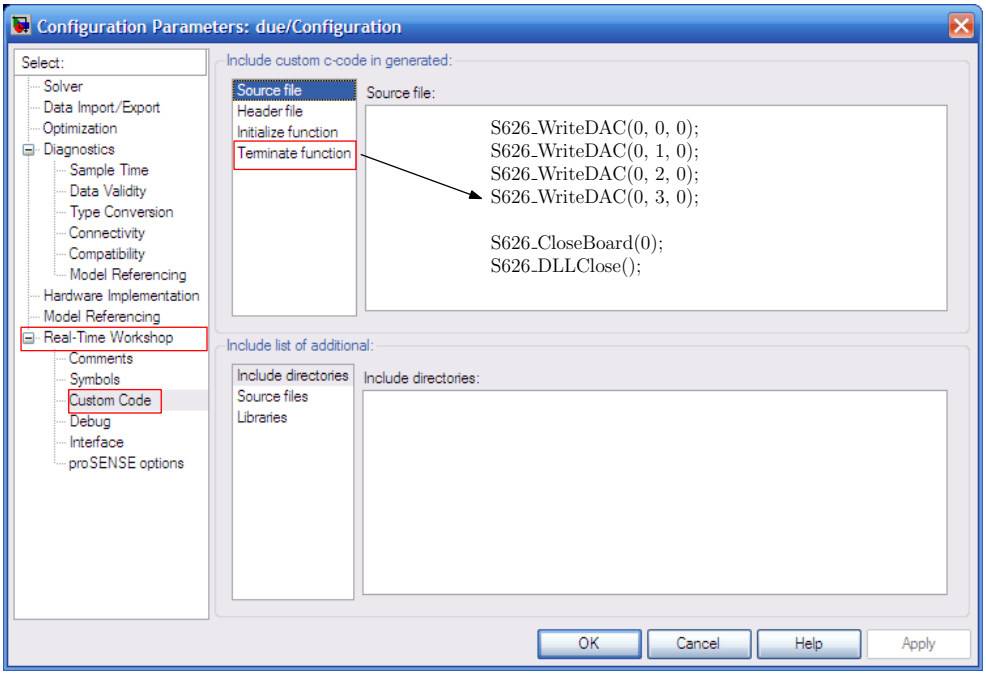


Figure A.5: Terminate the communication and reset DAC channels.

A.1.1 S-function for I/O communication

```
#define S_FUNCTION_NAME Sfunc_dac_enc
#define S_FUNCTION_LEVEL 2
/*<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<*/
/* %%%-SFUNWIZ_defines_Changes_BEGIN --- EDIT HERE TO _END */
#define NUM_INPUTS          3
/* Input Port  0 */
#define IN_PORT_0_NAME      key
#define INPUT_0_WIDTH       1
#define INPUT_DIMS_0_COL    1
#define INPUT_0_DTYPE       real_T
#define INPUT_0_COMPLEX     COMPLEX_NO
#define IN_0_FRAME_BASED    FRAME_NO
#define IN_0_DIMS           1-D
#define INPUT_0_FEEDTHROUGH 1
#define IN_0_ISSIGNED        0
#define IN_0_WORDLENGTH      8
#define IN_0_FIXPOINTSCALING 1
#define IN_0_FRACTIONLENGTH  9
#define IN_0_BIAS            0
#define IN_0_SLOPE           0.125
/* Input Port  1 */
#define IN_PORT_1_NAME      dacdata0
#define INPUT_1_WIDTH       1
#define INPUT_DIMS_1_COL    1
#define INPUT_1_DTYPE       int32_T
#define INPUT_1_COMPLEX     COMPLEX_NO
#define IN_1_FRAME_BASED    FRAME_NO
#define IN_1_DIMS           1-D
#define INPUT_1_FEEDTHROUGH 1
#define IN_1_ISSIGNED        0
#define IN_1_WORDLENGTH      8
#define IN_1_FIXPOINTSCALING 1
#define IN_1_FRACTIONLENGTH  9
#define IN_1_BIAS            0
#define IN_1_SLOPE           0.125
/* Input Port  2 */
#define IN_PORT_2_NAME      dacdata1
#define INPUT_2_WIDTH       1
#define INPUT_DIMS_2_COL    1
#define INPUT_2_DTYPE       int32_T
#define INPUT_2_COMPLEX     COMPLEX_NO
#define IN_2_FRAME_BASED    FRAME_NO
#define IN_2_DIMS           1-D
```

```
#define INPUT_2_FEEDTHROUGH 1
#define IN_2_ISSIGNED        0
#define IN_2_WORDLENGTH      8
#define IN_2_FIXPOINTSCALING 1
#define IN_2_FRACTIONLENGTH  9
#define IN_2_BIAS            0
#define IN_2_SLOPE           0.125

#define NUM_OUTPUTS          2
/* Output Port  0 */
#define OUT_PORT_0_NAME      y0
#define OUTPUT_0_WIDTH       1
#define OUTPUT_DIMS_0_COL    1
#define OUTPUT_0_DTYPE       int32_T
#define OUTPUT_0_COMPLEX     COMPLEX_NO
#define OUT_0_FRAME_BASED    FRAME_NO
#define OUT_0_DIMS           1-D
#define OUT_0_ISSIGNED        1
#define OUT_0_WORDLENGTH      8
#define OUT_0_FIXPOINTSCALING 1
#define OUT_0_FRACTIONLENGTH  3
#define OUT_0_BIAS            0
#define OUT_0_SLOPE           0.125
/* Output Port  1 */
#define OUT_PORT_1_NAME      y1
#define OUTPUT_1_WIDTH       1
#define OUTPUT_DIMS_1_COL    1
#define OUTPUT_1_DTYPE       int32_T
#define OUTPUT_1_COMPLEX     COMPLEX_NO
#define OUT_1_FRAME_BASED    FRAME_NO
#define OUT_1_DIMS           1-D
#define OUT_1_ISSIGNED        1
#define OUT_1_WORDLENGTH      8
#define OUT_1_FIXPOINTSCALING 1
#define OUT_1_FRACTIONLENGTH  3
#define OUT_1_BIAS            0
#define OUT_1_SLOPE           0.125

#define NPARAMS              0

#define SAMPLE_TIME_0        INHERITED_SAMPLE_TIME
#define NUM_DISC_STATES      0
#define DISC_STATES_IC       [0]
#define NUM_CONT_STATES      0
#define CONT_STATES_IC       [0]

#define SFUNWIZ_GENERATE_TLC 1
```

```
#define SOURCEFILES "__SFB____SFB____SFB__"
#define PANELINDEX           6
#define USE_SIMSTRUCT        0
#define SHOW_COMPILE_STEPS   0
#define CREATE_DEBUG_MEXFILE 0
#define SAVE_CODE_ONLY       1
#define SFUNWIZ_REVISION     3.0
/* %%%-SFUNWIZ_defines_Changes_END --- EDIT HERE TO _BEGIN */
/*<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<*/
#include "simstruc.h"

extern void Sfunc_dac_enc_Outputs_wrapper(const real_T *key,
const int32_T *dacdata0,
const int32_T *dacdata1,
int32_T *y0,
int32_T *y1);

/*====================*
 * S-function methods *
 *====================*/

/* Function: mdlInitializeSizes */

static void mdlInitializeSizes(SimStruct *S)
{
    DECL_AND_INIT_DIMSINFO(inputDimsInfo);
    DECL_AND_INIT_DIMSINFO(outputDimsInfo);
    ssSetNumSFcnParams(S, NPARAMS);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
    return; /* Parameter mismatch will be reported by Simulink */
    }

    ssSetNumContStates(S, NUM_CONT_STATES);
    ssSetNumDiscStates(S, NUM_DISC_STATES);

    if (!ssSetNumInputPorts(S, NUM_INPUTS)) return;
    /*Input Port 0 */
    ssSetInputPortWidth(S,  0, INPUT_0_WIDTH); /* */
    ssSetInputPortDataType(S, 0, SS_DOUBLE);
    ssSetInputPortComplexSignal(S,  0, INPUT_0_COMPLEX);
    ssSetInputPortDirectFeedThrough(S, 0, INPUT_0_FEEDTHROUGH);
    ssSetInputPortRequiredContiguous(S, 0, 1);
    /*direct input signal access*/

    /*Input Port 1 */
    ssSetInputPortWidth(S,  1, INPUT_1_WIDTH); /* */
    ssSetInputPortDataType(S, 1, SS_INT32);
```

```
        ssSetInputPortComplexSignal(S,  1, INPUT_1_COMPLEX);
        ssSetInputPortDirectFeedThrough(S, 1, INPUT_1_FEEDTHROUGH);
        ssSetInputPortRequiredContiguous(S, 1, 1);
        /*direct input signal access*/

        /*Input Port 2 */
        ssSetInputPortWidth(S,  2, INPUT_2_WIDTH); /* */
        ssSetInputPortDataType(S, 2, SS_INT32);
        ssSetInputPortComplexSignal(S,  2, INPUT_2_COMPLEX);
        ssSetInputPortDirectFeedThrough(S, 2, INPUT_2_FEEDTHROUGH);
        ssSetInputPortRequiredContiguous(S, 2, 1);
        /*direct input signal access*/


        if (!ssSetNumOutputPorts(S, NUM_OUTPUTS)) return;
        /* Output Port 0 */
        ssSetOutputPortWidth(S, 0, OUTPUT_0_WIDTH);
        ssSetOutputPortDataType(S, 0, SS_INT32);
        ssSetOutputPortComplexSignal(S, 0, OUTPUT_0_COMPLEX);
        /* Output Port 1 */
        ssSetOutputPortWidth(S, 1, OUTPUT_1_WIDTH);
        ssSetOutputPortDataType(S, 1, SS_INT32);
        ssSetOutputPortComplexSignal(S, 1, OUTPUT_1_COMPLEX);

        ssSetNumSampleTimes(S, 1);
        ssSetNumRWork(S, 0);
        ssSetNumIWork(S, 0);
        ssSetNumPWork(S, 0);
        ssSetNumModes(S, 0);
        ssSetNumNonsampledZCs(S, 0);

        ssSetOptions(S, (SS_OPTION_EXCEPTION_FREE_CODE |
        SS_OPTION_USE_TLC_WITH_ACCELERATOR |
        SS_OPTION_WORKS_WITH_CODE_REUSE));
    }

    # define MDL_SET_INPUT_PORT_FRAME_DATA
    static void mdlSetInputPortFrameData(SimStruct  *S,
    int_T      port,
    Frame_T    frameData)
    {
        ssSetInputPortFrameData(S, port, frameData);
    }
    /* Function: mdlInitializeSampleTimes */

    static void mdlInitializeSampleTimes(SimStruct *S)
    {
```

```
    ssSetSampleTime(S, 0, SAMPLE_TIME_0);
    ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_SET_INPUT_PORT_DATA_TYPE
static void
mdlSetInputPortDataType(SimStruct *S, int port, DTypeId dType)
{
    ssSetInputPortDataType( S, 0, dType);
}
#define MDL_SET_OUTPUT_PORT_DATA_TYPE
static void
mdlSetOutputPortDataType(SimStruct *S, int port, DTypeId dType)
{
    ssSetOutputPortDataType(S, 0, dType);
}

#define MDL_SET_DEFAULT_PORT_DATA_TYPES
static void mdlSetDefaultPortDataTypes(SimStruct *S)
{
    ssSetInputPortDataType( S, 0, SS_DOUBLE);
    ssSetOutputPortDataType(S, 0, SS_DOUBLE);
}
/* Function: mdlOutputs */

static void mdlOutputs(SimStruct *S, int_T tid)
{
const real_T *key  = (const real_T*) ssGetInputPortSignal(S,0);
const int32_T *dacdata0  =
                    (const int32_T*) ssGetInputPortSignal(S,1);
const int32_T *dacdata1  =
                    (const int32_T*) ssGetInputPortSignal(S,2);
int32_T *y0  = (int32_T *)ssGetOutputPortRealSignal(S,0);
int32_T *y1  = (int32_T *)ssGetOutputPortRealSignal(S,1);

Sfunc_dac_enc_Outputs_wrapper(key, dacdata0, dacdata1, y0, y1);
}

/* Function: mdlTerminate */

static void mdlTerminate(SimStruct *S)
{
}

#ifdef  MATLAB_MEX_FILE
#include "simulink.c"
#else
```

```
#include "cg_sfun.h"
#endif
```

A.1.2 S-function wrapper for I/O communication

```
/* Include Files */
#if defined(MATLAB_MEX_FILE)
#include "tmwtypes.h"
#include "simstruc_types.h"
#else
#include "rtwtypes.h"
#endif

#include "Win626.h"
#include "APP626.H"
#include "CounterConst.h"

#define u_width 1
#define y_width 1

/* Output functions */

void Sfunc_dac_enc_Outputs_wrapper(const real_T *key,
                          const int32_T *dacdata0,
                          const int32_T *dacdata1,
                          int32_T *y0,
                          int32_T *y1)
{
     y0[0] = S626_CounterReadLatch( 0, CNTR_0A );
     y1[0] = S626_CounterReadLatch( 0, CNTR_1A );

     if (*key == 1)

     {
         S626_WriteDAC(0, 0, dacdata0[0]);
         S626_WriteDAC(0, 1, dacdata1[0]);
     }

     else if (*key == 0 )

     {
         S626_WriteDAC(0, 0, 0);
         S626_WriteDAC(0, 1, 0);
         S626_WriteDAC(0, 2, 0);
```

```
            S626_WriteDAC(0, 3, 0);
        }
    }
```

A.1.3 TLC file for I/O communication

```
%implements  Sfunc_dac_enc "C"

%function BlockTypeSetup(block, system) Output
  %openfile externs

  extern void Sfunc_dac_enc_Outputs_wrapper(const real_T *key,
                          const int32_T *dacdata0,
                          const int32_T *dacdata1,
                          int32_T *y0,
                          int32_T *y1);
  %closefile externs
  %<LibCacheExtern(externs)>
  %%
%endfunction


%function Outputs(block, system) Output
/* S-Function "Sfunc_dac_enc_wrapper" Block: %<Name> */

  %assign pu0 = LibBlockInputSignalAddr(0, "", "", 0)
  %assign pu1 = LibBlockInputSignalAddr(1, "", "", 0)
  %assign pu2 = LibBlockInputSignalAddr(2, "", "", 0)
  %assign py0 = LibBlockOutputSignalAddr(0, "", "", 0)
  %assign py1 = LibBlockOutputSignalAddr(1, "", "", 0)
  %assign py_width = LibBlockOutputSignalWidth(0)
  %assign pu_width = LibBlockInputSignalWidth(0)

Sfunc_dac_enc_Outputs_wrapper(%<pu0>,%<pu1>,%<pu2>,%<py0>,%<py1>);

  %%
%endfunction
```

A.2 Kinematic model and transposed jacobian

The procedure for creating the S-function that implements the forward kinematics and the transposed jacobian matrix is much more simpler that the one

used for the I/O board communication. It follows these steps:

- C code: Create the C code for solving the kinematic chain of the mechanism in order to determine:
  1. θ_2 and θ_3, shown in Fig. 3.5.
  2. The end effector position (Eq. 3.2).
  3. The (transposed) Jacobian matrix (Eq. 3.4).
  4. The command torque due to the haptic interaction, with (Eq. 3.10) or without (Eq. 3.9) gravity compensation.
- S-function builder: Create an S-function from C code created before. Incorporate the object files of your compiled C code. The S-function builder will generate the same source files as before: sfun.c (Subsection A.2.1), sfun_wrapper.c (Subsection A.2.2) and sfun.tlc (Subsection A.2.3).
- Custom code settings: Use the Custom Code pane to insert the generated wrapper file, in the same way it was entered for I/O communication.

A.2.1 S-function

```
#define S_FUNCTION_NAME Calcolo_Jacobiano
#define S_FUNCTION_LEVEL 2
/*<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<*/
/* %%%-SFUNWIZ_defines_Changes_BEGIN --- EDIT HERE TO _END */
#define NUM_INPUTS          6
/* Input Port  0 */
#define IN_PORT_0_NAME      theta1
#define INPUT_0_WIDTH       1
#define INPUT_DIMS_0_COL    1
#define INPUT_0_DTYPE       real_T
#define INPUT_0_COMPLEX     COMPLEX_NO
#define IN_0_FRAME_BASED    FRAME_NO
#define IN_0_DIMS           1-D
#define INPUT_0_FEEDTHROUGH 1
#define IN_0_ISSIGNED        0
#define IN_0_WORDLENGTH      8
#define IN_0_FIXPOINTSCALING 1
#define IN_0_FRACTIONLENGTH  9
#define IN_0_BIAS            0
```

```
#define IN_0_SLOPE           0.125
/* Input Port  1 */
#define IN_PORT_1_NAME      theta4
#define INPUT_1_WIDTH       1
#define INPUT_DIMS_1_COL    1
#define INPUT_1_DTYPE       real_T
#define INPUT_1_COMPLEX     COMPLEX_NO
#define IN_1_FRAME_BASED    FRAME_NO
#define IN_1_DIMS           1-D
#define INPUT_1_FEEDTHROUGH 1
#define IN_1_ISSIGNED        0
#define IN_1_WORDLENGTH      8
#define IN_1_FIXPOINTSCALING 1
#define IN_1_FRACTIONLENGTH  9
#define IN_1_BIAS            0
#define IN_1_SLOPE           0.125
/* Input Port  2 */
#define IN_PORT_2_NAME      Fx
#define INPUT_2_WIDTH       1
#define INPUT_DIMS_2_COL    1
#define INPUT_2_DTYPE       real_T
#define INPUT_2_COMPLEX     COMPLEX_NO
#define IN_2_FRAME_BASED    FRAME_NO
#define IN_2_DIMS           1-D
#define INPUT_2_FEEDTHROUGH 1
#define IN_2_ISSIGNED        0
#define IN_2_WORDLENGTH      8
#define IN_2_FIXPOINTSCALING 1
#define IN_2_FRACTIONLENGTH  9
#define IN_2_BIAS            0
#define IN_2_SLOPE           0.125
/* Input Port  3 */
#define IN_PORT_3_NAME      Fy
#define INPUT_3_WIDTH       1
#define INPUT_DIMS_3_COL    1
#define INPUT_3_DTYPE       real_T
#define INPUT_3_COMPLEX     COMPLEX_NO
#define IN_3_FRAME_BASED    FRAME_NO
#define IN_3_DIMS           1-D
#define INPUT_3_FEEDTHROUGH 1
#define IN_3_ISSIGNED        0
#define IN_3_WORDLENGTH      8
#define IN_3_FIXPOINTSCALING 1
#define IN_3_FRACTIONLENGTH  9
#define IN_3_BIAS            0
#define IN_3_SLOPE           0.125
/* Input Port  4 */
```

```
#define IN_PORT_4_NAME      enableF
#define INPUT_4_WIDTH       1
#define INPUT_DIMS_4_COL    1
#define INPUT_4_DTYPE       real_T
#define INPUT_4_COMPLEX     COMPLEX_NO
#define IN_4_FRAME_BASED    FRAME_NO
#define IN_4_DIMS           1-D
#define INPUT_4_FEEDTHROUGH 1
#define IN_4_ISSIGNED        0
#define IN_4_WORDLENGTH      8
#define IN_4_FIXPOINTSCALING 1
#define IN_4_FRACTIONLENGTH  9
#define IN_4_BIAS            0
#define IN_4_SLOPE           0.125
/* Input Port  5 */
#define IN_PORT_5_NAME      enableP
#define INPUT_5_WIDTH       1
#define INPUT_DIMS_5_COL    1
#define INPUT_5_DTYPE       real_T
#define INPUT_5_COMPLEX     COMPLEX_NO
#define IN_5_FRAME_BASED    FRAME_NO
#define IN_5_DIMS           1-D
#define INPUT_5_FEEDTHROUGH 1
#define IN_5_ISSIGNED        0
#define IN_5_WORDLENGTH      8
#define IN_5_FIXPOINTSCALING 1
#define IN_5_FRACTIONLENGTH  9
#define IN_5_BIAS            0
#define IN_5_SLOPE           0.125

#define NUM_OUTPUTS          10
/* Output Port  0 */
#define OUT_PORT_0_NAME      X_ef
#define OUTPUT_0_WIDTH       1
#define OUTPUT_DIMS_0_COL    1
#define OUTPUT_0_DTYPE       real_T
#define OUTPUT_0_COMPLEX     COMPLEX_NO
#define OUT_0_FRAME_BASED    FRAME_NO
#define OUT_0_DIMS           1-D
#define OUT_0_ISSIGNED        1
#define OUT_0_WORDLENGTH      8
#define OUT_0_FIXPOINTSCALING 1
#define OUT_0_FRACTIONLENGTH  3
#define OUT_0_BIAS            0
#define OUT_0_SLOPE           0.125
/* Output Port  1 */
#define OUT_PORT_1_NAME      Y_ef
```

```
#define OUTPUT_1_WIDTH       1
#define OUTPUT_DIMS_1_COL    1
#define OUTPUT_1_DTYPE       real_T
#define OUTPUT_1_COMPLEX     COMPLEX_NO
#define OUT_1_FRAME_BASED    FRAME_NO
#define OUT_1_DIMS           1-D
#define OUT_1_ISSIGNED        1
#define OUT_1_WORDLENGTH      8
#define OUT_1_FIXPOINTSCALING 1
#define OUT_1_FRACTIONLENGTH  3
#define OUT_1_BIAS            0
#define OUT_1_SLOPE           0.125
/* Output Port  2 */
#define OUT_PORT_2_NAME      theta2
#define OUTPUT_2_WIDTH       1
#define OUTPUT_DIMS_2_COL    1
#define OUTPUT_2_DTYPE       real_T
#define OUTPUT_2_COMPLEX     COMPLEX_NO
#define OUT_2_FRAME_BASED    FRAME_NO
#define OUT_2_DIMS           1-D
#define OUT_2_ISSIGNED        1
#define OUT_2_WORDLENGTH      8
#define OUT_2_FIXPOINTSCALING 1
#define OUT_2_FRACTIONLENGTH  3
#define OUT_2_BIAS            0
#define OUT_2_SLOPE           0.125
/* Output Port  3 */
#define OUT_PORT_3_NAME      theta3
#define OUTPUT_3_WIDTH       1
#define OUTPUT_DIMS_3_COL    1
#define OUTPUT_3_DTYPE       real_T
#define OUTPUT_3_COMPLEX     COMPLEX_NO
#define OUT_3_FRAME_BASED    FRAME_NO
#define OUT_3_DIMS           1-D
#define OUT_3_ISSIGNED        1
#define OUT_3_WORDLENGTH      8
#define OUT_3_FIXPOINTSCALING 1
#define OUT_3_FRACTIONLENGTH  3
#define OUT_3_BIAS            0
#define OUT_3_SLOPE           0.125
/* Output Port  4 */
#define OUT_PORT_4_NAME      J
#define OUTPUT_4_WIDTH       2
#define OUTPUT_DIMS_4_COL    2
#define OUTPUT_4_DTYPE       real_T
#define OUTPUT_4_COMPLEX     COMPLEX_NO
#define OUT_4_FRAME_BASED    FRAME_NO
```

```
#define OUT_4_DIMS           2-D
#define OUT_4_ISSIGNED        1
#define OUT_4_WORDLENGTH      8
#define OUT_4_FIXPOINTSCALING 1
#define OUT_4_FRACTIONLENGTH  3
#define OUT_4_BIAS            0
#define OUT_4_SLOPE           0.125
/* Output Port  5 */
#define OUT_PORT_5_NAME      J_T
#define OUTPUT_5_WIDTH       2
#define OUTPUT_DIMS_5_COL    2
#define OUTPUT_5_DTYPE       real_T
#define OUTPUT_5_COMPLEX     COMPLEX_NO
#define OUT_5_FRAME_BASED    FRAME_NO
#define OUT_5_DIMS           2-D
#define OUT_5_ISSIGNED        1
#define OUT_5_WORDLENGTH      8
#define OUT_5_FIXPOINTSCALING 1
#define OUT_5_FRACTIONLENGTH  3
#define OUT_5_BIAS            0
#define OUT_5_SLOPE           0.125
/* Output Port  6 */
#define OUT_PORT_6_NAME      M_F
#define OUTPUT_6_WIDTH       2
#define OUTPUT_DIMS_6_COL    1
#define OUTPUT_6_DTYPE       real_T
#define OUTPUT_6_COMPLEX     COMPLEX_NO
#define OUT_6_FRAME_BASED    FRAME_NO
#define OUT_6_DIMS           2-D
#define OUT_6_ISSIGNED        1
#define OUT_6_WORDLENGTH      8
#define OUT_6_FIXPOINTSCALING 1
#define OUT_6_FRACTIONLENGTH  3
#define OUT_6_BIAS            0
#define OUT_6_SLOPE           0.125
/* Output Port  7 */
#define OUT_PORT_7_NAME      M_P
#define OUTPUT_7_WIDTH       2
#define OUTPUT_DIMS_7_COL    1
#define OUTPUT_7_DTYPE       real_T
#define OUTPUT_7_COMPLEX     COMPLEX_NO
#define OUT_7_FRAME_BASED    FRAME_NO
#define OUT_7_DIMS           2-D
#define OUT_7_ISSIGNED        1
#define OUT_7_WORDLENGTH      8
#define OUT_7_FIXPOINTSCALING 1
#define OUT_7_FRACTIONLENGTH  3
```

```
#define OUT_7_BIAS            0
#define OUT_7_SLOPE           0.125
/* Output Port  8 */
#define OUT_PORT_8_NAME      M1
#define OUTPUT_8_WIDTH       1
#define OUTPUT_DIMS_8_COL    1
#define OUTPUT_8_DTYPE       real_T
#define OUTPUT_8_COMPLEX     COMPLEX_NO
#define OUT_8_FRAME_BASED    FRAME_NO
#define OUT_8_DIMS           1-D
#define OUT_8_ISSIGNED        1
#define OUT_8_WORDLENGTH      8
#define OUT_8_FIXPOINTSCALING 1
#define OUT_8_FRACTIONLENGTH  3
#define OUT_8_BIAS            0
#define OUT_8_SLOPE           0.125
/* Output Port  9 */
#define OUT_PORT_9_NAME      M4
#define OUTPUT_9_WIDTH       1
#define OUTPUT_DIMS_9_COL    1
#define OUTPUT_9_DTYPE       real_T
#define OUTPUT_9_COMPLEX     COMPLEX_NO
#define OUT_9_FRAME_BASED    FRAME_NO
#define OUT_9_DIMS           1-D
#define OUT_9_ISSIGNED        1
#define OUT_9_WORDLENGTH      8
#define OUT_9_FIXPOINTSCALING 1
#define OUT_9_FRACTIONLENGTH  3
#define OUT_9_BIAS            0
#define OUT_9_SLOPE           0.125

#define NPARAMS              0

#define SAMPLE_TIME_0        INHERITED_SAMPLE_TIME
#define NUM_DISC_STATES      0
#define DISC_STATES_IC       [0]
#define NUM_CONT_STATES      0
#define CONT_STATES_IC       [0]

#define SFUNWIZ_GENERATE_TLC 1
#define SOURCEFILES "__SFB____"
#define PANELINDEX           6
#define USE_SIMSTRUCT        0
#define SHOW_COMPILE_STEPS   0
#define CREATE_DEBUG_MEXFILE 0
#define SAVE_CODE_ONLY       0
#define SFUNWIZ_REVISION     3.0
```

```
/* %%%-SFUNWIZ_defines_Changes_END --- EDIT HERE TO _BEGIN */
/*<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<*/
#include "simstruc.h"

extern void Calcolo_Jacobiano_Outputs_wrapper(
const real_T *theta1,
const real_T *theta4,
const real_T *Fx,
const real_T *Fy,
const real_T *enableF,
const real_T *enableP,
real_T *X_ef,
real_T *Y_ef,
real_T *theta2,
real_T *theta3,
real_T *J,
real_T *J_T,
real_T *M_F,
real_T *M_P,
real_T *M1,
real_T *M4);

/*====================*
 * S-function methods *
 *====================*/

/* Function: mdlInitializeSizes */

static void mdlInitializeSizes(SimStruct *S)
{

    DECL_AND_INIT_DIMSINFO(inputDimsInfo);
    DECL_AND_INIT_DIMSINFO(outputDimsInfo);
    ssSetNumSFcnParams(S, NPARAMS);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
    return; /* Parameter mismatch will be reported by Simulink */
    }

    ssSetNumContStates(S, NUM_CONT_STATES);
    ssSetNumDiscStates(S, NUM_DISC_STATES);

    if (!ssSetNumInputPorts(S, NUM_INPUTS)) return;
    /*Input Port 0 */
    ssSetInputPortWidth(S,  0, INPUT_0_WIDTH); /* */
    ssSetInputPortDataType(S, 0, SS_DOUBLE);
    ssSetInputPortComplexSignal(S,  0, INPUT_0_COMPLEX);
    ssSetInputPortDirectFeedThrough(S, 0, INPUT_0_FEEDTHROUGH);
```

```
    ssSetInputPortRequiredContiguous(S, 0, 1);
/*direct input signal access*/

    /*Input Port 1 */
    ssSetInputPortWidth(S,  1, INPUT_1_WIDTH); /* */
    ssSetInputPortDataType(S, 1, SS_DOUBLE);
    ssSetInputPortComplexSignal(S,  1, INPUT_1_COMPLEX);
    ssSetInputPortDirectFeedThrough(S, 1, INPUT_1_FEEDTHROUGH);
    ssSetInputPortRequiredContiguous(S, 1, 1);
    /*direct input signal access*/

    /*Input Port 2 */
    ssSetInputPortWidth(S,  2, INPUT_2_WIDTH); /* */
    ssSetInputPortDataType(S, 2, SS_DOUBLE);
    ssSetInputPortComplexSignal(S,  2, INPUT_2_COMPLEX);
    ssSetInputPortDirectFeedThrough(S, 2, INPUT_2_FEEDTHROUGH);
    ssSetInputPortRequiredContiguous(S, 2, 1);
    /*direct input signal access*/

    /*Input Port 3 */
    ssSetInputPortWidth(S,  3, INPUT_3_WIDTH); /* */
    ssSetInputPortDataType(S, 3, SS_DOUBLE);
    ssSetInputPortComplexSignal(S,  3, INPUT_3_COMPLEX);
    ssSetInputPortDirectFeedThrough(S, 3, INPUT_3_FEEDTHROUGH);
    ssSetInputPortRequiredContiguous(S, 3, 1);
    /*direct input signal access*/

    /*Input Port 4 */
    ssSetInputPortWidth(S,  4, INPUT_4_WIDTH); /* */
    ssSetInputPortDataType(S, 4, SS_DOUBLE);
    ssSetInputPortComplexSignal(S,  4, INPUT_4_COMPLEX);
    ssSetInputPortDirectFeedThrough(S, 4, INPUT_4_FEEDTHROUGH);
    ssSetInputPortRequiredContiguous(S, 4, 1);
    /*direct input signal access*/

    /*Input Port 5 */
    ssSetInputPortWidth(S,  5, INPUT_5_WIDTH); /* */
    ssSetInputPortDataType(S, 5, SS_DOUBLE);
    ssSetInputPortComplexSignal(S,  5, INPUT_5_COMPLEX);
    ssSetInputPortDirectFeedThrough(S, 5, INPUT_5_FEEDTHROUGH);
    ssSetInputPortRequiredContiguous(S, 5, 1);
    /*direct input signal access*/


    if (!ssSetNumOutputPorts(S, NUM_OUTPUTS)) return;
    /* Output Port 0 */
    ssSetOutputPortWidth(S, 0, OUTPUT_0_WIDTH);
```

```
ssSetOutputPortDataType(S, 0, SS_DOUBLE);
ssSetOutputPortComplexSignal(S, 0, OUTPUT_0_COMPLEX);
/* Output Port 1 */
ssSetOutputPortWidth(S, 1, OUTPUT_1_WIDTH);
ssSetOutputPortDataType(S, 1, SS_DOUBLE);
ssSetOutputPortComplexSignal(S, 1, OUTPUT_1_COMPLEX);
/* Output Port 2 */
ssSetOutputPortWidth(S, 2, OUTPUT_2_WIDTH);
ssSetOutputPortDataType(S, 2, SS_DOUBLE);
ssSetOutputPortComplexSignal(S, 2, OUTPUT_2_COMPLEX);
/* Output Port 3 */
ssSetOutputPortWidth(S, 3, OUTPUT_3_WIDTH);
ssSetOutputPortDataType(S, 3, SS_DOUBLE);
ssSetOutputPortComplexSignal(S, 3, OUTPUT_3_COMPLEX);
/* Output Port 4 */
outputDimsInfo.width = OUTPUT_4_WIDTH;
ssSetOutputPortDimensionInfo(S, 4, &outputDimsInfo);
ssSetOutputPortMatrixDimensions( S ,4, OUTPUT_4_WIDTH,
OUTPUT_DIMS_4_COL);
ssSetOutputPortFrameData(S, 4, OUT_4_FRAME_BASED);
ssSetOutputPortDataType(S, 4, SS_DOUBLE);
ssSetOutputPortComplexSignal(S, 4, OUTPUT_4_COMPLEX);
/* Output Port 5 */
outputDimsInfo.width = OUTPUT_5_WIDTH;
ssSetOutputPortDimensionInfo(S, 5, &outputDimsInfo);
ssSetOutputPortMatrixDimensions( S ,5, OUTPUT_5_WIDTH,
OUTPUT_DIMS_5_COL);
ssSetOutputPortFrameData(S, 5, OUT_5_FRAME_BASED);
ssSetOutputPortDataType(S, 5, SS_DOUBLE);
ssSetOutputPortComplexSignal(S, 5, OUTPUT_5_COMPLEX);
/* Output Port 6 */
outputDimsInfo.width = OUTPUT_6_WIDTH;
ssSetOutputPortDimensionInfo(S, 6, &outputDimsInfo);
ssSetOutputPortMatrixDimensions( S ,6, OUTPUT_6_WIDTH,
OUTPUT_DIMS_6_COL);
ssSetOutputPortFrameData(S, 6, OUT_6_FRAME_BASED);
ssSetOutputPortDataType(S, 6, SS_DOUBLE);
ssSetOutputPortComplexSignal(S, 6, OUTPUT_6_COMPLEX);
/* Output Port 7 */
outputDimsInfo.width = OUTPUT_7_WIDTH;
ssSetOutputPortDimensionInfo(S, 7, &outputDimsInfo);
ssSetOutputPortMatrixDimensions( S ,7, OUTPUT_7_WIDTH,
OUTPUT_DIMS_7_COL);
ssSetOutputPortFrameData(S, 7, OUT_7_FRAME_BASED);
ssSetOutputPortDataType(S, 7, SS_DOUBLE);
ssSetOutputPortComplexSignal(S, 7, OUTPUT_7_COMPLEX);
/* Output Port 8 */
```

```
    ssSetOutputPortWidth(S, 8, OUTPUT_8_WIDTH);
    ssSetOutputPortDataType(S, 8, SS_DOUBLE);
    ssSetOutputPortComplexSignal(S, 8, OUTPUT_8_COMPLEX);
    /* Output Port 9 */
    ssSetOutputPortWidth(S, 9, OUTPUT_9_WIDTH);
    ssSetOutputPortDataType(S, 9, SS_DOUBLE);
    ssSetOutputPortComplexSignal(S, 9, OUTPUT_9_COMPLEX);

    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, 0);
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);

    ssSetOptions(S, (SS_OPTION_EXCEPTION_FREE_CODE |
    SS_OPTION_USE_TLC_WITH_ACCELERATOR |
    SS_OPTION_WORKS_WITH_CODE_REUSE));
}

# define MDL_SET_INPUT_PORT_FRAME_DATA
static void mdlSetInputPortFrameData(SimStruct  *S,
int_T      port, Frame_T    frameData)
{
    ssSetInputPortFrameData(S, port, frameData);
}

/* Function: mdlInitializeSampleTimes */

static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, SAMPLE_TIME_0);
    ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_SET_INPUT_PORT_DATA_TYPE
static void mdlSetInputPortDataType(SimStruct *S, int port,
DTypeId dType)
{
    ssSetInputPortDataType( S, 0, dType);
}
#define MDL_SET_OUTPUT_PORT_DATA_TYPE
static void mdlSetOutputPortDataType(SimStruct *S, int port,
DTypeId dType)
{
    ssSetOutputPortDataType(S, 0, dType);
}
```

```
#define MDL_SET_DEFAULT_PORT_DATA_TYPES
static void mdlSetDefaultPortDataTypes(SimStruct *S)
{
    ssSetInputPortDataType( S, 0, SS_DOUBLE);
    ssSetOutputPortDataType(S, 0, SS_DOUBLE);
}

/* Function: mdlOutputs */

static void mdlOutputs(SimStruct *S, int_T tid)
{
const real_T *theta1 = (const real_T*)ssGetInputPortSignal(S,0);
const real_T *theta4 = (const real_T*)ssGetInputPortSignal(S,1);
const real_T *Fx = (const real_T*)ssGetInputPortSignal(S,2);
const real_T *Fy = (const real_T*)ssGetInputPortSignal(S,3);
const real_T *enableF = (const real_T*)ssGetInputPortSignal(S,4);
const real_T *enableP = (const real_T*)ssGetInputPortSignal(S,5);
real_T *X_ef = (real_T *)ssGetOutputPortRealSignal(S,0);
real_T *Y_ef = (real_T *)ssGetOutputPortRealSignal(S,1);
real_T *theta2 = (real_T *)ssGetOutputPortRealSignal(S,2);
real_T *theta3 = (real_T *)ssGetOutputPortRealSignal(S,3);
real_T *J = (real_T *)ssGetOutputPortRealSignal(S,4);
real_T *J_T = (real_T *)ssGetOutputPortRealSignal(S,5);
real_T *M_F = (real_T *)ssGetOutputPortRealSignal(S,6);
real_T *M_P = (real_T *)ssGetOutputPortRealSignal(S,7);
real_T *M1 = (real_T *)ssGetOutputPortRealSignal(S,8);
real_T *M4 = (real_T *)ssGetOutputPortRealSignal(S,9);

    Calcolo_Jacobiano_Outputs_wrapper(theta1, theta4, Fx, Fy,
    enableF, enableP, X_ef, Y_ef, theta2, theta3,
    J, J_T, M_F, M_P, M1, M4);
}



/* Function: mdlTerminate */

static void mdlTerminate(SimStruct *S)
{
}

#ifdef  MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

A.2.2 S-function wrapper

```
/* Include Files */

#if defined(MATLAB_MEX_FILE)
#include "tmwtypes.h"
#include "simstruc_types.h"
#else
#include "rtwtypes.h"
#endif

#include <stdio.h>
#include <iostream.h>
#include <math.h>
#include "calcolo_theta23_EF.h"
#include "calcolo_jacobiano.h"
#include "jacobiano_trasposto.h"
#include "coppie_forzeEF.h"
#include "coppie_forzePeso.h"
#include "coppia_totale.h"

#define u_width 1
#define y_width 1

 * Output functions */
void Calcolo_Jacobiano_Outputs_wrapper(const real_T *theta1,
const real_T *theta4,
const real_T *Fx,
const real_T *Fy,
const real_T *enableF,
const real_T *enableP,
real_T *X_ef,
real_T *Y_ef,
real_T *theta2,
real_T *theta3,
real_T *J,
real_T *J_T,
real_T *M_F,
real_T *M_P,
real_T *M1,
real_T *M4)
{

calcolo_theta23_EF (theta1[0], theta4[0], &theta2[0],
&theta3[0], &X_ef[0], &Y_ef[0]);
calcolo_jacobiano (theta1[0], theta2[0],
```

```
theta3[0], theta4[0], J);
jacobiano_trasposto (J, J_T);
coppie_forzeEF (enableF[0], Fx[0], Fy[0],
J_T, M_F);
coppie_forzePeso (enableP[0], theta1[0], theta2[0],
theta3[0], theta4[0], M_P);
coppia_totale (M_F, M_P, &M1[0], &M4[0]);
}
```

A.2.3 TLC file

```
%implements  Calcolo_Jacobiano "C"

%function BlockTypeSetup(block, system) Output
  %openfile externs

  extern void Calcolo_Jacobiano_Outputs_wrapper(
                          const real_T *theta1,
                          const real_T *theta4,
                          const real_T *Fx,
                          const real_T *Fy,
                          const real_T *enableF,
                          const real_T *enableP,
                          real_T *X_ef,
                          real_T *Y_ef,
                          real_T *theta2,
                          real_T *theta3,
                          real_T *J,
                          real_T *J_T,
                          real_T *M_F,
                          real_T *M_P,
                          real_T *M1,
                          real_T *M4);
  %closefile externs
  %<LibCacheExtern(externs)>
  %%
%endfunction

%function Outputs(block, system) Output
/* S-Function "Calcolo_Jacobiano_wrapper" Block: %<Name> */

  %assign pu0 = LibBlockInputSignalAddr(0, "", "", 0)
  %assign pu1 = LibBlockInputSignalAddr(1, "", "", 0)
  %assign pu2 = LibBlockInputSignalAddr(2, "", "", 0)
```

```
    %assign pu3 = LibBlockInputSignalAddr(3, "", "", 0)
    %assign pu4 = LibBlockInputSignalAddr(4, "", "", 0)
    %assign pu5 = LibBlockInputSignalAddr(5, "", "", 0)
    %assign py0 = LibBlockOutputSignalAddr(0, "", "", 0)
    %assign py1 = LibBlockOutputSignalAddr(1, "", "", 0)
    %assign py2 = LibBlockOutputSignalAddr(2, "", "", 0)
    %assign py3 = LibBlockOutputSignalAddr(3, "", "", 0)
    %assign py4 = LibBlockOutputSignalAddr(4, "", "", 0)
    %assign py5 = LibBlockOutputSignalAddr(5, "", "", 0)
    %assign py6 = LibBlockOutputSignalAddr(6, "", "", 0)
    %assign py7 = LibBlockOutputSignalAddr(7, "", "", 0)
    %assign py8 = LibBlockOutputSignalAddr(8, "", "", 0)
    %assign py9 = LibBlockOutputSignalAddr(9, "", "", 0)
    %assign py_width = LibBlockOutputSignalWidth(0)
    %assign pu_width = LibBlockInputSignalWidth(0)
    Calcolo_Jacobiano_Outputs_wrapper(%<pu0>, %<pu1>, %<pu2>,
    %<pu3>, %<pu4>, %<pu5>, %<py0>, %<py1>, %<py2>, %<py3>,
    %<py4>, %<py5>, %<py6>, %<py7>, %<py8>, %<py9> );

    %%
  %endfunction
```

A.3 Virtual world

Usually, the end-effector position of the haptic device is determined by reading the position of the actuators and processing such measurements through the direct kinematic model of the actual device; this position is associated to a virtual object inside the virtual world (avatar), thus achieving the interaction between the haptic device and the virtual world. This interaction is represented in the Matlab/Simulink model shown in Fig. A.6. This model updates the fields of a virtual world using Matlab/Simulink data and implements haptic and graphic rendering.

If a signal with zero value is given to the enable port, both haptic and graphic rendering are disabled. To permanently enable the block, it must receive the value of one. The Device Pos input port must receive a 3-dimensional end effector position of a haptic input device (see Fig. A.7). The Info output port provides some information about debugging, monitoring if there is any communication problem. The Force output port is used to output the generating force in response to interaction of the haptic input device with the object.

If the hapticsEnabled field is FALSE, haptic rendering of the virtual world will bypass the corresponding object, instead if is TRUE, haptic rendering

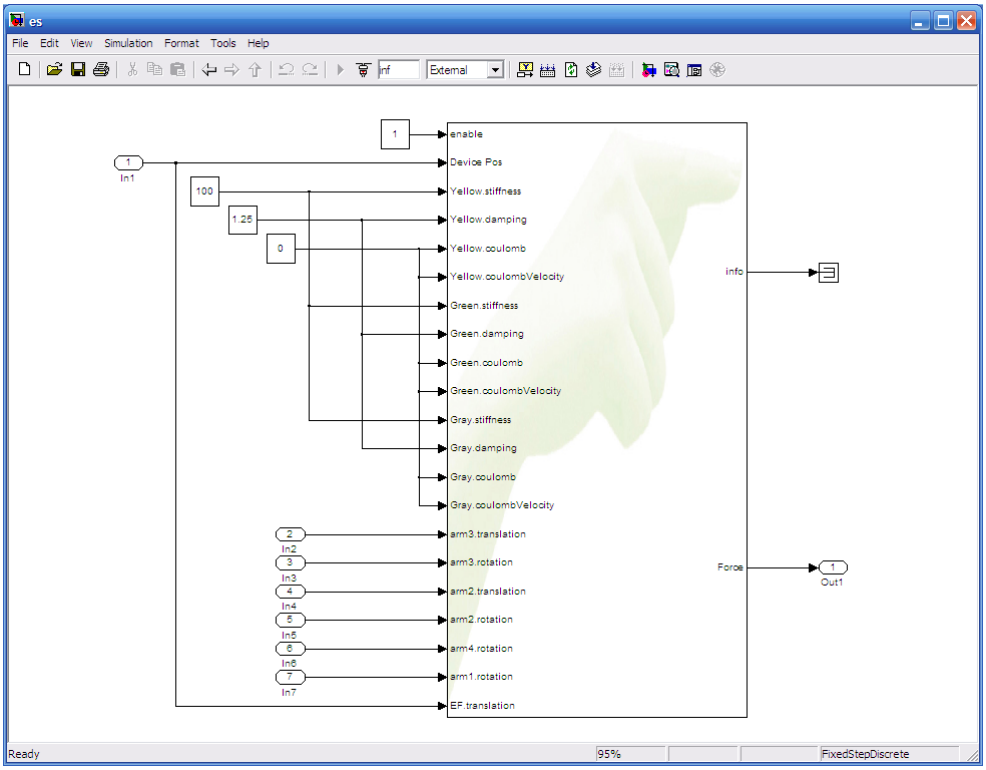


Figure A.6: Haptic and graphic rendering of a virtual world using Matlab/Simulink.

is activated (see Fig. A.8). The haptic fields stiffness, damping, coulomb, coulombVelocity, advancedFriction, stiction, stictionVelocity are used to describe the haptic properties of the object (see Fig. A.9).

As an example, the end effector position of the haptic device that interacts with the virtual environment is illustrated in Fig. A.10, in which the red sphere represents the avatar.

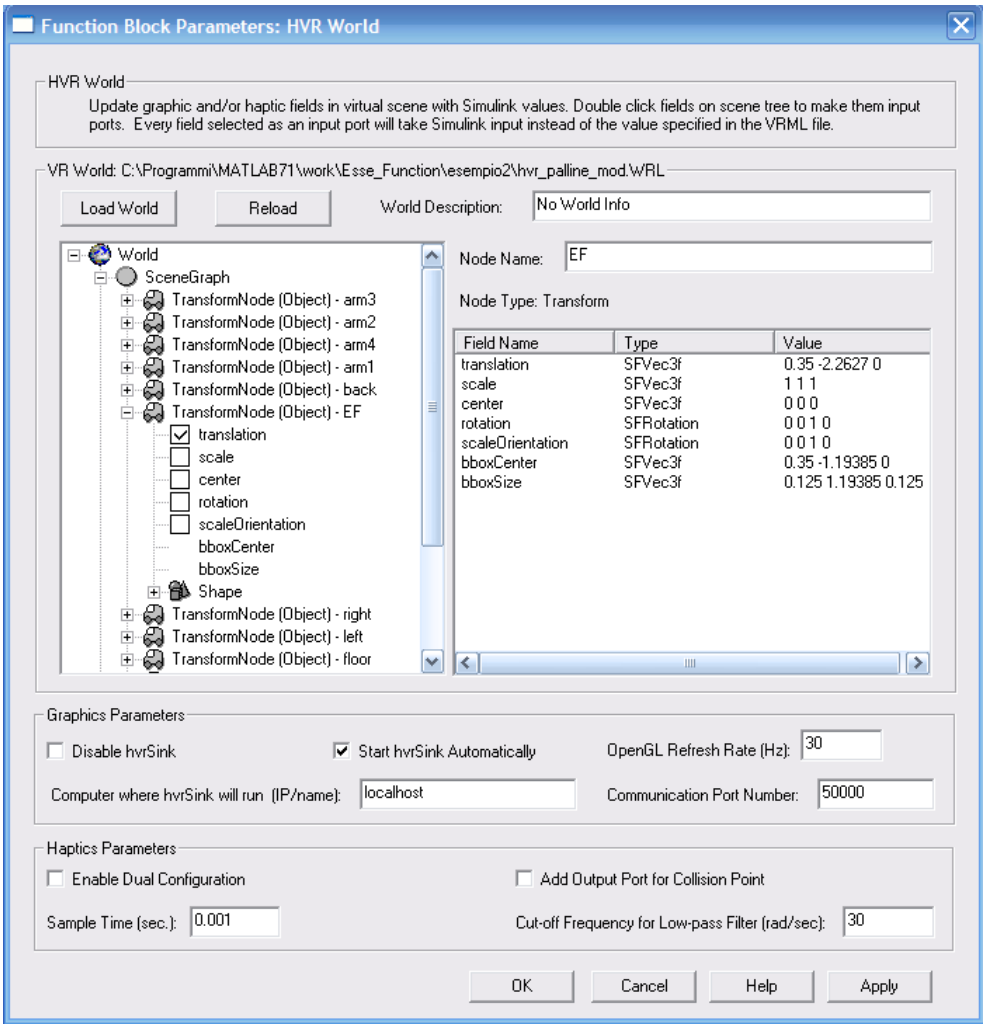


Figure A.7: Definition of the device's end effector position inside the virtual world.

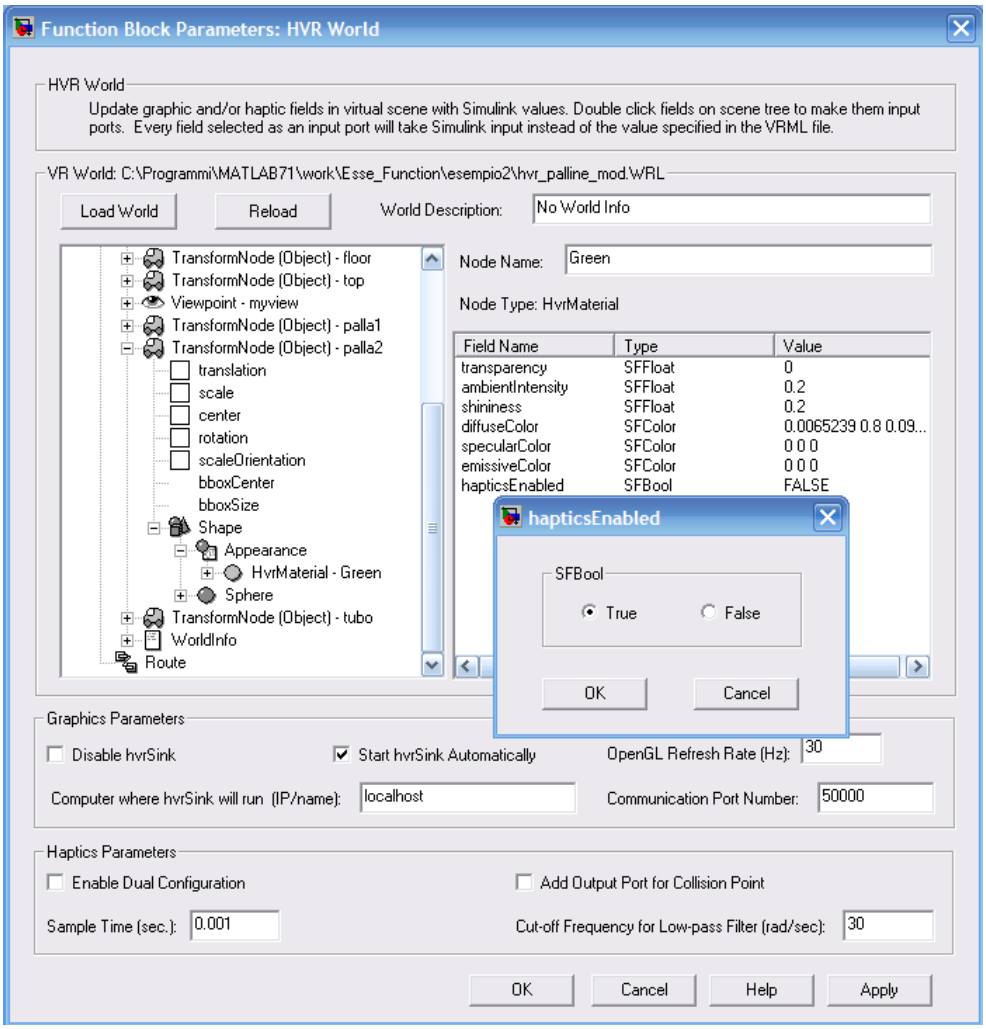


Figure A.8: Enabling the haptic properties of virtual objects.

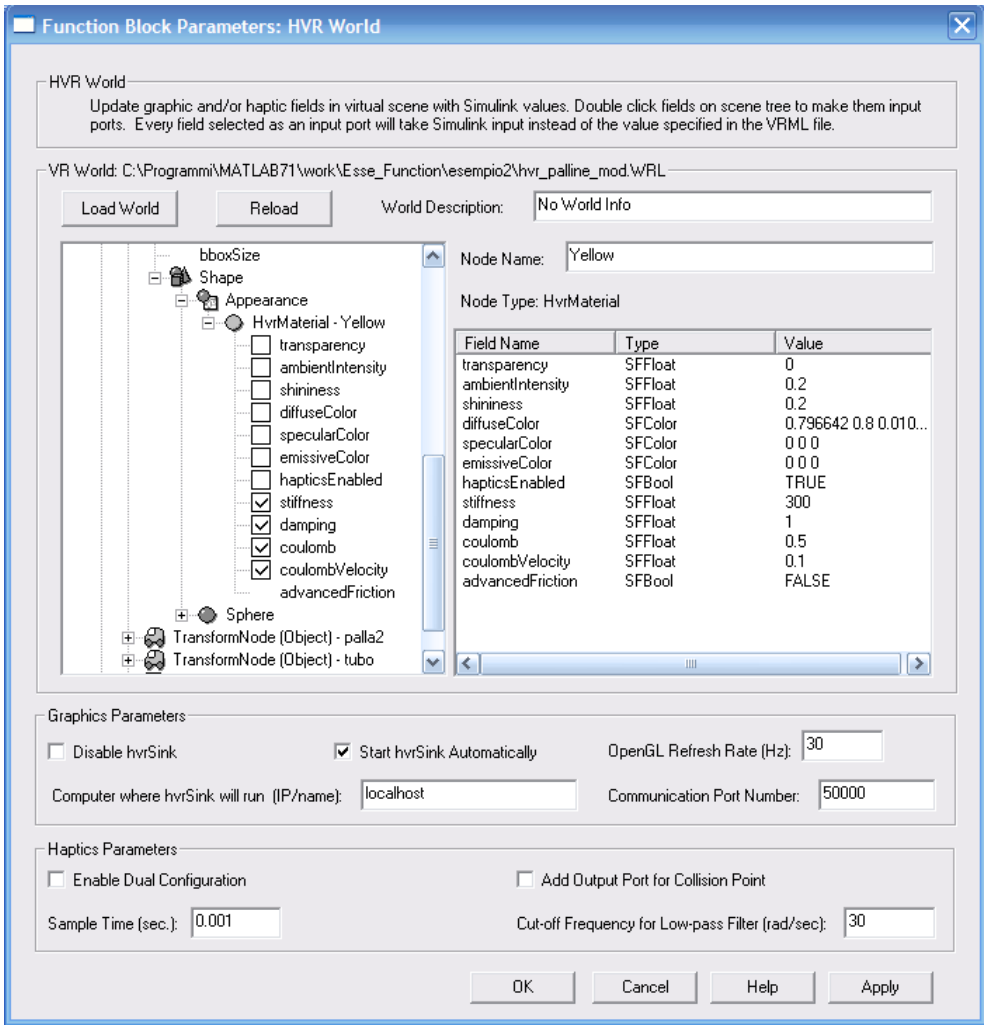


Figure A.9: Adding haptic properties of virtual objects.

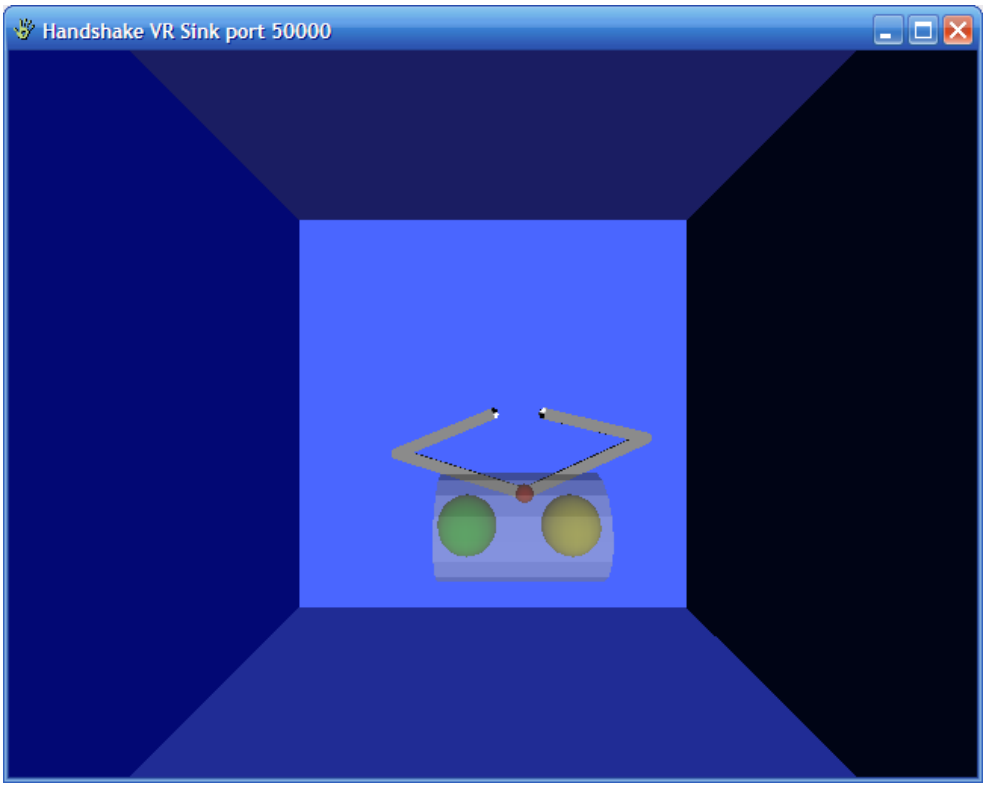


Figure A.10: The avatar interacting with the virtual world.

Appendix B

Robust acceleration control

This appendix introduces the concept of robust acceleration control based on disturbance observer (DOB). The concept of disturbance observer is developed based on motor dynamics. The robustness of such a method is discussed as well. Then, the sensorless torque estimation based on the reaction torque observer (RTOB), which is at the same time based on the concept of DOB, is presented too.

B.1 Acceleration control by disturbance observer

The dynamics of a motor can be described by the following equation:

$$J\frac{d\omega}{dt} + T_l = T_m \tag{B.1}$$

where T_m is the generated torque, T_l the load torque. J and ω the motor inertia and angular velocity, respectively. The dynamics are presented as a block diagram in the Laplace domain, as illustrated in Fig. B.1.

The total generated torque T_m is given by:

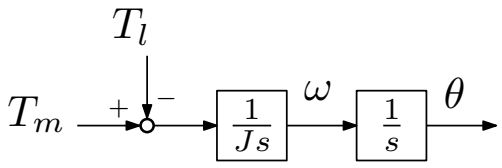


Figure B.1: Block diagram of a motor dynamics.

$$T_m = K_t I_a = K_t I_a^{ref} \tag{B.2}$$

where K_t is the torque coefficient and I_a the torque current. Fast switching devices make the power converter with feedback of torque current as a virtual current converter. In most cases, it is possible to use I_a as torque current reference I_a^{ref}.

The load torque can be considered by the following expression:

$$T_l = T_{int} + T_{ext} + (F_c + D\omega) \tag{B.3}$$

where T_{int} is the internal interaction torque that includes the Coriolis term, the centrifugal term and gravity term. T_{ext} is the reaction torque when the mechanical system actuates a force. F_c and $D\omega$ are the Coulomb and viscous friction, respectively. Substituting Eq. B.2 and Eq. B.3 in Eq. B.1, we have the following expression:

$$Js\omega = K_t I_a^{ref} - [T_{int} + T_{ext} + (F_c + D\omega)] \tag{B.4}$$

The parameters in Eq. B.4 are the inertia of the motor J and the torque coefficient K_t. J will change according to the mechanical configuration of motion system, as follows,

$$J = J_n + \Delta J \tag{B.5}$$

where J_n stands for nominal inertia and ΔJ represents the deviation from the nominal value. The torque coefficient will vary according to the position of the rotor in the electric motor due to irregular distribution of magnetic flux on the surface of the rotor:

$$K_t = K_{tn} + \Delta K_t \tag{B.6}$$

where K_{tn} denotes the nominal torque coefficient and ΔK_t denotes the deviation from the nominal value.

The components of the total disturbance torque T_{dis} are:

- The mechanical load: T_l.
- The variation of the self-inertia torque: $\Delta Js\omega$.
- The torque ripple of the motor: $-\Delta K_t I_a^{ref}$.

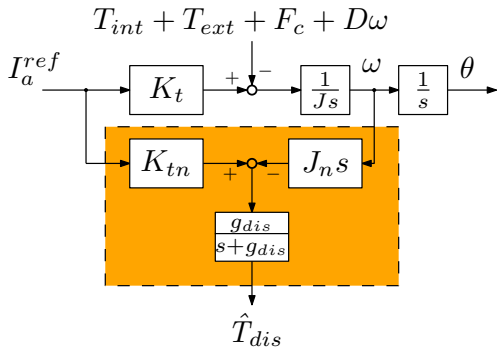


Figure B.2: Disturbance observer structure.

The disturbance torque T_{dis} can be represented by the following expression:

$$\begin{aligned} T_{dis} &= T_l + \Delta J s\omega - \Delta K_t I_a^{ref} \\ &= T_{int} + T_{ext} + F_c + D\omega + (J - J_n)s\omega + (K_{tn} - K_t)I_a^{ref} \end{aligned} \tag{B.7}$$

A disturbance observer is designed in order to cancel the disturbance. The estimated disturbance torque is obtained from the velocity ω and the current reference I_a^{ref}, as shown in Fig. B.2.

In order to reduce the noise due to the derivative term, the disturbance torque is estimated through the first order low-pass filter:

$$\hat{T}_{dis} = \frac{g_{dis}}{s + g_{dis}} T_{dis} \tag{B.8}$$

where g_{dis} is the cut-off frequency of the low-pass filter. If g_{dis} is large enough, the estimated disturbance torque is almost the same with respect to the actual one.

The disturbance torque is the sum of the load effect and the parameter variations. It is impossible to decompose it into each element. However, for the robust control purposes, it is enough to know only the sum of both. Therefore, the direct feedback of the identified disturbance torque will be effective for robust motion control. The feedback loop of the disturbance is just the same as the feedforward effect of the disturbance to cancel it. Hence, a robust motion controller is achieved by using the disturbance observer. Besides, the robust motion controller makes a motion system to be an acceleration control system, as shown in Fig. B.3. The effect of the disturbance torque is represented as a transfer function G_s:

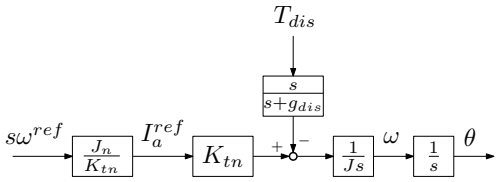


Figure B.3: A robust acceleration control system by using the disturbance observer.

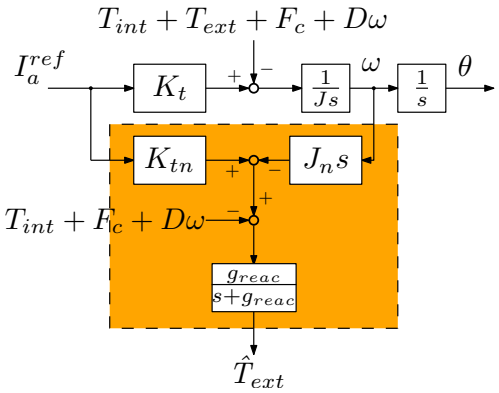


Figure B.4: Reaction torque estimation.

$$G_s = \frac{s}{s + g_{dis}} T_{dis} \tag{B.9}$$

G_s represents the sensitivity determining how much the disturbance torque influences the motion system. This is called the sensitivity function.

B.2 Reaction torque estimation

In section B.1, the disturbance, which its estimation is represented by Eq. B.8, is used for the implementation of a robust motion controller.

In the actual application, the estimated disturbance torque is effective for not only the disturbance compensation but also the identification in the mechanical parameters of the system.

The output of the disturbance observer is only the friction effect under the constant angular velocity motion. This issue makes possible the identification of the friction effect in a mechanical system. The external force effect is also

identified by using the estimated disturbance (Murakami et al., 1993). Here it is assumed that the friction effects are known beforehand by the above identification process. By implementing the angular accelerated motion, the system parameters K_{tn} and J_n are adjusted in the observer design so that they are close to the actual values, respectively. As a result, the disturbance observer estimates only the external force as follows:

$$\hat{T}_{ext} = \frac{g_{reac}}{s + g_{reac}} \left[(I_a^{ref} K_{tn} - J_n s\omega) - (T_{int} + F_c + D\omega) \right] \tag{B.10}$$

where $\hat{T}_{ext}$ is the estimated reaction torque and g_{reac} is the cutoff frequency of the reaction torque estimation. The identification process of the external force is summarized in Fig. B.4.

Bibliography

Adamovich, S. V., Merians, A. S., Boian, R., Tremaine, M., Burdea, G. S., Recce, M., and Poizner, H. (2005). A virtual reality based exercise system for hand rehabilitation post-stroke. *Presence, Special Issue on Virtual Rehabilitation*, 14:161–174.

Aisen, M. L., Krebs, H. I., Hogan, N., Mcdowell, F., and Volpe, B. T. (1997). The effect of robot-assisted therapy and rehabilitative training on motor recovery following stroke. *Arch Neurol*, 54(4):443–446.

Amirabdollahian, F., Loureiro, R., Gradwell, E., Collin, C., Harwin, W., and Johnson, G. (2007). Multivariate analysis of the fugl-meyer outcome measures assessing the effectiveness of gentle/s robot-mediated stroke therapy. *Journal of NeuroEngineering and Rehabilitation*, 4(1):4.

Anderson, R. and Spong, M. (1988). Bilateral control of teleoperators with time delay. In *Decision and Control, 1988., Proceedings of the 27th IEEE Conference on*, pages 167 –173 vol.1.

Anderson, R. and Spong, M. (1989a). Asymptotic stability for force reflecting teleoperators with time delays. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 1618 –1625 vol.3.

Anderson, R. and Spong, M. (1989b). Bilateral control of teleoperators with time delay. *Automatic Control, IEEE Transactions on*, 34(5):494 –501.

Baraff, D. (1994). Fast Contact Force Computation for Nonpenetrating Rigid Bodies. In *Computer graphics Proceedings, SIGGRAPH 1994*, pages 23–34, Orlando, USA.

Barker, R. N. and Brauer, S. G. (2005). Upper limb recovery after stroke: The stroke survivors' perspective. *Disability & Rehabilitation*, 27(20):1213–1223.

Benedetti, C., Franchini, M., and Fiorini, P. (2001). Stable tracking in variable time-delay teleoperation. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*.

Buttolo, P. and Hannaford, B. (1995). Advantages of actuation redundancy for the design of haptic displays. In *Proceedings, ASME Fourth Annual Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, volume DSC-57-2, pages 623–630, San Francisco.

Carignan, C. and Krebs, H. (2006). Telerehabilitation robotics: Bright lights, big future? *Journal of Rehabilitation Research & Development*, 43:695 – 710.

Colgate, J. and Brown, J. (1994). Factors affecting the z-width of a haptic display. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 3205 –3210 vol.4.

Colgate, J. E. and Hogan, N. (1988). Robust stability of dynamically interacting systems. *International Journal of Control*, 48(1):65–88.

Eusebi, L. and Melchiorri, C. (1998). Force reflecting telemanipulators with time-delay: stability analysis and control design. *Robotics and Automation, IEEE Transactions on*, 14(4):635 –640.

Fasoli, S. E., Krebs, H. I., Stein, J., Frontera, W. R., and Hogan, N. (2003). Effects of robotic therapy on motor impairment and recovery in chronic stroke. *Archives of Physical Medicine and Rehabilitation*, 84(4):477 – 482.

Fazekas, G., Horvath, M., and Toth, A. (2006). A novel robot training system designed to supplement upper limb physiotherapy of patients with spastic hemiparesis. *International journal of rehabilitation research*, 29(3):251–254.

Feng, X. and Winters, J. (2005). Unitherapy: a computer-assisted motivating neurorehabilitation platform for teleassessment and remote therapy. In *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on*.

Fugl-Meyer, A. R., Jääskö, L., Leyman, I., Olsson, S., and Steglind, S. (1975). The post-stroke hemiplegic patient. 1. a method for evaluation of physical performance. *Scandinavian journal of rehabilitation medicine.*, 7(1):13–31.

Goldberg, K., Mascha, M., Gentner, S., Rothenberg, N., Sutter, C., and Wiegley, J. (1995). Desktop teleoperation via the world wide web. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, volume 1, pages 654 –659 vol.1.

Handshake VR Inc. (2006). *Handshake proSENSE^{TM} Virtual Touch Toolbox v2.0 User's Guide*. Handshake VR Inc. Copyright ©2001-2006.

Hannaford, B. (1989a). A design framework for teleoperators with kinesthetic feedback. *Robotics and Automation, IEEE Transactions on*, 5(4):426–434.

Hannaford, B. (1989b). Stability and performance tradeoffs in bi-lateral telemanipulation. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 1764 –1767 vol.3.

Harris, J. E. and Eng, J. J. (January 2007). Paretic Upper-Limb Strength Best Explains Arm Activity in People With Stroke. *Physical Therapy*, 87(1):88–97.

Hashtrudi-Zaad, K. and Salcudean, S. (2002). Transparency in time-delayed systems and the effect of local force feedback for transparent teleoperation. *Robotics and Automation, IEEE Transactions on*, 18(1):108 –114.

Haykin, S. S. (1970). *Active Network Theory*. Reading, MA: Addison-Wesley.

Hesse, S., Werner, C., Pohl, M., Rueckriem, S., Mehrholz, J., and Lingnau, M. (2005). Computerized arm training improves the motor control of the severely affected arm after stroke: A single-blinded randomized trial in two centers. *Stroke*, 36(9):1960–1966.

Hogan, N. (1989). Controlling impedance at the man/machine interface. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 1626 –1631 vol.3.

Iida, W. and Ohnishi, K. (2004). Reproducibility and operationality in bilateral teleoperation. In *Advanced Motion Control, 2004. AMC '04. The 8th IEEE International Workshop on*, pages 217 – 222.

Jaeger, R. J. (2006). Guest Editorial: Rehabilitation robotics research at the National Institute on Disability and Rehabilitation Research. *Journal of Rehabilitation Research and Development*, 43(5):xvii–xx.

Jebsen, R. H., Taylor, N., Trieschmann, R. B., Trotter, M. J., and Howard, L. A. (1969). An objective and standardized test of hand function. *Archives of physical medicine and rehabilitation*, 50(6):311–319.

Kahn, L., Zygman, M., Rymer, W. Z., and Reinkensmeyer, D. (2006). Robot-assisted reaching exercise promotes arm movement recovery in chronic hemiparetic stroke: a randomized controlled pilot study. *Journal of NeuroEngineering and Rehabilitation*, 3(1):12.

Kassler, M. (1993). Introduction to the special issue on robotics for health care. *Robotica*, 11(06):493–494.

Kim, W., Hannaford, B., and Fejczy, A. (1992). Force-reflection and shared compliant control in operating telemanipulators with time delay. *Robotics and Automation, IEEE Transactions on*, 8(2):176 –185.

Krebs, H., Hogan, N., Aisen, M., and Volpe, B. (1998). Robot-aided neurorehabilitation. *Rehabilitation Engineering, IEEE Transactions on*, 6(1):75 –87.

Lawrence, D. (1993). Stability and transparency in bilateral teleoperation. *Robotics and Automation, IEEE Transactions on*, 9(5):624 –637.

Lederman, S. and Klatzky, R. (1987). Hand movements: a window into haptic object recognition. *Cognit. Psychol.*, 19(3):342 – 368.

Leung, G., Francis, B., and Apkarian, J. (1995). Bilateral controller for teleoperators with time delay via mu;-synthesis. *Robotics and Automation, IEEE Transactions on*, 11(1):105 –116.

Lum, P. S., Burgar, C. G., Shor, P. C., Majmundar, M., and der Loos, M. V. (2002). Robot-assisted movement training compared with conventional therapy techniques for the rehabilitation of upper-limb motor function after stroke. *Archives of Physical Medicine and Rehabilitation*, 83(7):952 – 959.

Masiero, S., Celia, A., Rosati, G., and Armani, M. (2007). Robotic-assisted rehabilitation of the upper limb after acute stroke. *Archives of Physical Medicine and Rehabilitation*, 88(2):142 – 149.

Mathiowetz, V., Volland, G., Kashman, N., and Weber, K. (1985a). Adult norms for the Box and Block Test of manual dexterity. *The American journal of occupational therapy: official publication of the American Occupational Therapy Association*, (6):386 – 391.

Mathiowetz, V., Weber, K., Kashman, N., and Volland, G. (1985b). Adult norms for the Nine Hole Peg Test of finger dexterity. *Occupational Therapy Journal of Research*, (5):24 – 37.

Merians, A. S., Poizner, H., Boian, R., Burdea, G., and Adamovich, S. (2006). Sensorimotor Training in a Virtual Reality Environment: Does It Improve Functional Recovery Poststroke? *Neurorehabilitation and Neural Repair*, 20(2):252–267.

Mirtich, B. and Canny, J. (1995). Impulse-based simulation of rigid bodies. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, I3D '95, pages 181–ff., New York, NY, USA. ACM.

Murakami, T., Yu, F., and Ohnishi, K. (1993). Torque sensorless control in multidegree-of-freedom manipulator. *Industrial Electronics, IEEE Transactions on*, 40(2):259 –265.

Nakayama, H., Jørgensen, H. S., Raaschou, H. O., and Olsen, T. S. (1994). Recovery of upper extremity function in stroke patients: the Copenhagen Stroke Study. *Arch Phys Med Rehabil*, 75(4):394–398.

Natori, K., Oboe, R., and Ohnishi, K. (2008). Stability analysis and practical design procedure of time delayed control systems with communication disturbance observer. *Industrial Informatics, IEEE Transactions on*, 4(3):185 –197.

Natori, K., Tsuji, T., Ohnishi, K., Hace, A., and Jezernik, K. (2010). Time-delay compensation by communication disturbance observer for bilateral teleoperation under time-varying delay. *Industrial Electronics, IEEE Transactions on*, 57(3):1050 –1062.

Nef, T. and Riener, R. (2005). Armin - design of a novel arm rehabilitation robot. In *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on*.

Niemeyer, G. and Slotine, J.-J. (1991a). Stable adaptive teleoperation. *Oceanic Engineering, IEEE Journal of*, 16(1):152 –162.

Niemeyer, G. and Slotine, J.-J. (1991b). Transient shaping in force-reflecting teleoperation. In *Advanced Robotics, 1991. 'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, pages 261 –266 vol.1.

Oboe, R. (2001). Web-interfaced, force-reflecting teleoperation systems. *Industrial Electronics, IEEE Transactions on*, 48(6):1257 –1265.

Oboe, R. (2003). Force-reflecting teleoperation over the internet: the jbit project. *Proceedings of the IEEE*, 91(3):449 – 462.

Oboe, R. and Fiorini, P. (1998). A Design and Control Environment for Internet-Based Telerobotics. 17(4):433–449.

Ottenbacher, K. J., Smith, P. M., Illig, S. B., Linn, R. T., Ostir, G. V., and Granger, C. V. (2004). Trends in Length of Stay, Living Setting, Functional Outcome, and Mortality Following Medical Rehabilitation. *JAMA: The Journal of the American Medical Association*, 292(14):1687–1695.

Oujamaa, L., Relave, I., Froger, J., Mottet, D., and Pelissier, J.-Y. (2009). Rehabilitation of arm function after stroke. literature review. *Annals of Physical and Rehabilitation Medicine*, 52(3):269 – 293.

Outpatient Service Trialists (2003). Therapy-based rehabilitation services for stroke patients at home. *Cochrane Database of Systematic Reviews*, (1).

Patton, J., Dawe, G., Scharver, C., Mussa-Ivaldi, F., and Kenyon, R. (2004). Robotics and virtual reality: the development of a life-sized 3-d system for the rehabilitation of motor function. In *Engineering in Medicine and Biology Society, 2004. IEMBS '04. 26th Annual International Conference of the IEEE*, volume 2, pages 4840 –4843.

Prange, G. B., Jannink, M. J. A., Groothuis-Oudshoorn, C. G. M., Hermens, H. J., and IJzerman, M. J. (2006). Systematic review of the effect of robot-aided therapy on recovery of the hemiparetic arm after stroke. *Journal of rehabilitation research and development*, 43(2):171–184.

Robinson, D., Pratt, J., Paluska, D., and Pratt, G. (1999). Series elastic actuator development for a biomimetic walking robot. In *Advanced Intelligent Mechatronics, 1999. Proceedings. 1999 IEEE/ASME International Conference on*, pages 561 –568.

Rohrer, O., Fasoli, S., Krebs, H. I., Hughes, R., Volpe, B., Frontera, W. R., Stein, J., and Hogan, N. (2002). Movement smoothness changes during stroke recovery. *J Neurosci*, 22:8297–8304.

Rosati, G., Cenci, S., Boschetti, G., Zanotto, D., and Masiero, S. (2009). Design of a single-dof active hand orthosis for neurorehabilitation. In *Rehabilitation Robotics, 2009. ICORR 2009. IEEE International Conference on*, pages 161 –166.

Salcudean, S. E., Yan, J., Hu, Z., and Loewen, P. D. (1995). Performance tradeoffs in optimization-based teleoperation controller design with applications to microsurgery experiments. pages 631–640.

Salisbury, J. K. and Craig, J. J. (1982). Articulated hands: Force control and kinematics issues. *The International Journal of Robotics Research*, 1(1):4–17.

Sano, A., Fujimoto, H., and Tanaka, M. (1998). Gain-scheduled compensation for time delay of bilateral teleoperation systems. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 3, pages 1916 –1923 vol.3.

Sawaki, L. (2005). Use-dependent plasticity of the human motor cortex in health and disease. *Engineering in Medicine and Biology Magazine, IEEE*, 24(1):36 –39.

Shadmehr, R. and Mussa-ivaldi, O. A. (1994). Adaptive representation of dynamics during learning of a motor task. *Journal of Neuroscience*, 14:3208–3224.

Shimoga, K. (1993). A survey of perceptual feedback issues in dexterous telemanipulation. i. finger force feedback. In *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*, pages 263 –270.

Spong, M. W. and Vidyasagar, M. (1989). *Robot Dynamics and Control*. Wiley.

Stramigioli, S., van der Schaft, A., Maschke, B., and Melchiorri, C. (2002). Geometric scattering in robotic telemanipulation. *Robotics and Automation, IEEE Transactions on*, 18(4):588 – 596.

Suzuki, A. and Ohnishi, K. (2010). Performance conditioning of time delayed bilateral teleoperation system by scaling down compensation value of communication disturbance observer. In *Advanced Motion Control, 2010 11th IEEE International Workshop on*, pages 524 –529.

The MathWorks Inc. (2005). *Writing S-Functions in C*. The MathWorks Inc. Copyright ©1994-2005.

Venema, S. and Hannaford, B. (2001). A probabilistic representation of human workspace for use in the design of human interface mechanisms. *Mechatronics, IEEE/ASME Transactions on*, 6(3):286–294.

Wolbrecht, E., Leavitt, J., Reinkensmeyer, D., and Bobrow, J. (2006). Control of a pneumatic orthosis for upper extremity stroke rehabilitation. In *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE*.

Wolfe, C. D. A. (2000). The impact of stroke. *British Medical Bulletin*, 56(2):275–286.

Yokokohji, Y. and Yoshikawa, T. (1994). Bilateral control of master-slave manipulators for ideal kinesthetic coupling-formulation and experiment. *Robotics and Automation, IEEE Transactions on*, 10(5):605 –620.

Yoshikawa, T. (1985). Manipulability of Robotic Mechanisms. *The International Journal of Robotics Research*, 4(2):3–9.

Zilles, C. and Salisbury, J. (1995). A constraint-based god-object method for haptic display. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 3, pages 146 –151.