The WAM arm: modelling, control and its application in a HMI based on gaze tracking

Thesis submitted for the degree of

Doctor of Philosophy

at the University of Leicester

BY

Zaira Pineda Rico

Department of Engineering

University of Leicester

2014

A tree too big to embrace
is born from a slender shoot
A nine-story tower
rises from a pile of earth
A thousand mile journey
begins with a single step

Tao Te Ching
Lao Tzu

Abstract

In this thesis we describe the design and implementation of a Human Machine Interface (HMI) based on gaze tracking proposed to control robot prostheses. Robot manipulators hold a strong similarity with arm prosthetics, we used a 7 degrees of freedom (DOF) whole arm manipulator to test our HMI in the execution of reaching and grasping tasks. We showed that the interface worked under different control strategies using several velocity profiles. The system was tested by ten subjects with encouraging results. We analysed the performance of the 7-DOF robot manipulator in order to determine the suitability of its application in the development of this project. The original setup of the manipulator worked under joint Proportional and Derivative (PD) control but considering the results of the initial analysis of the system we proposed two alternative control strategies aimed to improve the performance of the manipulator: a feedforward friction compensation technique and joint Proportional Integral and Derivative control (PID). We created a dynamic model of the 7-DOF manipulator in Simmechanics in order to have a better understanding of the system. The friction phenomena of the manipulator was identified, represented through a fitted model and included in the system's model with the aim of incrementing its accuracy with respect to the real system. The characteristics of the model made it suitable to test and to design control strategies for motion and friction compensation in MATLAB/Simulink. The model of the system was validated using data from the real robot arm and it was used later to tune the PID controllers of the joints of the 7-DOF manipulator using Iterative Feedback Tuning (IFT). Both experimental data and model simulations

were used for the tuning procedure considering two different approaches. The data obtained from the friction identification process was used to implement a module for feedforward friction compensation over the pre-configured joint PD control of the manipulator. The responses of the system when using joint PID control and joint PD control with gravity and friction compensation were compared in the execution of motion tasks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Several years have passed since researchers started to speculate about the advances in the development of the Brain-Machine Interfaces (BMI) and their implications in the future of neuroprosthetics (Nicolelis, 2001). The advances in technology have led to the development of new generations of electrodes and recording equipment, powerful computers and math-friendly programming languages that allow the implementation of new methods of signal processing used to control mechanical manipulators for the implementation of tasks in virtual or real-time contexts (Parag and Turner, 2008). In addition, research in neuroscience has helped to construct a new platform for the development of Brain-Computer Interfaces (BCI's) that work as prosthetic devices connected to the brain. This type of devices are aimed to help paralised people to regain movement or to compensate for organs or body functions that have been damaged as a consequence of illness, accidents, etcetera.

Although recent technological advances have been a key point in the development of different types of BCI, the concept is not new. The brain coordinates all the functions on the human body and relies on electrical activity to create a web of communication among neurons. These electrical activity can be recorded using electrodes. This recording is known as electroencephalogram (EEG). Single neuron

recordings in human have been documented since the 1960's when a recording in a patient for diagnostic and therapeutic brain surgery was published. From 1960 onwards single cell recordings were used in the design and implementation of BCI's in humans considering a variety of coding methods (Schwartz et al., 2001; Taylor et al., 2002, 2003). The subject is implanted with microelectrodes in the motor cortex and the registered single cell activity is processed using adaptive learning algorithms. This technique has proven to be effective in different BCI implementations in monkeys (Schwartz, 2004; Schwartz et al., 2006; Velliste et al., 2008) and in people with tetraplegia for controlling a cursor on a computer screen (Hochberg et al., 2006; Simeral et al., 2009; Kim et al., 2011) or for moving a robotic arm (Hochberg et al., 2012).

However, the invasive characteristics of the method may bring complications for the patient. Also, after the patient is implanted with the electrodes it has to be trained in order to be able to control its neuronal firing rates. The training period varies among patients and can take from weeks to months. There are subjects that are not able to successfully accomplish the training process.

In this thesis we propose the use of gaze tracking to control the position of a robot prosthetic device. Eye movements have been experimentally studied since the nineteen century (Yarbus, 1967; Wade and Tatler, 2005), which led to the development of devices and methods to scan eye movements. Among these methods we can mention the scleral search coil tracker, electrooculography (EOG) and different implementations of optic methods, which are based on the reflections of a beam of light onto the corneal surface. Electrooculography and eye tracking based on corneal reflections are popular methods used in Human Computer Interfaces (HCI). Eye tracking has been widely used in the past to implement HCI's designed to aid paralised people in the realisation of several tasks by controlling a cursor on a computer screen. These type of developments have been under study since 1989 through the implementa-

tion of various human computer interaction devices (Hutchinson et al., 1989; Jacob, 1991). HCI based on eye movements take advantage not only of gaze tracking but intentional blinking as control signals. For example, gaze tracking might be used as a directional reference in two dimensions while blinking patterns may work as binary signals. Using the measurable characteristics of eye movements it is possible to generate a set of control signals for the implementation of a more sophisticated HCI, like a prosthetic arm based on eye movements.

Commercial gaze tracking devices are non invasive, which reduce the health risks that invasive methods bring to patients undergoing surgery due to the implantation of electrodes, such as swelling, superficial skin infections, infections along lead wires, intracranial infections, bleeding, etc. There is also the possibility that the implanted electrode might move with respect to the surrounding tissue, that the tissue might affect the sensors or that the electrode might be rejected by the body (Wolpaw and Wolpaw, 2012).

The user does not require extensive training to use the gaze tracker, however it has to be aware of some considerations to make while wearing the device. It only takes from 10 to 20 minutes for the eye tracker to be fully operational in the user.

As mentioned, gaze tracking has been used before to control cursors in two dimensions using computer screens. We intend to use gaze tracking in a similar implementation but removing completely the use of the computer screen. This will give a more natural feeling to the use of the gaze tracking control although it will increase the complexity for the user. In this type of implementation the subject will have to rely in mental plots of the surrounding space since its eyes will be moving frequently. Also, the subject will have to be fully conscious about the relationship between their visual space and the gaze tracker visual space at all times. The design must consider a resting time when the subject can hold the manipulator's control on pause since making fixations for long periods is tiring for the muscles in

charge of the eye movements, bringing pain to the user.

Prosthetic arms in paralysed people help them to regain certain degree of independence. The principal objective of robotic superior limbs is to reproduce the movement of the arm by successfully accomplish grasping tasks with smooth and natural motions. Due to the use of gaze tracking has shown high potential for position control of cursors we expect to obtain a reliable system that allows to control the robot arm in several reaching and grasping tasks similar to those executed in past BCI implementations. We want to test the functionality of the system in different subjects in order to analyse the performance and the pertinence of the HMI. According to the results obtained in the experiments we will discuss about the suitability of designing a 3D gaze tracking control able to compete with similar BCI implementations.

If we consider that the physical similarities and performance requirements between the robot manipulator and the artificial prosthetic for the superior limb, most of the control schemes used in manipulators may be adapted to be used by prosthetic arms (Orin, 1980). This is why a robot manipulator can be considered as a good representation of the human arm and might be used as a platform to develop control strategies that can be used in prototypes of future artificial prosthetics. However, the design of suitable control strategies involves the consideration of mathematical representations of the system, such as kinematics and dynamics models. Mathematical modelling is fundamental in the designing process depending of the objectives of the implementation, the constraints of the task to be executed, and the desired robot's performance. The difficulty in obtaining these models varies according to the complexity of the kinematics of the mechanical structure and the number of degrees of freedom (DOF) (Khalil and Dombre, 2002). This thesis will also present the implementation of two alternative control strategies in the WAM arm proposed

with the aim of identifying a suitable technique that can be used to control de manipulator for future neuroprosthetic applications.

1.2 Contributions

The main contributions of this work are listed briefly:

- A dynamic model of the 7-DOF Barret WAM arm implemented in SimMechanics that includes the identification of friction using experimental data.
- The implementation of a friction compensation module in the joint controllers of the manipulator.
- The application of an iterative tuning technique that was originally designed for linear systems in a real non linear system using both real and simulated data.
- The design and implementation of a non invasive HMI based on gaze tracking that was successfully tested in several subjects in reaching and grasping tasks similar to those accomplished by paralysed patients through invasive BCI applications.

1.3 Publications

- Dynamic model of a 7-DOF Whole Arm Manipulator and validation from experimental data. Zaira Pineda Rico, Andrea Lecchini-Visintini, Rodrigo Quian Quiroga. International Conference on Informatics in Control, Automation and Robotics (ICINCO). 2012. 217-222.
- Iterative feedback tuning for the joint controllers of a 7-DOF whole arm manipulator. Zaira Pineda Rico, Andrea Lecchini-Visintini, Rodrigo Quian Quiroga. IEEE 51st Annual Conference on Decision and Control (CDC). 2012. 544-549.

1.4 Literature Review

1.4.1 Robot modelling and Robot Control

Robot manipulators have been used in the industry since the 1950s. Through the years several control schemes have been developed in order to fulfil the user's expectations in the realisation of particular tasks. In general, the mechanical structure of a robot manipulator consist of a sequence of rigid bodies called *links* that are interconnected by either rotating or translating mechanisms that work as articulations (or *joints*). The last link usually holds an *end-effector*, whose architecture makes it suitable to grasp objects (Siciliano et al., 2009). The specific configuration of the *links* and *joints* trace the requirements to be considered in the design of the control scheme used to move the manipulator.

A robot manipulator is a mechanical structure that can be commanded to accomplish a desired task and its motion involves a complex system composed by sensors and motors. The control computer that works as an interface between the user and the manipulator holds the controller of the system. The complexity of the controller varies according to the manipulator's architecture, its degrees of freedom and the task to be executed. The effectiveness of the controller will depend on the kinematics and dynamics considerations taken by the designer. In most cases these considerations are based on the behaviour that mathematical models of the manipulator exhibit to different types of stimulus (Khalil and Dombre, 2002).

The use of mathematical modelling in robotics has supported the design of several types of robots like animal inspired robots, wheeled mobile robots, human-like biped robots and manipulators with as few as two degrees of freedom, to more complex configurations with more than six degrees of freedom. In the particular case of robot manipulators that hold similarities with an artificial prosthetic arm,

many models and parameter identification methods have been developed according to the arm physical configuration, which depends entirely on the fabricator's target market. This is the case of the *Mitsubishi PA-10 robot arm* (Kennedy and Desai, 2004; Bompos et al., 2007; Lightcap and Banks, 2007) and the *PUMA 560 robot arm* (Armstrong et al., 1986; Neuman and Murray, 1987; Izadbakhsh, 2009), and systems to prototype such as the *Hyundai Robot Hardware in the Loop Simulation* (Yeon et al., 2005), whose particular structure allowed them to reach popularity not only in the industry but in research laboratories around the world, where they are employed for experimental verification and design. Moreover, some popular configurations as the Stanford and the Scara manipulators, have been properly analysed with the purpose of illustrating different methodologies for robot modelling and kinematic analysis (Spong et al., 2006).

Recently, *Whole Arm Manipulators (WAM)* caught the researchers' attention because of their human-like kinematics and near-zero joint friction (Barrett Technology Inc, 2008b). These characteristics facilitate the design of controllers and provide efficiency in obtaining more accurate data for the identification of the robot parameters. In the case of the *Whole Arm Manipulators*, as for other manipulators with a serial link configuration, according to the desired characteristics of the controller behaviour the mathematical and dynamics equations can be calculated using recursive methods such as Euler-Lagrange, Newton-Euler or a combination of both (Sousa et al., 2009). Most of the mathematical models that have supported the design of controllers for a Whole Arm Manipulator (WAM) are based in the computation of multi-links serial robot's mathematical equations. These equations are obtained using the Newton-Euler recursive method to find the Coriolis, centrifugal and inertial forces observed when the end-effector is in motion, and use the Jacobian approach for mapping in order to minimise singularity conditions that increase the computation load of the control algorithm (Lau and Wai, 2002).

This methodology involves both operational and joint forces. In order to avoid significant computation time, some authors have found in MATLAB/SimMechanics a comfortable tool to design mechanical systems used for experimental verification (Yeon et al., 2005).

The computation of the dynamic model of a robot manipulator plays an important role in the simulation of motion, the analysis of the manipulator's structure and the design of optimal control algorithms. The performance of robot manipulators with multiple links is always compromised due to the gravity force affecting the dynamics of the system, among other non linearities such as friction force and inertial forces. Frequently, the minimisation of the error in the execution of tasks will demand the implementation of a suitable controller able to deal with the effects of gravity, or the addition of a gravity compensation module to the control scheme.

The most common control strategies used in robot manipulators are the joint Proportional Derivative (PD) control and the Proportional Integral and Derivative control. PD controllers are simple to implement and their parameters are relatively easy to find, but they are not able to compensate for the effects of the gravity force. Hence, modules for gravity compensation have to be included in the control loop in order to eliminate the offset error in the response of the system (Kelly et al., 2005). On the other hand, the integral component of the PID controller is able to neutralise the influence of gravity during the execution of tasks. PID control is the most popular strategy implemented in the industry for robot control (Spong et al., 2006).

Gravity compensation is an important issue in robot control, however is the existence of other disturbances what represent a real challenge when designing control strategies for robot manipulators. Friction force in the joints of the manipulator,

for example, will never guarantee a zero error in the output response of the controllers. Commonly, the effects of friction must be cancelled separately through the implementation of a compensation strategy. The most recommendable action to succeed on the later is to identify the actual friction phenomena manifested in the system and use this data to create a mathematical model that can serve as a reference.

Olsen and Petersen (Olsen and Petersen, 2001) presented a method to find the model parameters of a robot in order to support the design of model-based robot controllers. They also proposed the computation of the model parameters of friction using experimental methods for two links of the Mitsubishi PA-10 robot. Kostic and colleagues (Kostic et al., 2004) showed the importance of accurate friction modelling in model based nonlinear control. Their work summarise a procedure to improve the performance of an RRR-Robotic arm and validated their results with a writing task. Other contribution to the model of the Mitsubishi PA-10 robot is presented in (Bompos et al., 2007), where the authors describe the full identification of the model parameters, the estimation of the stiffness of the joints and a new nonlinear friction model for the joints of the manipulator. The verification of the model is made through an end effector trajectory tracking task.

There are several models used in friction research. Armstrong and colleagues published a survey where all the theory behind friction phenomena is explained and proposed an integrated friction model based on seven parameters that include the pre-sliding displacement, the Coulomb friction, the viscous friction, the Stribeck curve friction and the friction level at breakaway (Armstrong-Helouvry et al., 1994). Canudas de Wit and colleagues proposed a dynamic model that considered the contact surfaces as contact between bristles (Canudas de Wit et al., 1995) and illustrated the procedure of identification of friction phenomena to validate the

model using experimental data (Canudas de Wit and Lischinsky, 1997). Hensen and colleagues presented two grey-box models and validated them using experimental data obtained from a rotating arm (Hensen et al., 2000).

In some cases the influence of friction does not affect significantly the performance of the system, so a compensation technique might not be required. In other cases the control strategy might be effective enough to compensate the error caused by the non linearities brought by the friction phenomena in the joints of the manipulator. For example, a PID control is capable of dealing with friction when all its parameters are conveniently selected (Lewis et al., 2004). However, tuning the parameters of the PID is not an easy task and frequently requires the implementation of techniques based on iterative methods that use experimental data (Hamamoto et al., 2003; Hildebrand et al., 2005). Iterative techniques allow to select the control parameters according to the actual performance of the system affected by disturbances.

In 1994 Hjlmarsson and colleagues proposed and iterative model-free method for tuning control parameters, which proved to be efficient in simulations of a linear control scheme. The method uses a minimising criterion based on the computation of the Hessian using the Newton method and the gradient of the tracking error (Hjalmarsson et al., 1994). Using a similar minimising criterion, Sjöberg and Agarwal presented a repetitive method for non linear systems that computes the gradient of the error following a series of experiments where the input signal changes slightly with respect to a principal reference signal (Sjoberg and Agarwal, 1996). A similar method was described by De Bruyne and colleagues in (De Bruyne et al., 1997). The technique proposed by Hjalmarsson proved to be also effective in non linear systems, although its implementation worked only under limited conditions (Hjalmarsson, 1998, 2002). Considering a different approach, Hamamoto and colleagues presented a

method to tune the controllers of a two-mass spring system. The method suggest the separate tuning of the feedback and feedforward controllers and compute the Hessian using a quasi Newton method (Hamamoto et al., 2003). Sjöberg and colleagues extended their work in (Sjorberg et al., 2003) and developed a complete algorithm for the controller optimisation of nonlinear systems following a series of experiments where the reference signal is changing according to a variable parameter. They used these data to compute the gradient of the error. The method was tested only through simulated systems.

The iterative feedback tuning methods previously described were designed to tune the parameters of the controllers without depending on the mathematical model of the system. However, these methods might be suitable to implement on an accurate mathematical model of the system that includes non linearities such as the inertia and the friction forces. The described works about modelling of robot manipulators indicate that it is suitable to implement an accurate dynamic model of the system using its inertial information and using a friction identification technique to measure the friction phenomena that affects its performance.

1.4.2 Human computer interaction based on eye movements

There are several methods to scan eye movements; however, Electrooculography (EOG) and gaze tracking are the most popular techniques for the development of Human Computer Interaction devices (HCI). The eye movements commonly used as control signals include saccades, fixations and intentional blinking.

Electrooculography allows to measure blinks and the movement of the eyes by positioning skin electrodes near to the eyes in order to register the resting potential of the retina (Wade and Tatler, 2005). This method is widely used in ophthalmologic diagnosis and has been popular in the development of HCI due to its non invasive nature. Some of the most relevant developments in EOG based HCI are

summarised below.

In 1990 LaCourse and Hludik Jr. developed a communication tool that consisted in a visual interface in a computer screen displaying several targets. Each of the targets could be selected by making a fixation on its presentation on the screen. The authors imply that the system is prompt to control output drivers and mechanical or electrical devices (LaCourse and Hludik, 1990). In 1998 Tecce and colleagues developed a cursor that represented a moving fixation point on a computer display (Tecce et al., 1998), the cursor helped to spell complete sentences using an alphabet matrix displayed on the screen. In 2002 Barea and colleagues implemented a system that controlled the direction of an electric wheelchair by using a cursor to select different commands on a on-board computer (Barea et al., 2002). In 2007 Borghetti and colleagues developed a low cost and easy-to-use HCI based on EOG signals. Their system gave a platform to help the user to communicate and to control external devices (Borghetti et al., 2007). Other developments of devices that measure eye movements have been designed considering their future implementations in HCI's and HMI's. For example, in 2010 Deng and colleagues developed a new eye movement tracker using EOG signals that achieved more than 90% of accuracy (Deng et al., 2010). The same year Bulling and colleagues proposed a method to analyse eye movements using repetitive patterns for eye based activity recognition (EAR) (Bulling et al., 2010). In 2012 proposed a new eye control method based on wavelet transform and neural networks with and error of less than 2 degrees (Barea et al., 2012). According to the authors, the systems are prompt to be used in the design of computer interfaces controlled by eye movements.

Although EOG based HCI has kept researchers interested through the past years, optic based eye tracking devices have been highly developed for commercial purposes. The optic methods for eye tracking use the reflection properties of

the cornea to measure the rotation of the eye with respect to a reference. A beam of light, usually infra red (IR), is projected onto the eye from a fixed source. When the eye moves the position of the light relative to the cornea will change giving a point of reference to register eye rotations. The principle of corneal reflection can be implemented in head mounted devices that use two cameras to record simultaneously the user's front scene and the eye with the reflection of the IR light on the cornea. These devices known as gaze trackers synchronize the images of the scene camera with the eye image to find the gaze point.

There have been several advances in eye/gaze tracking based human computer interfaces. In 1989 Hutchinson and colleagues implemented a prosthetic device that consisted in a stand-alone workstation with an interface that displayed an option menu, where the options of the menu were activated by fixation. The team used a near-infrared/camera assembly to measure the gaze point (Hutchinson et al., 1989). In 2005 Noureddin and colleagues proposed the use of two cameras for gaze tracking. The system considered not only the corneal reflection but it recognised the pupil in real time using image processing techniques. This set up was able to compute the gaze position in the presence of head motion and was designed for HCI implementations (Noureddin et al., 2005). In 2008 Sesin and colleagues proposed a system for gaze tracking to control a mouse cursor. They used neural networks (ANN) that required a short training session to reduce the sporadic motion of the eye (saccades) (Sesin et al., 2008b,a). Similarly, Varona and colleagues developed an interface that recognised gestures to move a mouse cursor (Varona et al., 2007). In 2009 De Santis and Iacoviello developed an eye tracking procedure based on pupil detection in real time, using several levels of segmentations between successive frames of a camera pointing to the face (De Satis and Iacoviello, 2009). The system was specifically designed to be used by disabled people in computer interfaces. Also in 2009, Tall and colleagues presented an

interface using on screen buttons to control the direction and speed of a remote vehicle. The set up included a mouse-pointing low-cost webcam eye tracker and two commercial eye tracking systems (Tall et al., 2009). In 2011 Rantanen and colleagues designed a head mounted gaze tracker with facial movement detection. The device was wireless and it was suggested to act as an input for HCI using the the gaze as a pointer and facial gestures for selection (Rantanen et al., 2011). Arai and Mardiyanto used a head mounted gaze tracking system to control an interface for phoning, reading e-book/e-comic/e-learning and that allowed internet browsing and TV information extraction (Arai and Mardiyanto, 2011). In 2012 Krolak and Strumillo implemented a system for eye blink detection using image processing methods matching based eye tracking and eye blink detection. They constructed an interface with a screen mouse, virtual keyboard, internet browsing and a panel with short-cuts to user-selected computer programs (Krolak and Strumillo, 2012).

Other novel implementations are presented in (Reale et al., 2011) and (Zhao et al., 2012). Reale and colleagues designed an HCI that includes not only gaze tracking as control signal, but the use of gestures, head pose, hand pointing and mouth motions. The authors also developed a new calibration approach to find the three dimensional (3D) eyeball location, eyeball radius, and fovea position. On the other hand, Zhao and colleagues developed a game system based on head pose to control a virtual robot walking in a virtual maze environment.

1.5 Organisation of the thesis

This thesis is organised in seven chapters.

Chapter 1 is an introduction to the contents of the thesis. It describes the motivation behind this work and gives a summary of past research that encouraged the techniques used in the development of this project.

Chapter 2 contains the theoretical background found in the literature that helped

to sustain the implemented methods and materials.

Chapter 3 describes the dynamic model of the 7 degrees of freedom (DOF) robot manipulator and its validation using experimental data.

Chapter 4 presents a feed-forward friction compensation technique that was implemented on the Proportional and Derivative (PD) joint control with gravity compensation that was configured in the 7-DOF manipulator.

Chapter 5 explains the tuning process of the 7 Proportional Integral and Derivative (PID) controllers parameters from the joints of the robot manipulator.

Chapter 6 follows the design and implementation of a Human Machine Interface based on gaze tracking and its application to control a robot manipulator.

Chapter 7 contains the general conclusions derived from this work and offers the insights of proposals for future work.

Chapter 2

Theoretical Background

Technological advances have led to the development of several types of prosthetic devices designed to aid paralysed people. These devices are designed to compensate for failures that the user organs or limbs may have, offering a comfortable and natural feeling. In order to achieve the later, the control of these devices is usually based on bio signals. Among the prostheses based on bio signals we can mention neural prostheses, myoeletric prostheses and robotic prostheses.

Myolectric prostheses use muscles electric activity to control mechanical devices in order to execute a task, for example the rotation of a wrist, the flexion of an elbow or the grasp of a hand. Neural prostheses are devices based on brain activity that are capable to compensate for neural damage that affects motor, sensory or cognitive processes. There are several types of neural prostheses: visual prostheses, auditory implants, cognitive prostheses and motor implants. Visual prostheses stimulate neurons in the visual system in order create an image in the brain. Auditory implants such as cochlear implants, auditory brain stem implants and auditory midbrain implants, stimulate auditory nerves aiding the sound processing in the brain. Cognitive prostheses are used to replicate brain functions that are originally executed by tissue that has been damaged. Motor implants are aimed for conscious control of movement of mechanical limbs, or robotic prosthetics. Robotic prosthetics are built as complete systems that include sensors, controllers and

actuators to control sophisticated mechanical structures such as robotic arms.

When a computer is used to process neural activity in order to control a device, the system is known as Brain Computer Interface (BCI). There are several developments that have helped to encourage the research in the field of BCI. Braingate, for example, is a very sophisticated system that consist in an array of microelectrodes that can be implanted in the brain to record signals, and uses a decoder to find useful commands to control external devices such as a computer, a wheelchair, or a prosthetic robotic limb (BrainGate). In most cases, the development of the interfaces intended to control prosthetic robotic limbs is made using commercial robotic arms which anatomy is similar to the human arm. The DLR Light Weight Robot developed by the Institute of Robotics and Mechatronics in the German Aerospace Center, and the DEKA which belongs to a project sponsored by the Defense Advanced Research Projects Agency and the U.S. Army Research Office are popular devices that fit this description. There are also computer interfaces that use eye movements as control signals, although these are mainly used for communication, web browsing, control of applications and control of peripheral devices.

Robotic arms, also known as robot manipulators, follow diverse trajectories in order to complete a desired task. The accurate motion of the mechanical structure relies on forces applied to the joints of the manipulator using controllers. There are several linear and non linear control techniques used to control the joints of robot manipulators, considering the system as either a single-input/single-output (SISO) or a Multi-input/Multi-output (MIMO) system.

When the robot manipulator is considered as a SISO system, each link is controlled independently and the coupling effects are regarded as disturbances (Spong et al., 2006). If the disturbances are constant a compensator may be applied. Among the

most used compensators in robotics are the Proportional Derivative (PD) and Proportional Integral and Derivative (PID). When the joint trajectory is time varying, the compensator may be supported by adding a feedforward path to the output of the compensator. If the mathematical model of the system is known, a dynamical system based on such a model (observer) can be used to compensate for the disturbances using an estimation of the full state of the system.

When the manipulator is considered as a MIMO system the control problem is non linear. The control techniques are then thought to cancel the non linearities based on the dynamic model of the manipulator, in most cases. Common strategies used to compensate non linearities in MIMO systems are PD control with gravity compensation, inverse dynamics control, robust control and adaptive control.

Inverse dynamics control is based on the exact cancellation of non linearities using a non linear state feedback. If a perfect compensation can not be reached we can consider robust control and adaptive control. The implementation of robust control depends on an estimation of the uncertainties and the tolerance of the mechanical structure in order to establish control inputs. Adaptive control uses an on line estimation of the dynamic model making on line adaptations that compensate for the external disturbances.

PID control is a control strategy widely used in the industry due to the fact that the integral component of the controller can achieve zero steady state error, while keeping the gains small. This type of controller has robust performance under several operating conditions and its design is simple due to it is based on a small number of parameters.

However, non linear phenomena may appear when the integral term accumulates a significant error that breaks the feedback loop. This may happen due to the actuator limitations and can be avoided by limiting the controller output or by using external reset feedback. Tuning a PID controller is not an easy task, the designer must rely on

specific methods in order to find the proper values of the parameters. In most cases the initial tuning is made though computer simulation considering the mathematical model of the system. Once the controller is implemented, on-line tuning is used to adjust the parameters through the execution of several experiments.

The basic rules for PID tuning are the Ziegler-Nichols rules. These rules apply under different conditions. If the plant dynamics are not known accurately, the method uses the transient response of the plant to a unit step input and the values of the parameters are chosen according to the output. In a second version of the method, the proportional gain is changed until it reaches a critical value K_u, at this time the output exhibits sustained oscillations with period P, the parameters are chosen considering the values of K_u and P (Ogata, 2010). Other common method to tune the controller using the transient response of the plant to a unit step input is the relay feedback auto tuning. In this method a relay is connected in a feedback loop with the plant, since the amplitude of the oscillation and the relay output are proportional, the relay amplitude may be auto adjusted to reduce the oscillations.

It is possible to tune the parameters of PID controllers even if the dynamics of the system are complete unknown using Iterative Feedback Tuning (IFT). This method is based on the behaviour of the controller undergoing several experiments, and allows a closer analysis of the performance of the controller when executing a desired task. The method uses the collected data with a gradient based minimisation criterion in order to find the control parameters iteratively. Although the method was originally designed for linear systems it has been successfully applied to non linear systems. This type of tuning is limited to the execution of specific tasks and requires a series of experiments that in some manipulators are not suitable to perform.

2.1 Modelling in Robotics

From the mechanics viewpoint, a manipulator can be represented as a sequence of rigid bodies (*links*) interconnected by means of articulations (*joints*). Each of these links symbolise an important element of the manipulator's architecture: a positioning arm, a wrist used to give dexterity and the end-effector that performs the desired task. At the same time, the mobility of the manipulator's structure depends on the type of articulation that interconnects the links, which can be either prismatic or revolute. A prismatic joint produces a relative translational motion between two links, whereas a revolute joint produces a relative rotational motion between the two links (Siciliano et al., 2009). The characteristics of links and joints altogether set the workspace of the manipulator.

The mechanical structure described previously is connected to internal and external sensors that provide the computer, which holds the system's controller, with motion information (Spong et al., 2006). All this information about displacement, velocity and acceleration of the links is used to determine the controller behaviour, which objective is to manipulate the mechanical structure with proficiency so that the desired task can be executed successfully.

There are two mathematical analyses needed for the controller in order to position the manipulator in a required location: the kinematic and the dynamic study. The kinematic analysis tells the controller *where* to move whilst the dynamics reveals *how* to get there (Asada and Slotine, 1986). Consecuently, two mathematical models have to be developed to understand and effectively control the behaviour of the manipulator by measuring related data from the sensors: the end-effector position, the forces of the actuators, the velocities and the accelerations of the joints. The mentioned models will be subsequently explained.

2.1.1 Kinematic Modelling

There are two different kinematic problems that can be solved in order to identify the end-effector position and orientation according to the provided information about the joints. If the joint variables (angles and translations) are given, the position of the end-effector can be calculated by solving the forward kinematics problem; else if only the position of the end-effector is known, the values of the joint variables can be determined through the inverse kinematic problem.

Direct Kinematics

Each link of the structure can be numbered from 0 to n, letting the base link to be numbered as 0. At the same time it is convenient to attach a coordinate frame to each link. Specifically, frame $o_i x_i y_i z_i$ is the frame attached to link i, to represent the position and orientation of the end-effector using consecutive homogeneous transformations from the last frame, back to the base frame $o_0 x_0 y_0 z_0$, referred to as the *inertial frame* (Asada and Slotine, 1986; Spong et al., 2006).

To compute the position and orientation of the end-effector, the characteristics of each joint and the length of the links must be given, since each movement of the end-effector is completely related to the joints displacement. Assuming that each joint has a single degree of freedom, a robot manipulator with n joints will have $n + 1$ links. Each joint has an associated variable (namely q_i) that can be an angle (revolute joint) or displacement (prismatic joint)

$$q_i = \begin{cases} \theta_i \text{ if joint is revolute} \\ d_i \text{ if joint is prismatic} \end{cases} \tag{2.1}$$

A simple procedure to assign an attached coordinate frame to each link of the manipulator is the Denavit-Hartenberg (DH)[1] convention. The DH convention is widely used to describe the position and orientation of frame i relative to frame $(i - 1)$

[1]The Denavit-Hartenberg convention is explained in detail in Appendix A.

using the 4×4 matrix

$$A_i^{i-1} = \begin{bmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & a_i cos\theta_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sin\alpha_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

where the first three (3×1) column vectors contain the direction cosines of the coordinate axes of frame i, whilst the last (3×1) column vector represents the position of the origin of frame i (Asada and Slotine, 1986).

The position of the end-effector with respect to the inertial frame is described by the position vector of the origin and the unit vectors of a frame attached to the body. It can be expressed by the homogeneous transformation matrix

$$T_n^0 = \begin{bmatrix} n_n^0(q) & s_n^0(q) & a_n^0(q) & p_n^0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

where q is the $(n \times 1)$ vector of joint variables, n_n, s_n, a_n are the unit vectors of the frame attached to the end-effector, and p_n is the position vector of the origin of the n frame with respect to the origin of the base frame (Siciliano et al., 2009). The transformation matrix is obtained considering the n DH matrices that describe the position and orientation of each link and which, being calculated consecutively, give the end-effector position relative to the base frame as

$$T_n^0 = A_1^0(q_1)A_2^1(q_2)...A_n^{n-1}(q_n) \tag{2.4}$$

Inverse Kinematics

Solving the inverse kinematic problem is more complicated than finding a solution for the direct kinematic problem. This is because the equations involved in the

inverse kinematic problem are strongly related to the robot configuration. There are several methods to solve the inverse kinematic problem: the geometric method, the homogeneous transformation matrix based method and the kinematic decoupling method. One of the difficulties to face when solving the inverse kinematic problem is that the solution is not unique. In order to facilitate the calculation of the variables that satisfy the inverse kinematic problem is recommendable to examine the robot's configuration. The geometric method is recommended for manipulators with few degrees of freedom. The homogeneous transformation matrix based method may be used in more complex configurations. However, implementing this method involves a careful analysis of the set of equations that conform the direct kinematic model of the system. Finally, the kinematic decoupling method is usually used to calculate the position of the wrist by taking advantage of the results obtained when one of the other two methods has been applied to the positioning system of the arm (Barrientos et al., 1997).

If the direct kinematic model is known, using Equation 2.4 and taking the initial position as a reference, it can be shown that

$$\begin{aligned} (A_1^0(q_1))^{-1}T_n^0 &= A_2^1(q_2)A_3^2(q_3)...A_n^{n-1}(q_n) \\ (A_2^1(q_2))^{-1}(A_1^0(q_1))^{-1}T_n^0 &= A_3^2(q_3)...A_n^{n-1}(q_n) \\ (A_{n-1}^{n-2}(q_{n-1}))^{-1}...(A_2^1(q_2))^{-1}(A_1^0(q_1))^{-1}T_n^0 &= A_n^{n-1}(q_n) \end{aligned} \tag{2.5}$$

Once all the relations have been identified, an algorithm can be developed to compute the values of each of the joint variables.

2.1.2 Dynamic Modelling

The computation of the dynamic model of a robot manipulator plays an important role in the simulation of motion, the analysis of the manipulator's structure and the design of control algorithms. Simulating the manipulator's motion allows the

testing of control strategies and motion planning techniques without even having a physical system.

Two of the most used methods for the derivation of the equations of motion of a manipulator in the joint space are the Euler-Lagrange equations and the recursive Euler Formulation. The Euler-Lagrange equations are derived from Newton's second law of motion, considering the Lagrangian of the system that represents the associated force with the set of joint variables. The Lagrangian for a system with n DOF is

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_k} - \frac{\partial \mathcal{L}}{\partial q_k} = \tau_k; \quad k = 1, ..., n \tag{2.6}$$

where τ_k is the force associated with the joint variable q_k. Given that the links of the robot manipulator are considered as interconnected rigid bodies, the motions of these bodies are related kinematically and the kinematic analysis of the system can be compared to that of the rigid body with a general three dimensional motion using relative motion analysis [2] (Meriam and Kraige, 2007). At the same time, assuming that the mass of the body is concentrated in its center of mass, the potential energy due to gravity can be easily calculated (Spong et al., 2006).

The Newton-Euler formulation is based on a balance of all the forces acting on every link of the manipulator (Siciliano et al., 2009). The total torque in each link can be computed using a recursive analysis of velocities, accelerations and propagating forces, taking advantage of pre-established coordinate systems of reference (attached frames) that can be chosen using the Denavit-Hartenberg convention for simplicity. Therefore, a recursive algorithm can be used to compute the forces associated to each joint variable with respect to a defined inertial frame [3].

[2] Relative motion analysis refers to the measurement of displacement, velocity and acceleration of an object with respect to a moving reference.

[3] The Newton-Euler recursive algorithm is reviewed in Appendix B.

In general, the dynamic model equation for a robot manipulator is written in the form

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F = \tau \tag{2.7}$$

where $B(q)$ is the inertial matrix, $C(q,\dot{q})\dot{q}$ is the Coriolis-Centrifugal matrix, $G(q)$ is the gravity vector and F is the vector of friction, which is usually obtained using a fitted friction model.

Direct Dynamics and Inverse Dynamics

There are two types of problems that can be solved through the computation of the dynamic model: the direct dynamics and the inverse dynamics problem. The solution of the direct dynamics problem determines the value of the position, velocity and acceleration joint variables for the given applied torques, whilst in the inverse dynamics problem the torque is calculated using the given joint position, velocity and acceleration. Either the Direct Dynamics problem and the Inverse Dynamics problem are desirable to be solved using numerical methods.

2.1.3 Friction Modelling

It is recommendable to include a friction model in a mechanical system model when designing a proper controller. Since the non linearity of friction may affect the performance of the controller, the use of a suitable friction model permits an adequate compensation. There are many models of friction employed when modelling position controlled mechanism. These models include several components like the sliding friction (or Coulomb friction), the breakdown friction (or stiction) and the viscous friction. An usual form to represent a friction model is expressed as

$$F = F_C + (F_S - F_C)e^{-|v/v_\sigma|^{\delta_\sigma}} + F_v v \tag{2.8}$$

where v_σ represent the Stribeck velocity [4], F_C is the Coulomb friction level, F_S is the level of the stiction force and $F_v v$ is the viscous friction (Olsson et al., 1998; Johnson and Lorenz, 1992).

In most cases, if the manipulator is expected to displace at medium or medium-high velocities, the friction can be modelled considering the effects of the viscous friction and the Coulomb friction only. This simplified representation of friction is known as classic friction model

$$F = F_c sign(\dot{q}(t)) + \sigma_2 \dot{q}(t) \tag{2.9}$$

where F_c is the parameter for Coulomb friction, σ_2 is the viscous damping coefficient and $\dot{q}(t)$ is the velocity of the link. The value of the parameters for the classic friction model can be obtained from experimental data through the execution of joint rotations at different constant velocities as explained hereafter.

2.2 Control Theory in Manipulators

Proportional Derivative (PD) Control with Gravity compensation and Proportional Integral and Derivative (PID) Control are widely used strategies in robot control. PD controllers use the derivative of the error to correct the control signal of the system increasing its stability while the PID controllers take advantage of the integral action to remove the steady state error (Ogata, 2010). The influence of gravity on a mechanical system can be computed using its geometric configuration. In most cases, when the system is in motion this type of computation would be difficult to execute on line and therefore the desired joint positions given by the reference trajectory have to be used (Kelly et al., 2005). The computed value might

[4]Stribeck noticed that the change from static friction to Coulomb friction can be expressed as a continuous function of the velocity. The range of velocity in which the Stribeck effect is effective is called *Stribeck velocity* (Olsson et al., 1998).

be added to the control loop as a gravity compensation module in order to improve the performance of the controller.

2.2.1 Proportional Derivative Joint Control with Gravity Compensation

The joint Proportional Derivative control with gravity compensation is expressed in Equation 2.10, where the control joint torque $\tau_c(t)$ is given as the sum of the difference between the reference and measured position $\tilde{q}(t)$, known as position error, multiplied by a constant proportional gain K_P, the derivative of the error $\dot{\tilde{q}}(t)$ multiplied by a derivative gain K_D and the compensation for gravity $g(q)$, which is a function of the joint position.

$$\tau_c(t) = K_P\tilde{q}(t) + K_D\dot{\tilde{q}}(t) + g(q(t)) \tag{2.10}$$

The diagram in Figure 2.1 shows the configuration of a joint PD controller with gravity compensation, as implemented in the WAM arm used in the development of this thesis. This control strategy successfully cancels the effects of gravity but it does not compensate the friction phenomena manifested in the joints of the manipulator. Then, the response error when executing a desired joint rotation is always nonzero.

2.2.2 Proportional Integral and Derivative Joint Control

As previously mentioned, every joint of the 7 DOF manipulator has been configured by the manufacturer with a PD controller and gravity compensation. Nevertheless, the system may be configured to perform under joint PID control by properly adjusting the gains of the joint controllers. PID control may be considered as an extended PD control which, apart from being dependent of the position error and

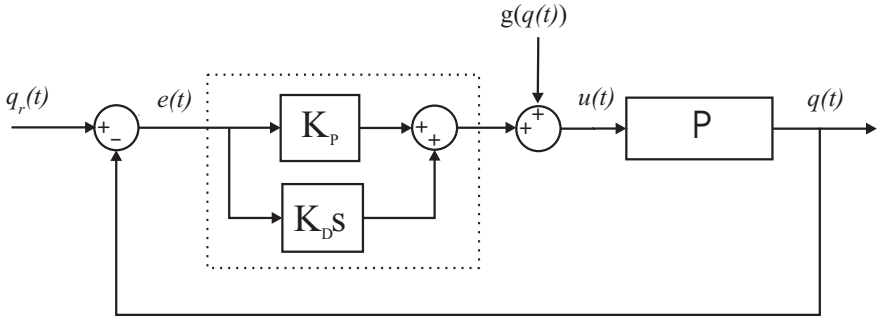


Figure 2.1: Configuration of the joint PD control with gravity compensation. $q_r(t)$ is the reference trajectory, P represents the manipulator, g is the gravity compensation as a function of the joint position $q(t)$, K_D and K_P are the derivative and proportional gains, respectively.

the derivative of the error, it considers the accumulation of the past errors through an integral term $\int_0^t \tilde{q}(\lambda)d\lambda$. The actuating torque is expressed as

$$\tau_c = K_P\tilde{q}(t) + K_D\dot{\tilde{q}}(t) + K_I\int_0^t \tilde{q}(\lambda)d\lambda \tag{2.11}$$

where $\tilde{q}(t)$ is the position error, $\dot{\tilde{q}}(t)$ is the derivative of the error and K_P, K_D and K_I are the proportional, derivative and integral gains respectively. The values for K_P, K_D and K_I are usually adjusted using different techniques according to the complexity of the process plant (Johnson and Moradi, 2005). Figure 2.2 shows the configuration of a joint PID controller.

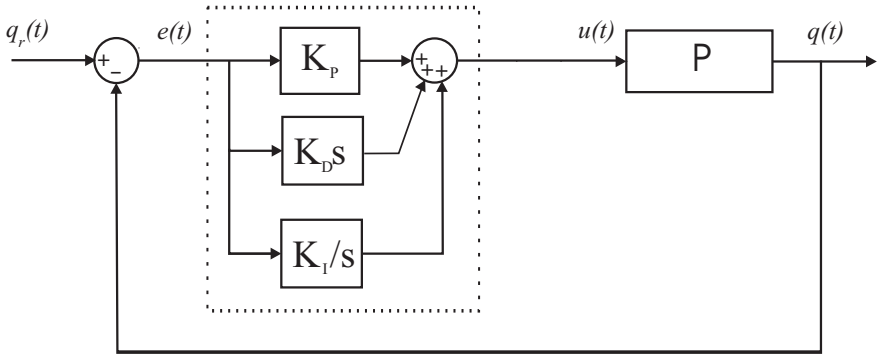


Figure 2.2: Configuration of the joint PID control. $r(t)$ is the reference trajectory, P represents the manipulator, K_D, K_I and K_P are the derivative, integral and proportional gains, respectively.

2.2.3 Iterative Feedback Tuning

Iterative feedback tuning is a technique used in iterative control design to find suitable controller parameters through experimental data. The method consists in the realisation of several experiments where the reference signal is changing. The obtained data is then used to construct a minimising criterion based on the error $\tilde{y}(\rho)$ of the output response. The method relies on the assumption that some disturbance v affects the performance of the system

$$\tilde{y}(\rho) = y(\rho) - y_r \tag{2.12}$$

where y_r is the desired system response to a reference signal r, $y(\rho)$ is the actual response as a function of ρ, which is the vector that contains the controller parameters. A control objective function $J(\rho)$ can be subsequently defined as a quadratic function of the expected value of the error $\tilde{y}(\rho)$ with respect to the disturbance v of the system

$$J(\rho) = \frac{1}{2N} \sum_{t=1}^{N} E[\tilde{y}(\rho)^2] \tag{2.13}$$

Whenever the condition of optimality $J'(\rho) = 0$ is met, the value of the parameter ρ, according to $\rho^* = \arg\min_\rho J(\rho)$, can be calculated iteratively by means of

$$\rho_{i+1} = \rho_i - \gamma_i R_i^{-1} \Big[\frac{\partial J(\rho_i)}{\partial \rho} \Big] \tag{2.14}$$

where R_i^{-1} is a positive definite matrix, γ_i is the iteration step and $\Big[\frac{\partial J(\rho_i)}{\partial \rho}\Big]$ is an estimate of the first derivative of the objective function.

The iterative feedback tuning technique is fully documented by Hjalmarsson in (Hjalmarsson, 2002). The value of γ_i in Equation 2.14 represents the size of the iteration step and it is variable through the tuning process. Generally this value is equal to 1 for the first iterations and becomes smaller as the instability point

is nearly reached. The rest of the parameters that shape Equation 2.14 are not straightforwardly found and its computation relies on experimental data.

Estimation of the gradient

The gradient of a function gives the direction of its rate of change. Considering Equation 2.13 the gradient of the objective function can be expressed as

$$\frac{\partial J(\rho)}{\partial \rho} = \frac{1}{N} \sum_{t=1}^{N} E\Big[\tilde{y}(\rho) \frac{\partial \tilde{y}(\rho)}{\partial \rho}\Big] \tag{2.15}$$

The value of $\tilde{y}(\rho)$ can be obtained with the execution of a closed loop experiment as the one shown in Figure 2.3.

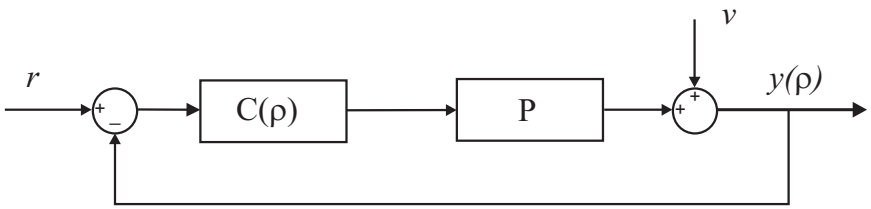


Figure 2.3: Closed loop experiment designed to find the value of $\tilde{y}(\rho)$. $C(\rho)$ represents the controller of the system as function of the current parameters contained in ρ, P represents the plant, r is the reference signal and $y(\rho)$ is the response of the system affected by a disturbance v.

The approximation of $\frac{\partial \tilde{y}(\rho)}{\partial \rho}$ requires the execution of a second experiment. The signal acquired in the previous experiment (let it be known as $y^1(\rho)$) subtracted from the reference signal r, endows the closed loop system of shown in Figure 2.3 and the output response is then filtered as shown in Figure 2.4. The following configuration was proposed by Hjalmarsson and colleagues in (Hjalmarsson et al., 1994).

Approximation of the Hessian

The matrix R_i in Equation 2.14 indicates the update direction for the estimation of the controller parameters in each iteration i and it is usually stated as the Hessian

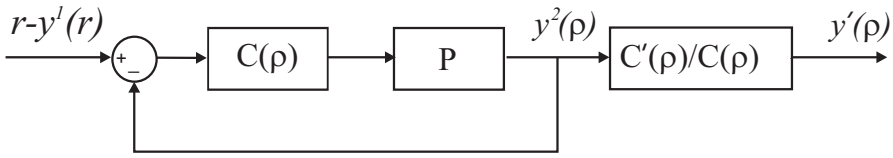


Figure 2.4: Closed loop experiment designed to find $\tilde{y}'(\rho)$, let $\tilde{y}^1(\rho)$ be the output of the experiment described in Figure 2.3. $C(\rho)$ is the controller of the system, P is the plant and $C'(\rho)$ is the derivative of the controller with respect to its parameters.

matrix of the objective function. The Hessian is a matrix formed with the second partial derivatives of a function of many variables, expressing its local curvature. A common choice for the approximation of the Hessian is to use the Gauss-Newton method

$$R_i = \frac{1}{N}\sum_{t=1}^{N}\Big(\Big[\frac{\partial\tilde{y}(\rho_i)}{\partial\rho}\Big]\Big[\frac{\partial\tilde{y}(\rho_i)}{\partial\rho}\Big]^T\Big) \tag{2.16}$$

where N is the number of samples, i represents the current iteration and ρ is the vector that contains the control parameters. The value of $\frac{\partial\tilde{y}(\rho_i)}{\partial\rho}$ for every iteration is obtained through the execution of both closed loop experiments described in Figure 2.3 and Figure 2.4.

A different technique to approximate the Hessian based on Broyden-Fletcher-Goldfarb-Shanno (BFGS) method for solving non linear optimisation problems has been proposed in (Hamamoto et al., 2003). In this technique the Hessian is approximated for each iteration i as a function of a previous estimation using

$$B_{k+1} = B_k + \frac{z_k z_k^T}{z_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} \tag{2.17}$$

Regarding to the equation, the Hessian $R_i = B_{k+1}$, $s_k = \rho_{k+1} - \rho_k$ and $z_k = J'(\rho_{k+1}) - J'(\rho_k)$. The matrix B_0 must be initialised for the first iteration using a positive definite matrix, an identity matrix is a common choice so that $B_0 = I$. For every iteration B_{k+1} is positive definite if B_k is positive definite and $z_k^T s_k > 0$. If the condition $z_k^T s_k > 0$ is not satisfied then $B_{k+1} = B_k$.

2.3 Gaze Tracking Theory

The eye processes visible light in order to generate visual information in the retina that is sent to the visual cortex of the brain through the optic nerve. The mayor features of the eye involved in the generation of visual information are shown in Figure 2.5 (Drake et al., 2010). The outer layer of the eye is a strong tissue known as sclera which is transparent in the front so that allows the refraction of the light in the cornea. The refracted light passes through the pupil to the retina where the image is created. The amount of light reaching the retina depends on the size of the pupil which is controlled by the iris (Tovee, 1996).

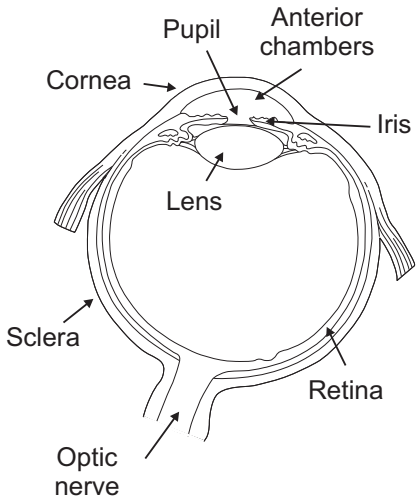


Figure 2.5: Simplified diagram of the structure of the eye (Drake et al., 2010).

The acquisition of visual signals involve a complex type of eye scanning, which execution depends on three dimensional rotations of the eye, performed by six specific muscles attached to the ocular globe. The location of the muscles is shown in Figure 2.6, the rectus muscles are in charge of the lateral and vertical motion while the oblique muscles execute longitudinal movements.

Every muscle has a main function when executing a specific type of eye rotation although they might have a minor participation in other actions:

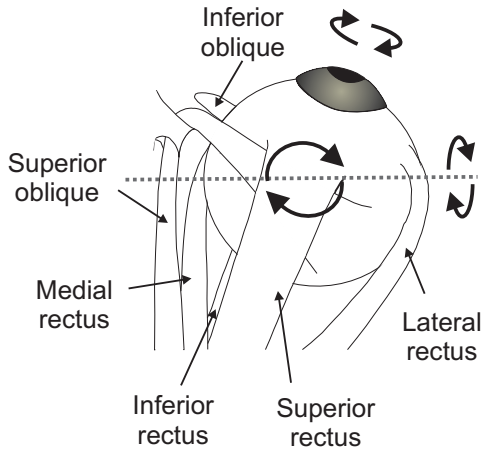


Figure 2.6: Top view of the eye. Six muscles are responsible for the three dimensional movement of the eyeball used in visual scanning (Snell and Lemp, 1998).

- Depression: Inferior rectus, Superior oblique.
- Elevation: Superior rectus, Inferior oblique.
- Extorsion: Inferior oblique, Inferior rectus.
- Intorsion: Superior oblique, Superior rectus.
- Abduction: Lateral rectus, Superior oblique, Inferior oblique.
- Adduction: Medial rectus, Superior rectus, Inferior rectus.

The process of visual scanning involves multiple voluntary and involuntary eye movements known as saccades. A voluntary saccade occurs when a subject moves its eyes to fixate the gaze in a specific object or point in space. However during the fixation process the eyes also execute rapid involuntary movements called involuntary saccades, and the axes of the eye present an irregular slow movement (drift) in addition to a high frequency oscillatory movement (tremor) that generates a noisy recording (Yarbus, 1967).

Gaze trackers are devices frequently used to measure the position of the gaze in a certain scene while dealing with the saccadic noisy signal. These devices are designed to take advantage of the physical characteristics of the eye in order to measure movement and rely on signal processing techniques to filter the signal so an

accurate approximation of the gaze may be computed. Some of the techniques used to filter the signal are finite-impulse response (FIR) filters, least square optimal filters altogether with FIR filtering, Kalman filtering and artificial neural networks (Spakov, 2012).

Most recent gaze tracker systems are head mounted and, as explained in Section 1.4.2, they involve different techniques in eye tracking to measure the gaze. The system used in the development of this work was a gaze tracker based on corneal reflection and pupil detection designed by Applied Science Laboratories (ASL). The Mobile Eye from ASL is a head mounted device that can be used in real time gaze tracking implementations. The acquisition system consist in a pair of glasses with a mounted set up of two cameras and an infra red (IR) light module. A diagram of the acquisition system is presented in Figure 2.7. The frontal camera records the front scene of the user while the lower camera records the image of the eye through an attached mirror. Beams of IR light are projected onto the cornea and are used as a reference point of the position of the eye with respect to the head, while the pupil is identified using the image of the eye. The angle of rotation of the eye can be found by finding the relationship between the corneal reflection and the pupil position. This information is then transposed on the scene view from the frontal camera and the gaze point is computed frame by frame (Applied Science Laboratories, 2008).

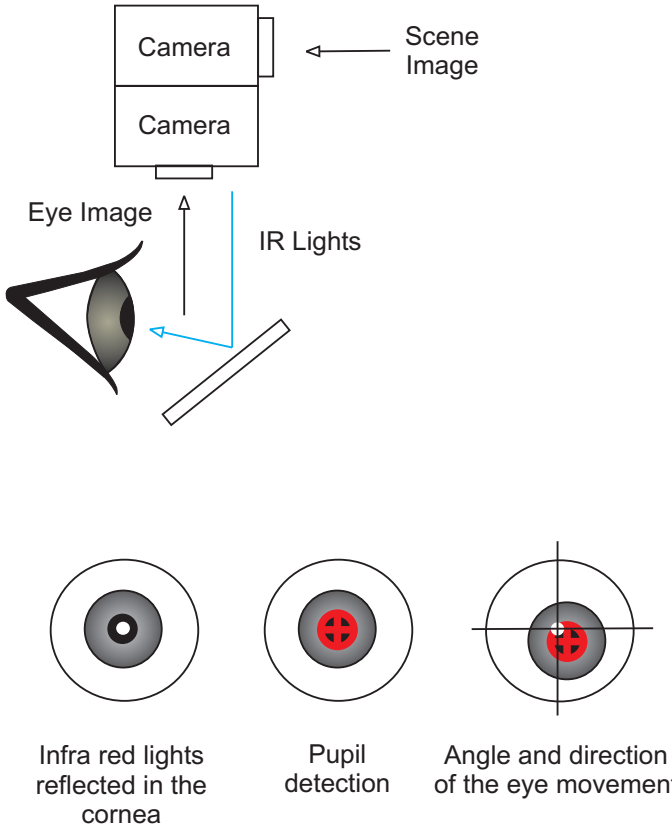


Figure 2.7: Diagram of the gaze tracking acquisition system. During the eye tracking process an infra red (IR) light is projected onto the cornea and it is used as a reference point to compute the position of the pupil when the eye rotates. The gaze point is finally calculated by translating the eye tracking information into the image of the scene given by the frontal camera.

Chapter 3

Dynamic model of a 7-DOF Whole Arm Manipulator

3.1 Introduction

Robot manipulators can be considered as a good representation of the human arm. This characteristic makes them suitable to be used as a platform to develop control strategies that could be implemented in prototypes for future artificial prosthetics. Following our objective of designing a HMI to control a robot arm in reaching and grasping tasks, as it was mention in Chapter 1, we decided to analyse the performance of the manipulator used in the development of this thesis. The manipulator was a 7 degrees of freedom (DOF) whole arm manipulator (WAM) that was configured with joint PD control and gravity compensation. The full description of the system can be found in Section 3.2. We made an analysis of performance based on the identification of the joint position errors presented during the execution of joint rotations, the final position error and the smoothness of the motion during the execution of trajectories. As a result of the analysis we noted that the joint position errors, although very small, never reached a zero value and that some trajectories were not as smooth as others. We proposed to configure the WAM arm to work under joint PID control in order to evaluate if the performance of the manip-

ulator improved. However, tuning the parameters of the joint controllers was not a straightforward process considering the non linearity of the system and the number of degrees of freedom of the manipulator. Under the circumstances we opted for creating a mathematical model that could represent the system accurately in order to facilitate the tuning process.

The computation of the dynamic model of a robot manipulator plays an important role in the simulation of motion, the analysis of the manipulator's structure and the design of optimal control algorithms. Also, the inclusion of the effects of friction in a mechanical system model helps to improve the performance of the controller to be implemented in the real system (Kostic et al., 2004; Indri, 2006). In most cases these mathematical models are implemented using high level computing languages such as MATLAB (Corke, 1996), C/C++ or Fortran. However, MATLAB/SimMechanics offers a good platform for experimental verification avoiding significant computation time. The capability of this tool for analysing the dynamics of mechanical systems yields to suitable results when the model of the system uses joints with individual primitives, and when all the manipulator's inertial parameters are known (MathWorks, 2011).

This chapter describes the design of the dynamic model of the 7-DOF WAM arm from *Barrett Technology Inc.* In order to implement the model we took advantage of the characteristics of Simmechanics as a platform suitable to simulate mechanical systems subject to constrains. The inertial characteristics of the model, the joint control strategy and the joint trajectories are based on the set up of the WAM arm.

3.2 The real system

3.2.1 Description of the system

The system is a 7 degrees of freedom (DOF) whole arm manipulator (WAM) from *Barrett Technology Inc.* It is a joint torque controlled manipulator equipped with configurable PD/PID control and gravity compensation. The information related to the joints configuration, joint motor drives, body part masses, centre of gravity and inertia matrix is provided by the manufacturer in the WAM ARM User's Manual (Barrett Technology Inc, 2008a). The WAM arm was working under the C language based controls library Btclient. This software did not have the ability to control the BH8-280 hand in conjunction with the WAM arm. During the development of the works described in Chapter 5 we actualised the software to a C++ based controls library called Libbarrett which was fully compatible to control the BH8-280 hand. With Btclient the system worked under joint PD control and gravity compensation and with Libbarrett the system worked under PID control and gravity compensation.

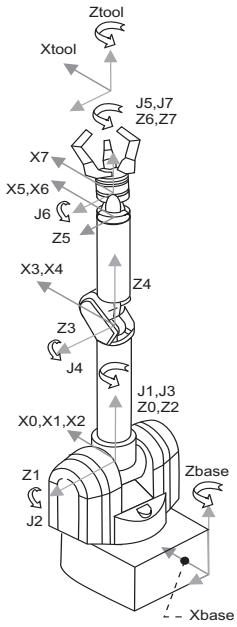


Figure 3.1: WAM 7-DOF Denavit-Hartenberg architecture with attached frames as shown in Barrett Technology Inc (2008a).

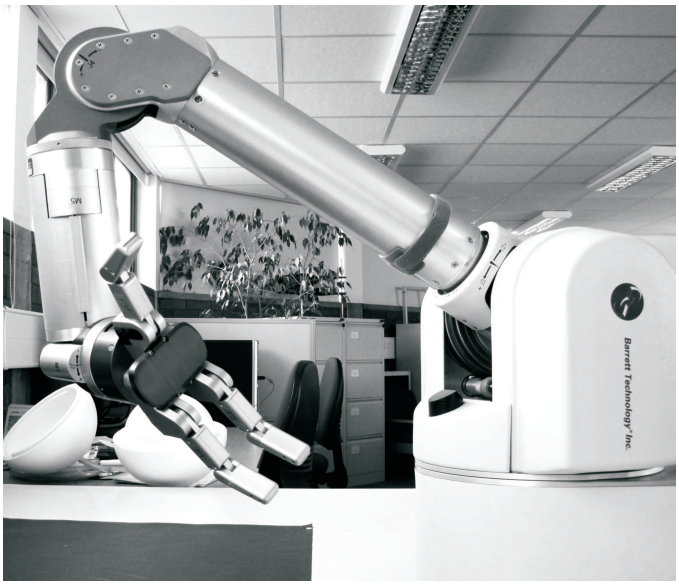


Figure 3.2: The Barret 7-DOF Whole Arm Manipulator with the BH8-series BarrettHand.

Figure 3.1 shows the configuration and attached frames of the 7-DOF system with a grasper. All the joints of the manipulator are 1 DOF revolute joints. An image of the real system is shown in Figure 3.2, consisting in a Barret 7-DOF WAM arm with an attached BH8-series BarrettHand.

For the development of the model the 7-DOF manipulator was configured with joint PD control and gravity compensation given by Equation (3.1), where the joint torque τ is expressed as the sum of the difference between the reference and measured position, namely position error $\tilde{q}$, multiplied by a constant proportional gain K_P, the derivative of the error $\dot{\tilde{q}}$ multiplied by a respective derivative gain K_D and the compensation for gravity g, which is a function of the joint position.

$$\tau = K_P\tilde{q} + K_D\dot{\tilde{q}} + g \tag{3.1}$$

3.2.2 Performance of the system

An initial analysis of the system was made in order to evaluate the performance of the WAM arm in the execution of joint rotations. We considered two sets of

experiments. The first set consisted in rotating every joint of the manipulator one at a time. In the second set of experiments all the joints of the manipulator were rotated together. In both sets of experiments the joints of the manipulator were rotated 0.5 radian at a velocity of 0.083 rad/sec. This angle of rotation was chosen so that it could be executed by all the joints of the manipulator, according to the kinematics of the mechanical structure. During the experiments we registered all the joints reference trajectories and the joint positions. The information gathered was used to compute the existing error between the desired reference position and the actual joint position. The joint position errors helped to analyse the behaviour of the joint controllers in the execution of the joint rotations.

At the end of each set of experiments, a relative percentage error of the final position of the joint was measured using

$$RPE = \left| \frac{\text{Joint Position Error}}{\text{Desired Joint Position}} \right| \times 100\% \tag{3.2}$$

The RPE was used to measure how accurate the joint motion was in reaching the desired end position. We also observed the registers of the actual joint trajectories since they showed the quality of the manipulator's motion during the task, giving that an irregular trajectory was equivalent to a shaky motion of the joint.

The results of the experiments were as follows. Figure 3.3 shows the registers of the reference trajectories and the joint positions obtained in the first set of experiments. It is difficult to have a fair appreciation of the position error in the plots corresponding to Joint 1, Joint 2, Joint 3, Joint 4 and Joint 5 since the differences between the reference trajectory and the position error are very small. The position error of Joint 6 and Joint 7 are more evident in the plots. It can be noticed that the trajectory of Joint 7 is not as smooth as the trajectory of the rest of the joints, this was translated into a shaky motion of the joint when executing the experiment. Figure 3.4 shows the position errors occurring in every joint of the

manipulator during the first set of experiments. It can be noticed in these plots that Joint 6 and Joint 7 present the largest position errors of approximately 0.017 and 0.027 rad, while the position errors of Joint 1, Joint 2, Joint 3, Joint 4 and Joint 5 are within an approximate range of 0.002 and 0.007 rad.
The relative percentage errors (RPE) of the joints of the manipulator computed using Equation 3.2 resulted as follows: Joint 1 had a RPE of 1.0876%, Joint 2 had a RPE of 0.2342%, Joint 3 had a RPE of 0.8666%, Joint 4 had a RPE of 0.2912%, Joint 5 had a RPE of 1.3592%, Joint 6 had a RPE of 3.2848% and Joint 7 had a RPE of 5.4132% .

Figure 3.5 shows the registers of the reference trajectories and the joint positions obtained in the second set of experiments. As in the case of the first set of experiments, it is difficult to have a fair appreciation of the position error in the plots corresponding to Joint 1, Joint 2, Joint 3, Joint 4 and Joint 5 since they are very small. Figure 3.6 shows the plots of the position errors obtained during the execution of the joint rotations. Joint 6 and Joint 7 had the largest position errors with an approximate values of 0.013 and 0.017 rad, while the position errors of Joint 1, Joint 2, Joint 3, Joint 4 and Joint 5 are within an approximate range of 0.001 and 0.006 rad.
The relative percentage errors (RPE) of all the joints of the manipulator obtained during the second set of experiments resulted as follows: Joint 1 had a RPE of 1.0656%, Joint 2 had a RPE of 0.2016%, Joint 3 had a RPE of 0.1732%, Joint 4 had a RPE of 0.2400%, Joint 5 had a RPE of 0.3884%, Joint 6 had a RPE of 2.6214% and Joint 7 had a RPE of 3.6664%.

After analysing the performance of the system under PD control and gravity compensation in the execution of joint rotations we noticed that Joint 2, Joint 3, Joint 4, Joint 5, Joint 6 and Joint 7 presented the largest end position error during the

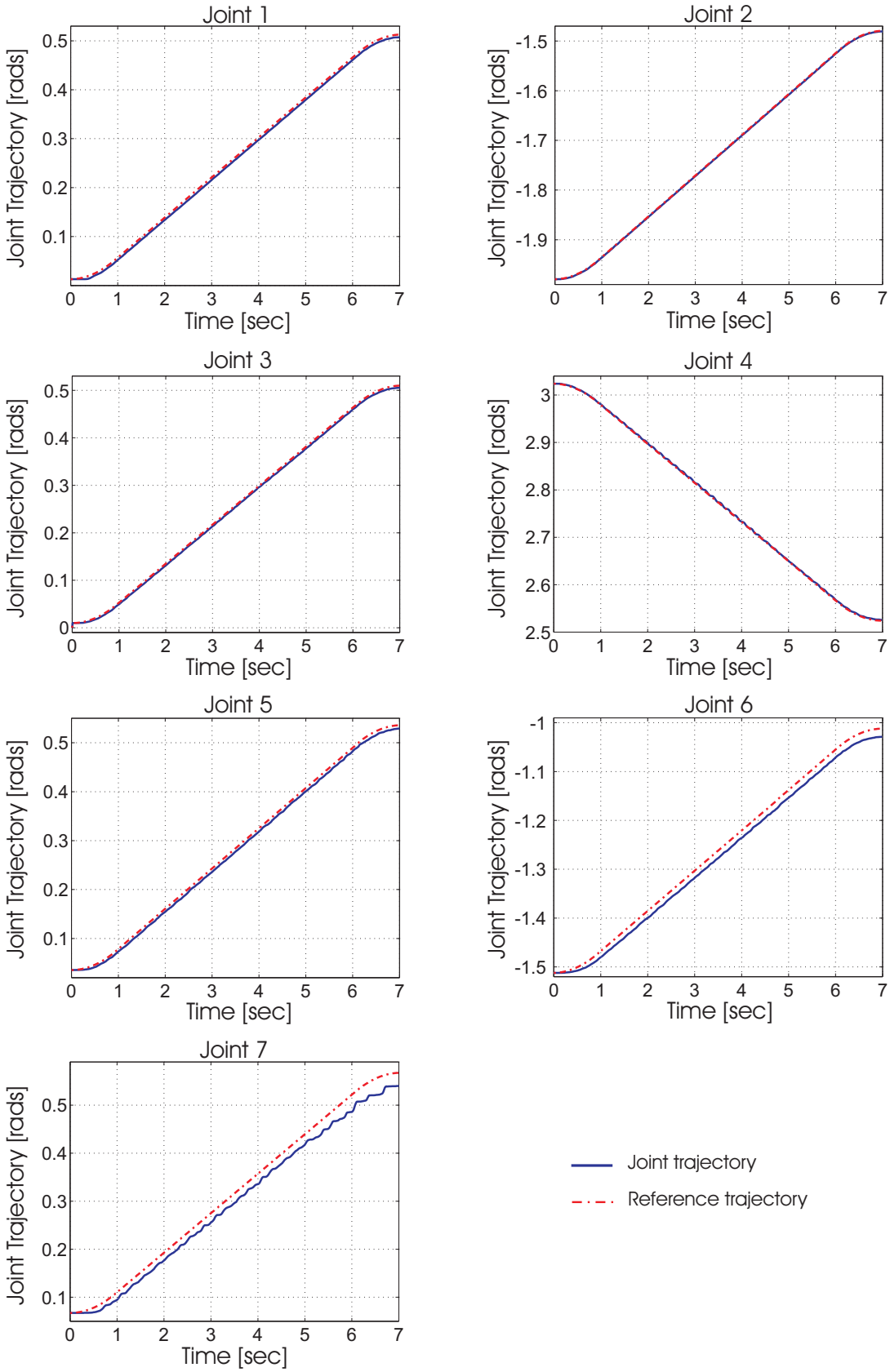


Figure 3.3: Joint trajectories of the 7 joints of the manipulator when executing a rotation of 0.5 rad. The rotations were executed by every joint one at a time.

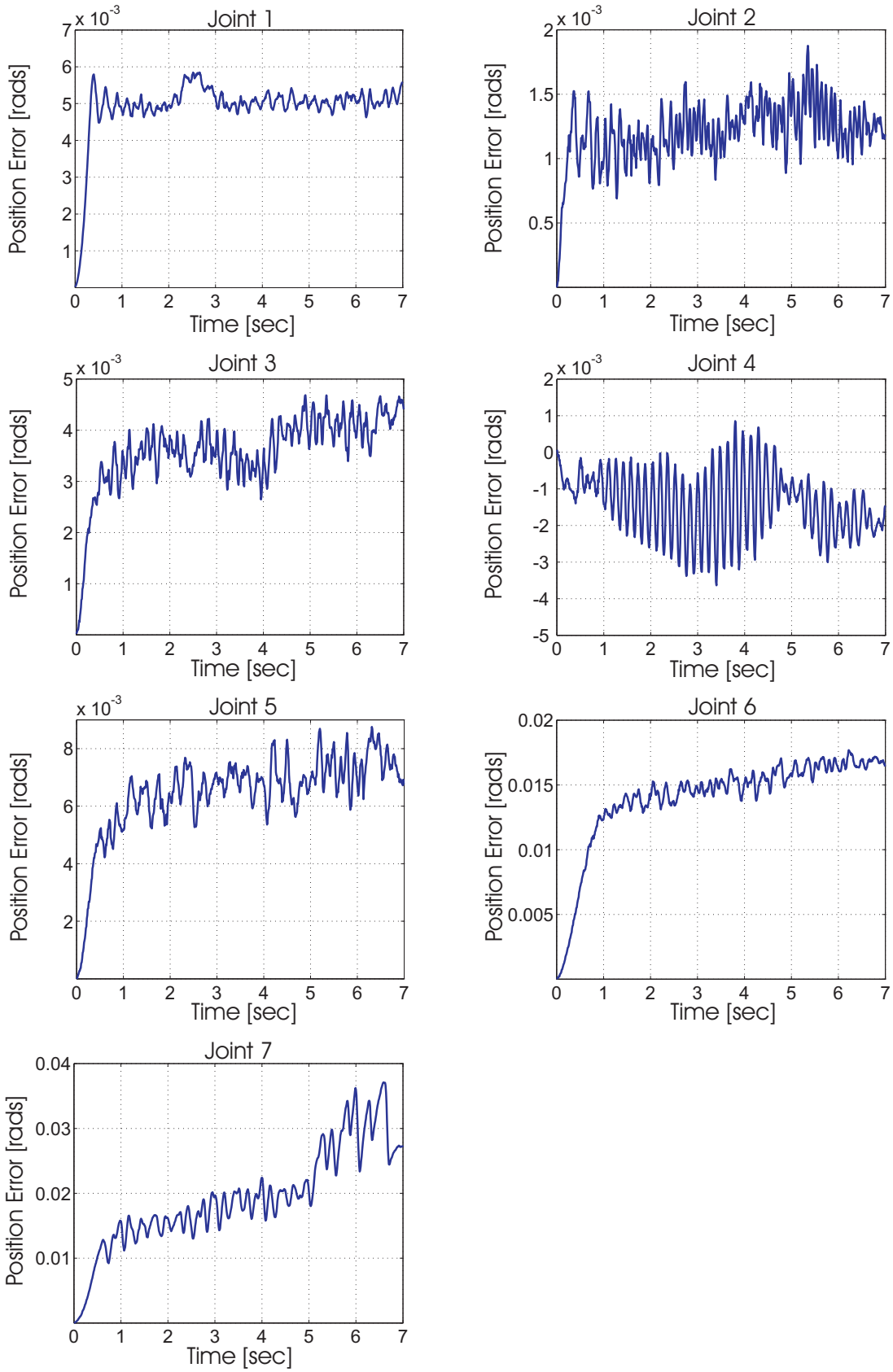


Figure 3.4: Position errors of the 7 joints of the manipulator when executing a rotation of 0.5 rad. The rotations were executed by every joint one at a time.

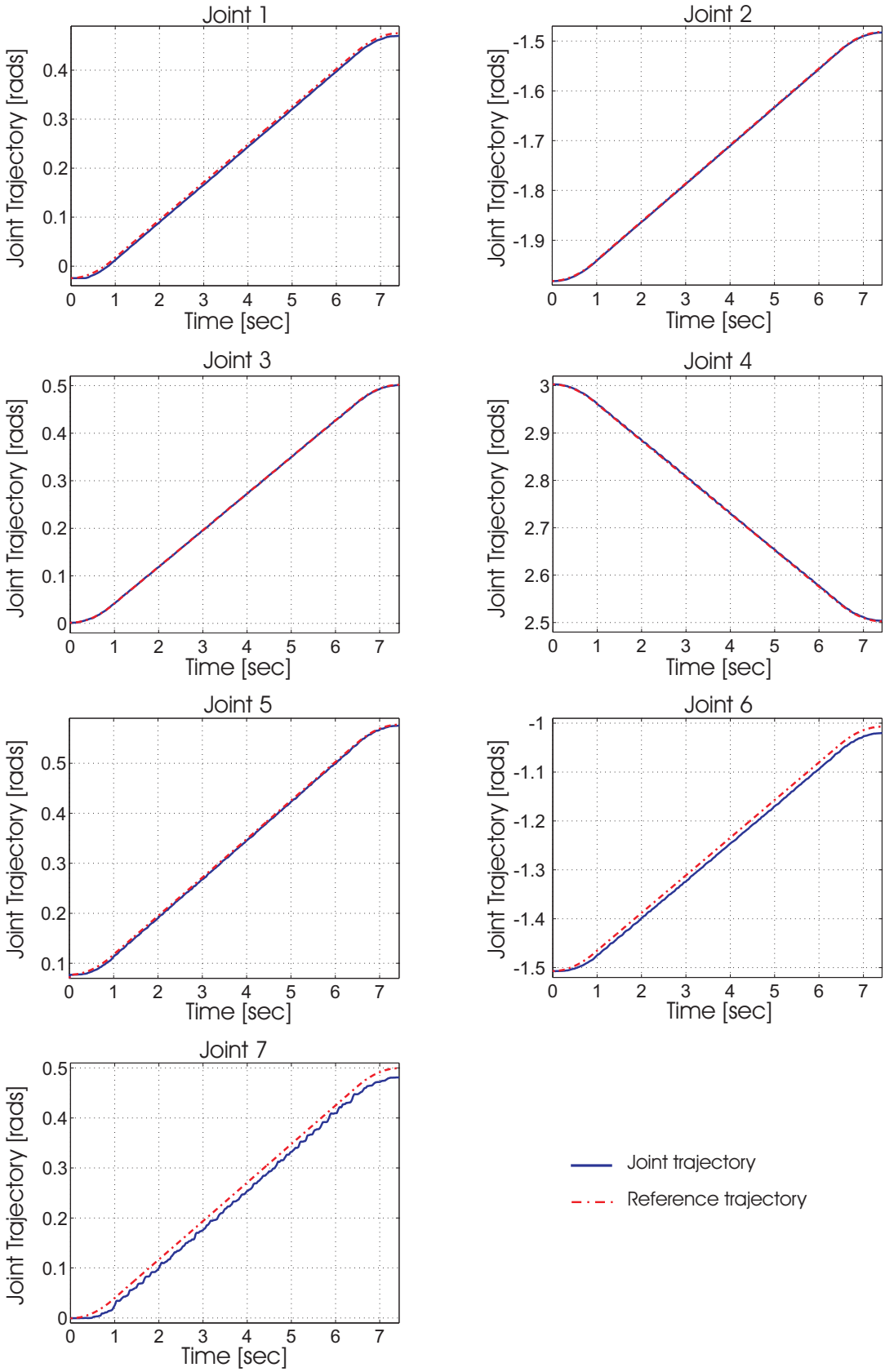


Figure 3.5: Joint trajectories of the 7 joints of the manipulator when executing a rotation of 0.5 rad. The rotations were executed all together.

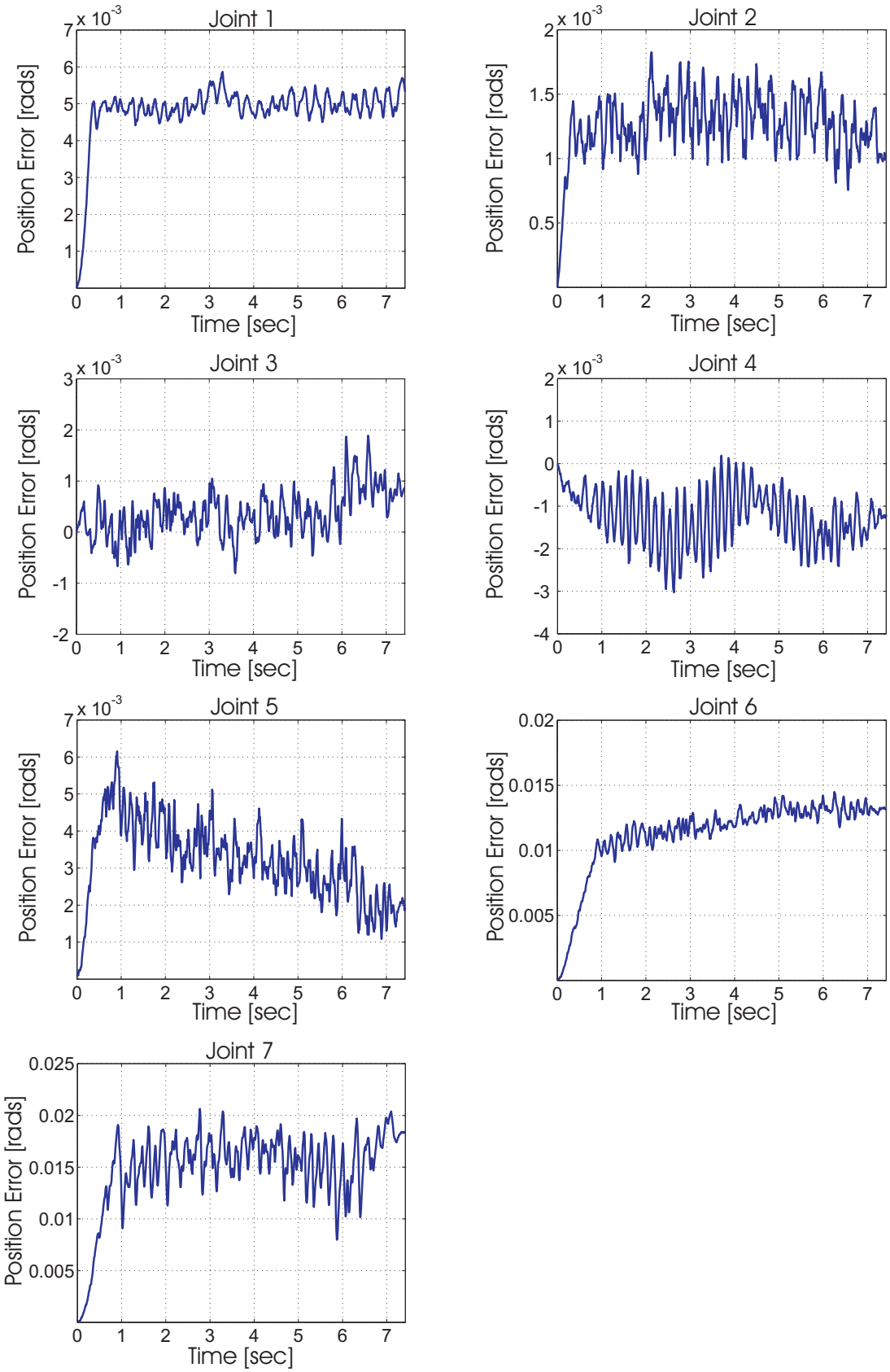


Figure 3.6: Position errors of the 7 joints of the manipulator when executing a rotation of 0.5 rad. The rotations were executed all together.

first set of experiments, when rotating each joint at a time, having a RPE of 0.079%, 0.849%, 0.057%, 1.268%, 1.622% and 4.771%, respectively. Joint 1 presented a larger error during the execution of the second set of experiments, when all the joints were rotated together, having a RPE of 1.120%. It is evident that the performance of the manipulator is good in general. However, due to the manipulator is intended to be used in neuroprosthetic applications, we are looking to reduce the motion errors as much as possible. Also, in the case of Joint 6 and Joint 7 we looked forward for obtaining smoother rotations that were able to reduce the shaky motion seen during the experiments.

As mentioned, the manipulator was intended to be used in future applications of neuroprosthetics. In some of these applications high accuracy is imperative so we proposed to use Proportional Integral Derivative (PID) control in the WAM arm with the aim of reducing the RPE to less than 1% in all the joints of the manipulator. In robot vision applications, for example, the system relies on a camera to find the position of objects and uses the computation of the object's centre of mass in order to define a grasping strategy, so a high level of accuracy in the end positioning of the manipulator is needed. Our second objective was to obtain smoother joint trajectories during the execution of tasks.

3.3 The Model of the system

Considering the previous description of the real system, the 7-DOF WAM model consisted of four blocks:

- Seven modules that compute the joint reference trajectories.
- The dynamic model of the system.
- Fitted friction models for each joint of the manipulator.
- Joint PD controllers configured and tuned as those implemented in the real system.

The dynamic model of the manipulator is based on the configuration and physical characteristics of the real system; the parameters of the friction model are determined through the identification of the joint frictions whilst the joint trajectory and joint controller are emulations of those existent on the real system.

3.3.1 Trajectory Generation

The reference signal used to perform every joint rotation is a linear segment parabolic blend (LSPB) trajectory defined as follows: up to a time t_c the trajectory is parabolic with linear velocity, at t_c the trajectory changes to linear with constant velocity and zero acceleration and finally, after a time $(t_f - t_c)$ the trajectory changes to parabolic again and the velocity decreases linearly until reaching zero. The motion produced when applying this velocity profile to any joint is translated in a rotation from the initial joint position q_i of the manipulator to q_f radian. Equation (3.3) defines the described trajectory, where q_i is the initial position, q_f is the final position reached in a time t_f, $\dot{q_c} = \ddot{q_c} t_c$ is the value of the constant velocity exhibited from time t_c to time $(t_f - t_c)$ and $\ddot{q_c}$ is the value of the desired constant acceleration.

$$q(t) = \left\{ \begin{array}{ll} q_i + \frac{1}{2}\ddot{q_c}t^2 & 0 \leq t \leq t_c \\ \frac{1}{2}(q_f + q_i - \dot{q_c}t_f) + \dot{q_c}t & t_c < t \leq t_f - t_c \\ q_f - \frac{1}{2}\ddot{q_c}(t_f - t)^2 & t_f - t_c < t \leq t_f \end{array} \right\} \tag{3.3}$$

An example of the trajectory given by Equation (3.3) can be seen in Figure 3.7 where a rotation of 0.5 rad is performed.

In order to generate the LSPB joint trajectory, the real system sets the value of the time of change t_c as 1, and the velocity $\dot{q_c}$ and acceleration $\ddot{q_c}$ are calculated using Equation (3.4) and Equation (3.5), taking into account the measured values of the initial and the final joint position q_i and q_f. In simulations, the joint trajectory

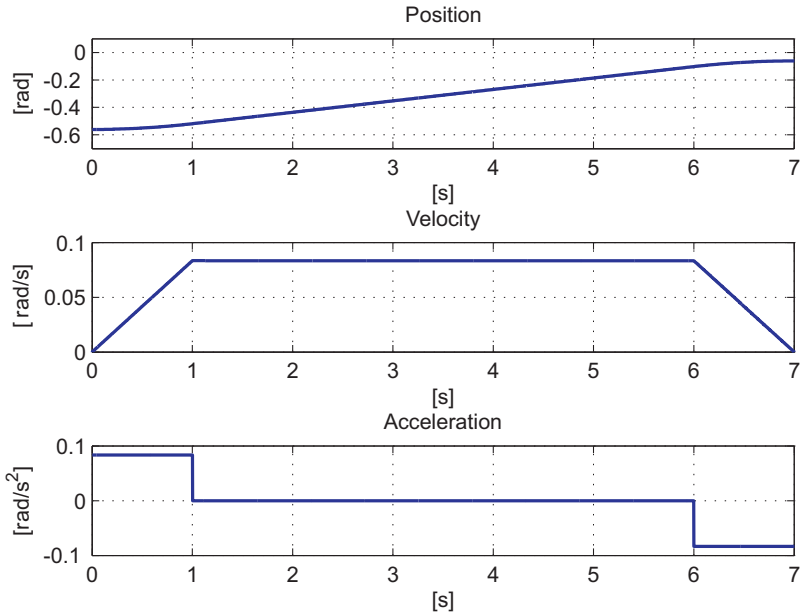


Figure 3.7: Example of a trapezoidal velocity profile when the values of the variables are set as follows $q_i = -0.55$, $q_f = -0.05$ rads, $\ddot{q_c} = 0.083$ rads/s, $t_f = 7$s and $t_c = 1$.

is emulated by the model considering the experimental values of the initial joint position q_i, the final joint position q_f and the execution time t_f. The value of t_c is set as 1 and the velocity and acceleration are also calculated using Equation (3.4) and Equation (3.5). Finally, taking all the parameters previously estimated, the joint trajectory is generated using Equation (3.3).

$$\dot{q_c} = \frac{4(q_f - q_i)}{t_f^2 - (-2(t_c - 0.5t_f))^2} \tag{3.4}$$

$$\ddot{q_c} = \dot{q_c} \tag{3.5}$$

3.3.2 The Dynamic Model

The dynamic model of the WAM Arm was designed and simulated using MATLAB/SimMechanics, and most of the inertial data provided by the manufacturer had to be adapted according to the inertial reference system. The MATLAB/SimMechanics toolbox is a block diagram modelling environment like Simulink, created by MATLAB, to design and simulate mechanical systems. The

toolbox contains several modules that represent particular bodies and which inertial properties can be specified by the user. A great advantage of using SimMechanics in modelling mechanical systems is that the toolbox is prompt to be used with Simulink so that control routines can be added with ease in order to analyse the behaviour of the system's dynamics under motion constraints.

According to the Barrett WAM arm datasheet the system has zero backlash and near-zero friction. First, we designed the model of the whole arm manipulator without considering the friction affecting the joints and later we compared the performance of the WAM arm with the performance of the model in the execution of joint rotations. As those used for the initial analysis of performance of the WAM arm described in Section 3.2.2, two sets of experiments were made. The first set consisted in rotating every joint of the manipulator one at a time. In the second set of experiments all the joints of the manipulator were rotated together. In both sets of experiments the joints of the manipulator were rotated 0.5 radian at a velocity of 0.083 rad/sec.

Figure 3.8 shows the registers of the reference trajectories and the joint positions obtained in the first set of experiments. According to the plots, the model trajectories were close to the reference trajectories, meaning that the model had smaller position errors compared to the real system. The plots of the joint position errors shown in Figure 3.9 support the later, the values of the model position errors did not approach the values corresponding to the real system position errors. The registers of the reference trajectories, the joint positions and the computation of the joint position errors obtained during the second set of experiments are shown in Figure 3.10 and Figure 3.11. The results are similar to those obtained during the first set of experiments, the model trajectories were closer to the reference trajectories rather than the real joint positions and the values of the model position errors did not approach the values corresponding to the real system position errors.

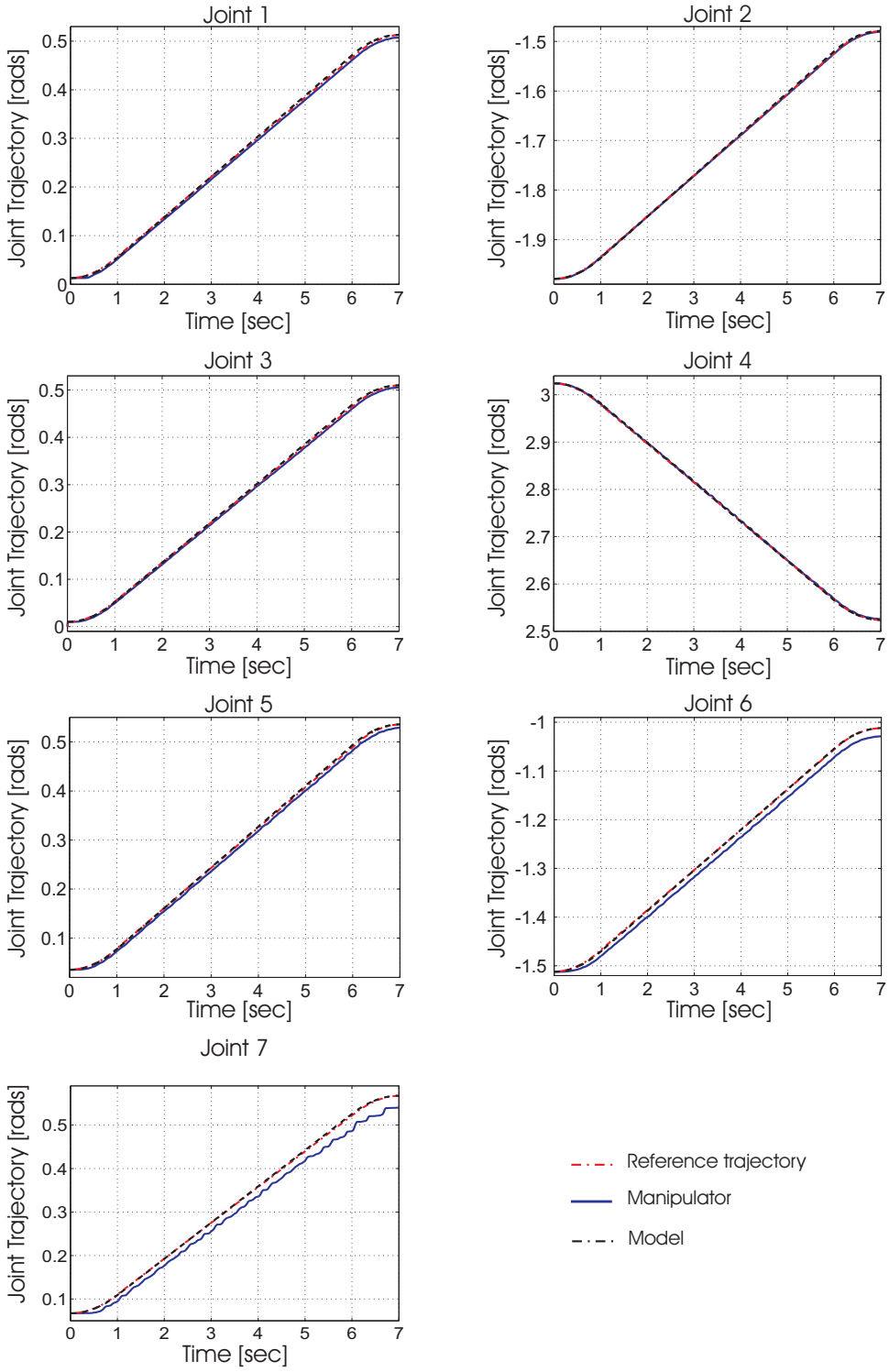


Figure 3.8: Joint trajectories of the 7-DOF manipulator when executing a rotation of 0.5 rad. The graphs show the response of the real system compared with the response of the model. The joint rotations were executed one joint at a time.

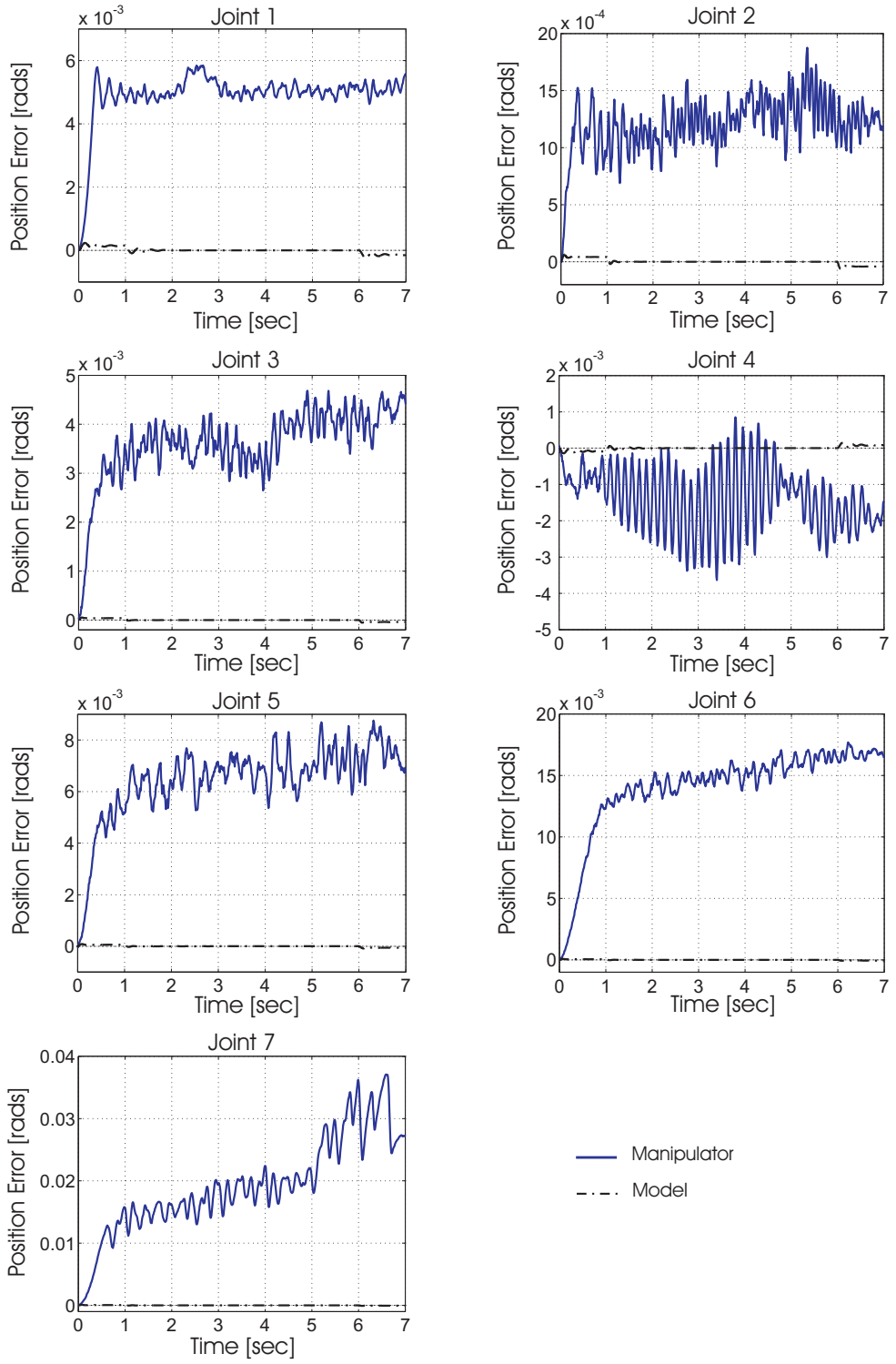


Figure 3.9: Position error of the 7-DOF manipulator when executing a rotation of 0.5 rad. The graphs show the response of the real system compared with the response of the model. The joint rotations were executed one joint at a time.

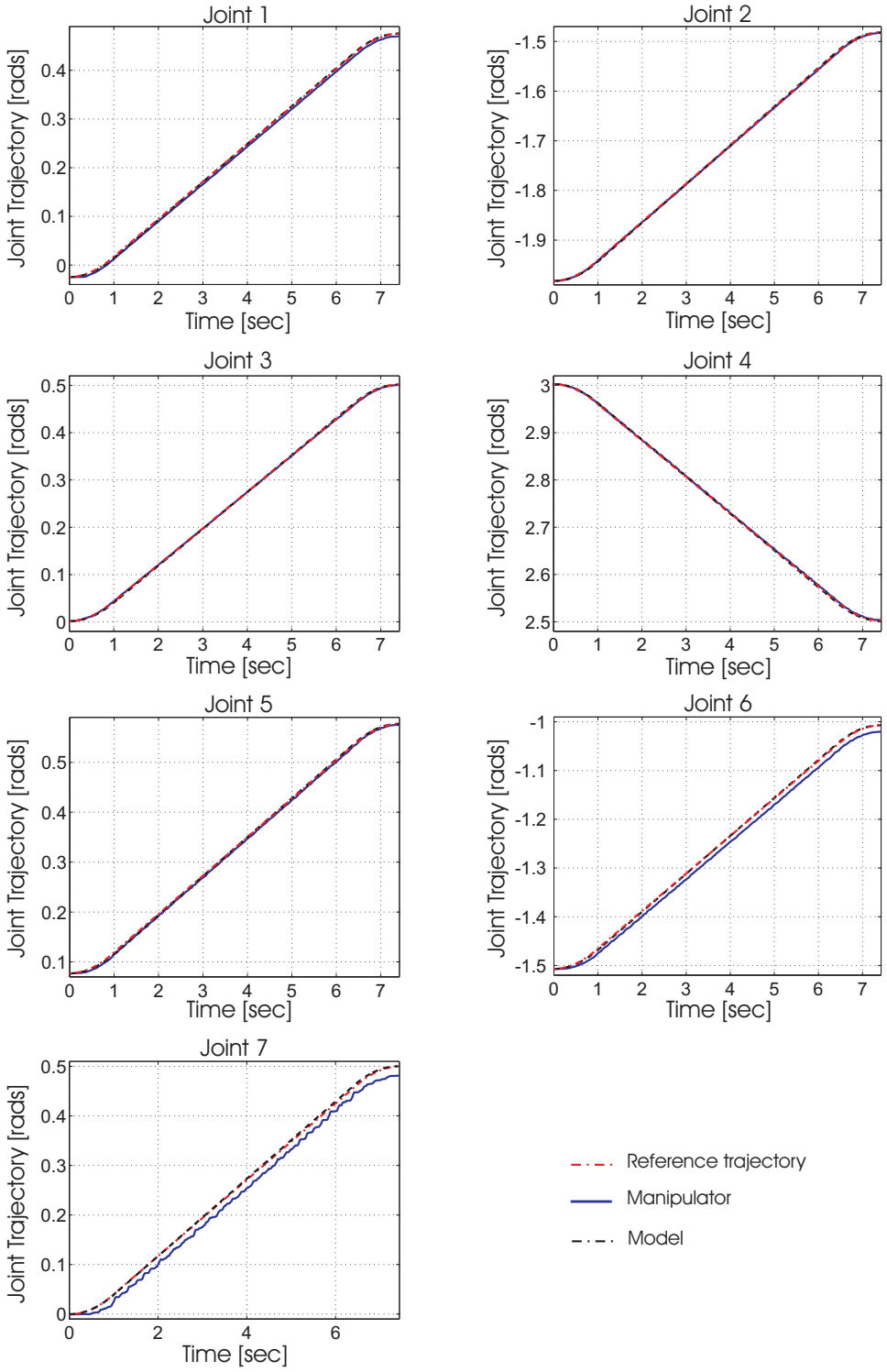


Figure 3.10: Joint trajectories of the 7-DOF manipulator when executing a rotation of 0.5 rad. The graphs show the response of the real system compared with the response of the model. The joint rotations were executed all together.

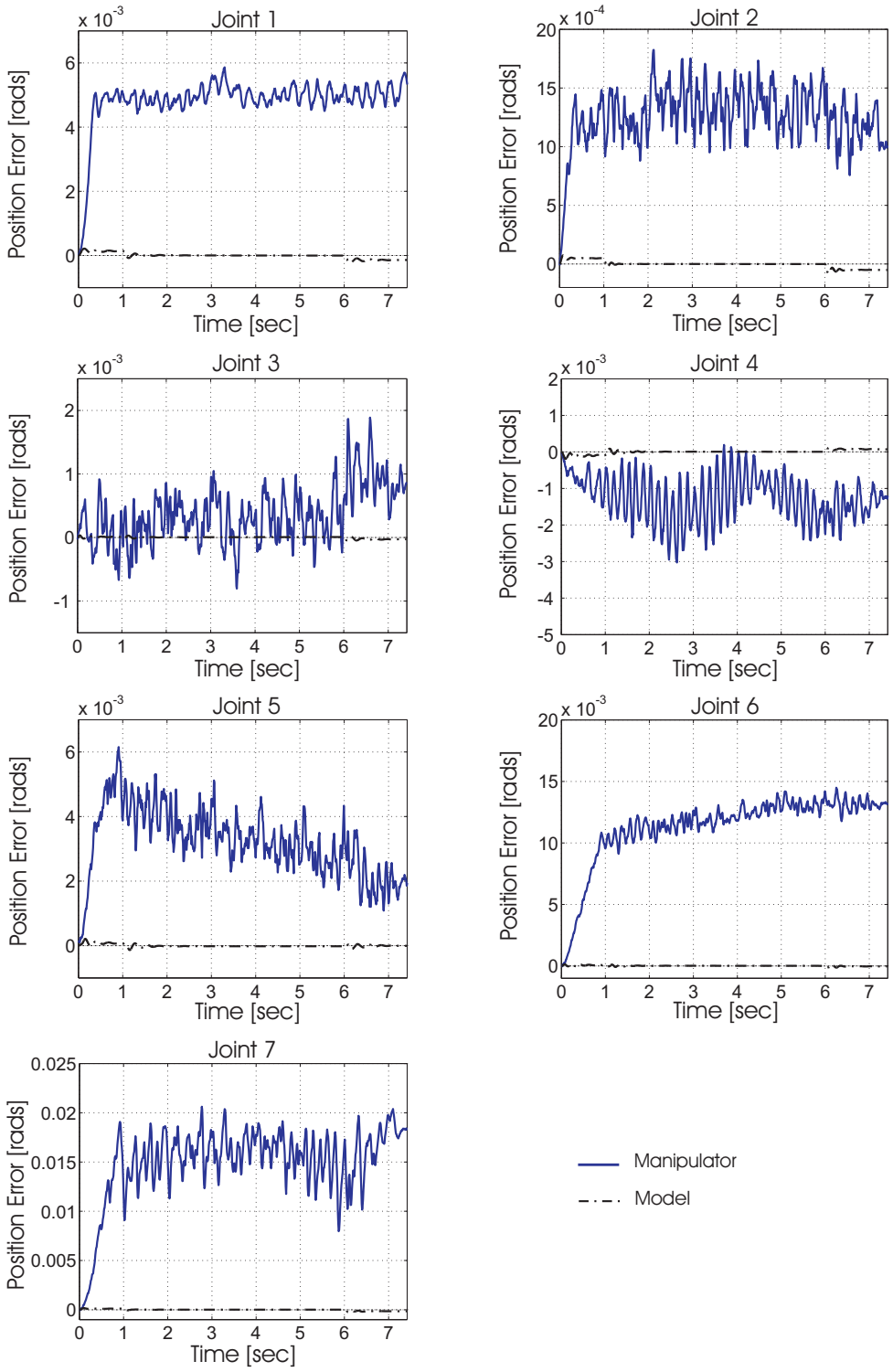


Figure 3.11: Position error of the 7-DOF manipulator when executing a rotation of 0.5 rad. The graphs show the response of the real system compared with the response of the model. The joint rotations were executed all together.

Since there are clear differences when comparing the performance of the model without friction against the performance of the real system, we decided to consider the friction forces affecting the joints of the manipulator in order to increase the accuracy of the model with respect to the real manipulator.

3.3.3 The model with Friction

The classic model of friction is a good representation of friction when the system moves at medium low and medium high velocities. The classic friction model includes two parameters: the Coulomb friction parameter F_c and the viscous friction parameter σ_2. There is a simple method to find the value of the parameters using real data. The method is based on several experiments were the joints of the manipulator are rotated at several velocities, positive and negative. Considering the torque necessary to perform the desired rotation, a torque versus velocity plot is constructed, this plot is known as friction velocity map. The process of friction identification is fully explained in Chapter 4. The classic friction model is expressed mathematically as

$$F = F_c sign(\dot{q}(t)) + \sigma_2 \dot{q}(t) \tag{3.6}$$

where $\dot{q}(t)$ is the velocity.

The parameters used to construct the friction model for each joint of the 7-DOF whole arm manipulator are listed in Table 4.1. The friction models were added to the model of the whole arm manipulator and its performance was compared to the performance of the WAM arm in the execution of joint rotations, following the same methodology described in Section 3.2.2 and Section 3.3.2.

Two sets of experiments were made, the first set consisted in rotating every joint of the manipulator one at a time and the second set of experiments consisted in rotating all the joints of the manipulator together. In both sets of experiments

the joints of the manipulator were rotated 0.5 radian at a velocity of 0.083 rad/sec. Figure 3.12 shows the registers of the reference trajectories and the joint positions obtained in the first set of experiments, when every joint of the manipulator was rotated individually. The model trajectories of Joint 1, Joint 2, Joint 3 and Joint 4 were very close to the real trajectory but since the position errors were very small this similarity was not evident in the plots. In the plots of Joint 5, Joint 6 and Joint 7 the model-real system similarity was more evident since the position errors in this joints were a bit larger. Figure 3.13 shows the joint position errors. The values of the model errors approached the values corresponding to the real system errors in all the joints of the manipulator while executing the joint rotations.

Figure 3.14 shows the registers of the reference trajectories and the joint positions obtained in the second set of experiments, when all the joints were rotated together. It is difficult to see the accuracy of the model versus the real system in the plots of Joint 1, Joint 2, Joint 3, Joint 4 and Joint 5. The joint trajectories obtained in Joint 6 and Joint 7 are evidently similar to the joint trajectories of the real system. Figure 3.15 shows the computed position errors. The position errors of the model are similar to the position errors of the real system in Joint 1, Joint 2, Joint 4, Joint 5, Joint 6 and Joint 7, but larger in the case of Joint 3.

3.4 Results

Several joint rotational motion experiments were performed with the aim of evaluating the accuracy of the model with respect to the real manipulator. First, we programmed the real system to execute the desired rotation while keeping recordings of the joint reference trajectory and the joint positions. We took the recorded data to compute the position error. Subsequently, we emulated the joints reference trajectory of the real system in the model and we ran a simulation of the joints rotation while keeping a record of the joints position error. The response of the

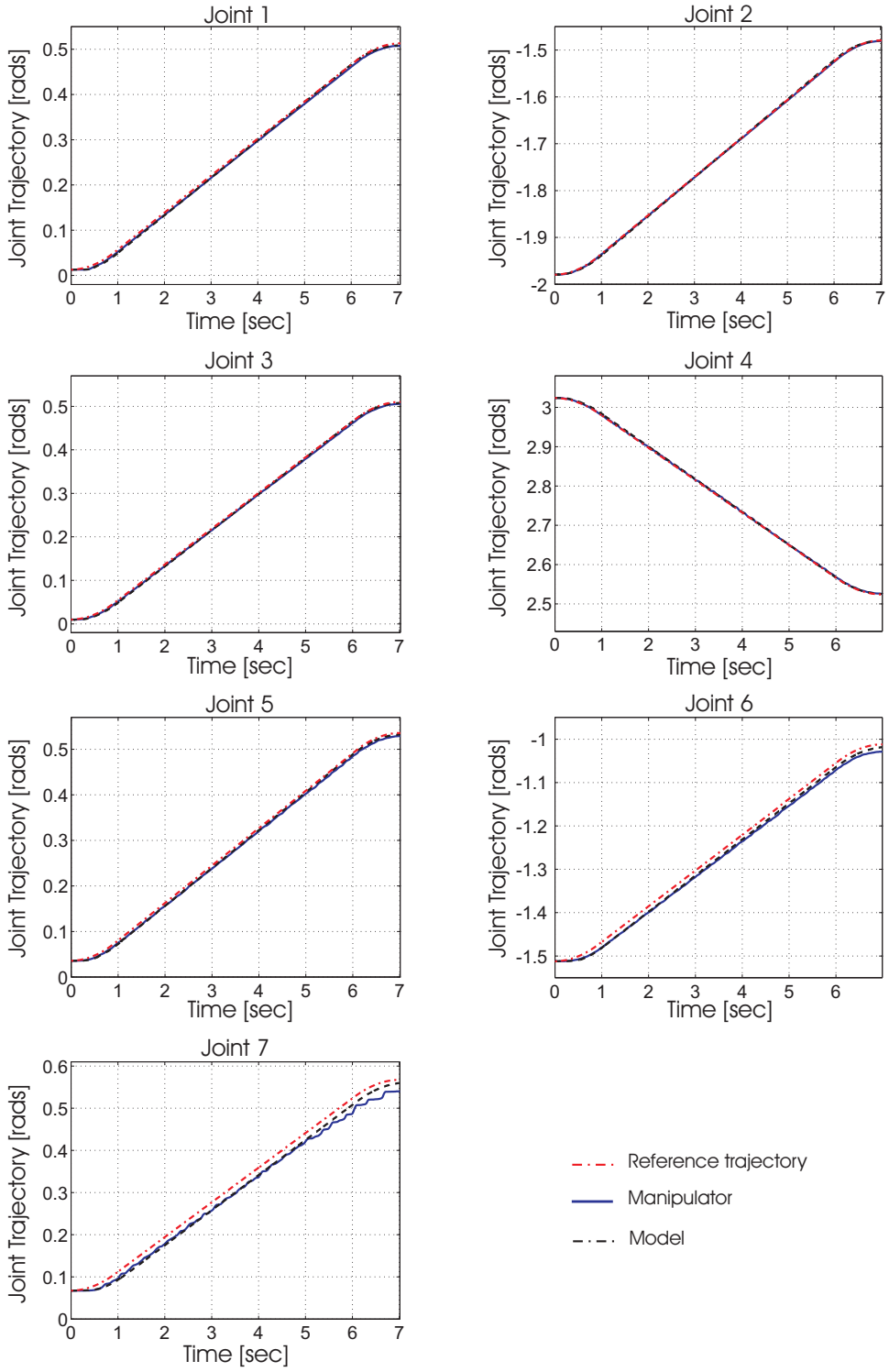


Figure 3.12: Joint trajectories of the 7-DOF manipulator when executing a rotation of 0.5 rad. The graphs show the response of the real system compared with the response of the model with friction. The joint rotations were executed one at a time.

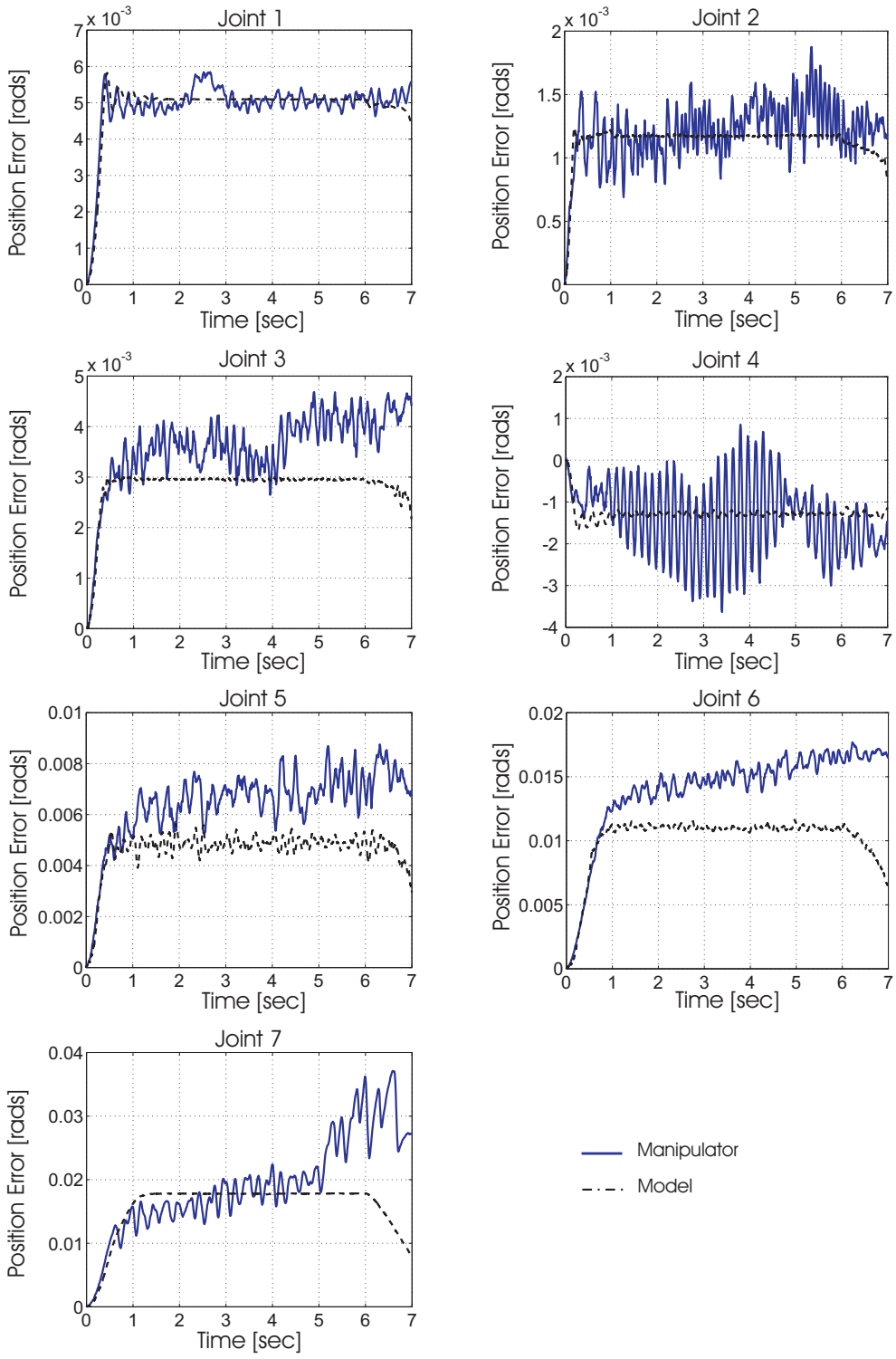


Figure 3.13: Position errors of the 7-DOF manipulator when executing a rotation of 0.5 rad. The graphs show the response of the real system compared with the response of the model with friction. The joint rotations were executed one at a time.

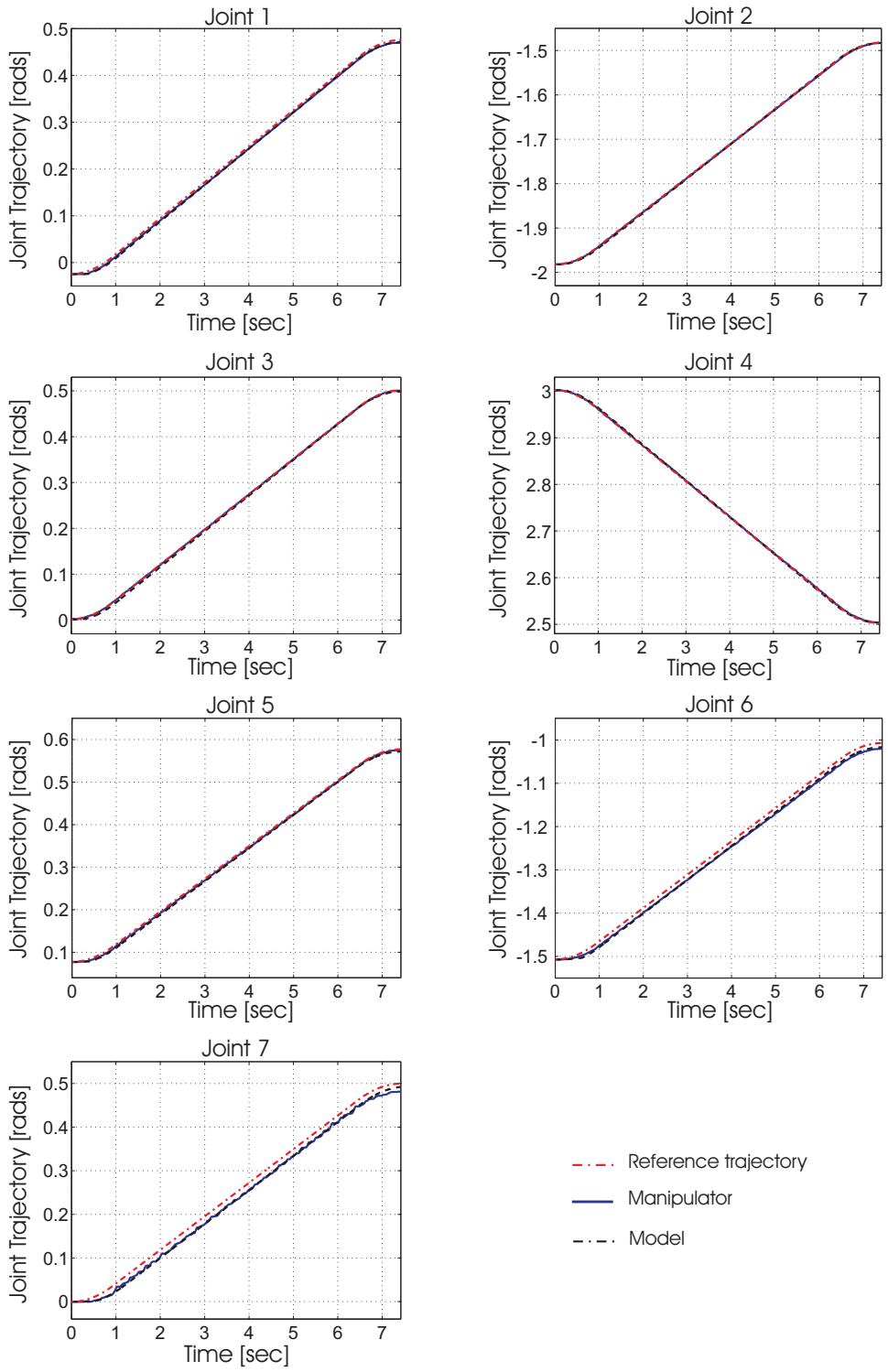


Figure 3.14: Joint trajectories of the 7-DOF manipulator when executing a rotation of 0.5 rad. The graphs show the response of the real system compared with the response of the model with friction. The joint rotations were executed all together.

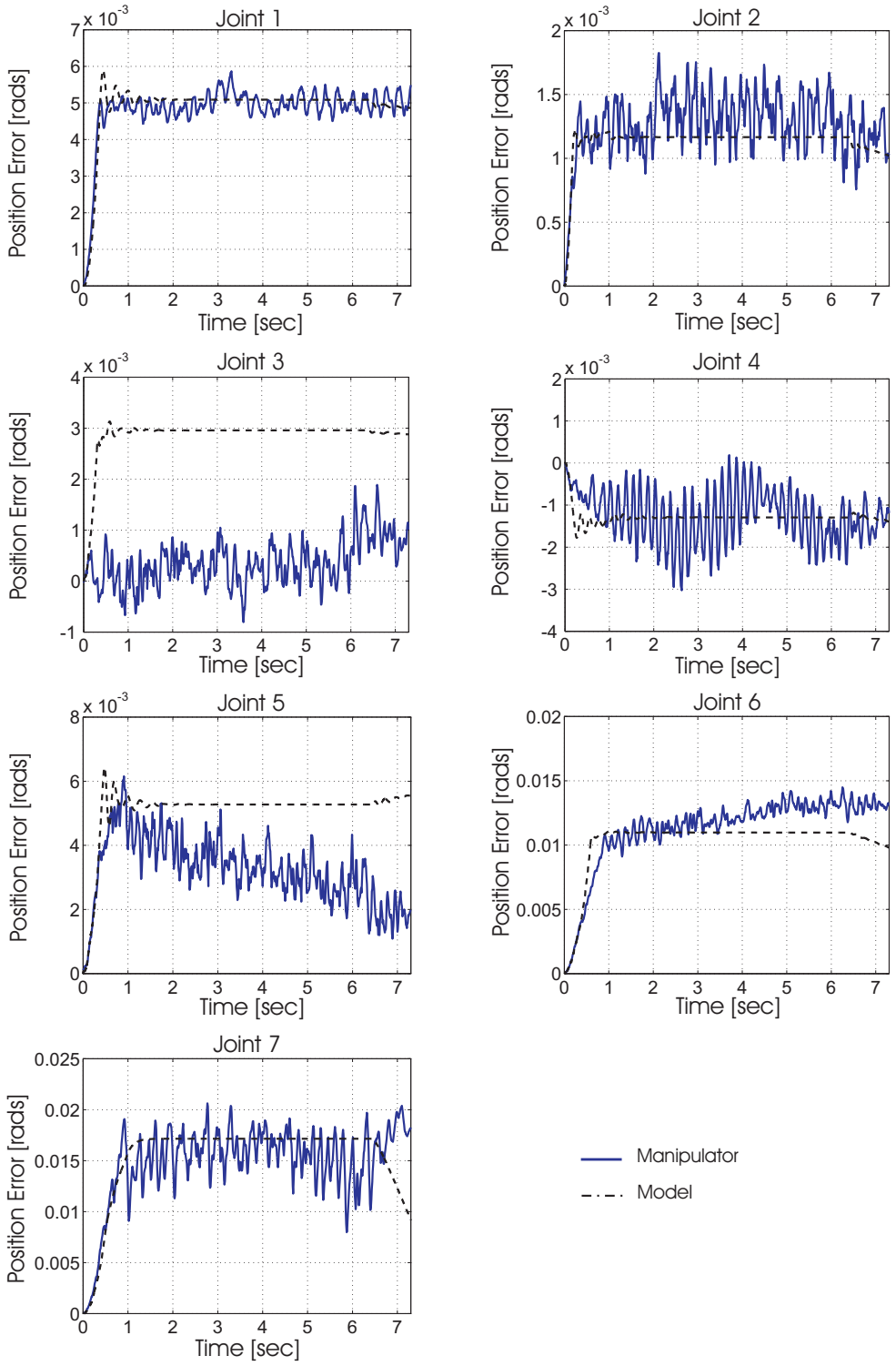


Figure 3.15: Position error of the 7-DOF manipulator when executing a rotation of 0.5 rad. The graphs show the response of the real system compared with the response of the model with friction. The joint rotations were executed all together

model was compared to the response of the system considering the value of the position errors. The position error was chosen as a performance measurement in order to have a closer comparison of the response of both the real manipulator and the model when executing the exact same trajectory since, as seen in the initial performance evaluation of the system, the joint positions were very close to the reference.

The real system uses joint PD controllers with gravity compensation, and therefore, throughout the simulations joint PD controllers were implemented in the model of the system and zero gravity was assumed. The joint rotation used in all the experiments consisted in a rotation of 0.5 rad at a velocity of 0.083 rad/sec.

A total of eight experiments, divided in two sets, were performed. During the first set of experiments every link of the manipulator was rotated one at a time, giving a total of seven simulations. In the second set only one experiment was performed, in which all the joints were rotated at the same time, hence only one simulation was needed. The second set of experiments was made to evaluate reliability, because in real applications the manipulator operates in Cartesian space and all the joints rotate at the same time.

The Barrett WAM arm datasheet states that the friction in the joints of the manipulator is nearly zero. We implemented a model of the system that did not considered the friction affecting the joints of the manipulator. The graphs in Figure 3.8 and Figure 3.9, show a comparison of the position error obtained in the first set of simulations. The plots show that the performance of the model is not a fair representation of the real system, since the joint positions measured in the model are more similar to the reference trajectory than to the joint positions of the real system, and the position errors in the real system are larger than those observed in the model.

Similar results were obtained during the second set of simulations as shown in Figure 3.10 and Figure 3.11.

Since the performance of the model without friction was not accurate with respect to the real system, we included classic friction models for all the joints of the manipulator. The plots in Figures 3.12 and 3.13 show a comparison of the position error obtained in the first set of simulations. The real and simulated joint position errors are very similar in Joints 1, Joint 2, Joint 4 and Joint 5 so that we could think of the real signal as a noisy representation of the simulated signal. This peculiarity is expected since the measurements taken from the real manipulator are exposed to several sources of noise such as transducer noise and sensor errors. The real position error in Joint 3 is a bit larger that the one obtained in the simulation. Opposite to Joint 3, the real position error in Joint 6 is slightly smaller in comparison with the simulated position error. In the case of Joint 7, the simulated position error is similar to the real position error during the first 4 seconds, after that the real position error increments progressively while the simulated position error remains constant and decreases during the last second of the Joint rotation.

The evaluation of the joint position error obtained during both the real and simulated system helped to prove the veracity of the model with friction joint per joint and to show that the response of this model was accurate.

Figure 3.14 and Figure 3.15 show the results of the second set of simulations. The position error of Joint 1, Joint 2 and Joint 5 are the same for both the real and the simulated systems. In Joint 3, the simulated position error is larger that the real position error. In Joint 4, the real position error is slightly larger that the simulated position error during most part of the joint rotation, and they are almost equal at the end of the rotation. Similarly, in joint 6, the end value of the simulated position error and the real position error are almost the same but they are different during the execution of the joint trajectory. The results obtained when rotating Joint 7 are the opposite of the last two cases, during the fist 6 seconds of rotation the values of the real and simulated position error are very similar but change in the last second of the trajectory.

3.5 Conclusions

This chapter described the dynamic model of a 7-DOF whole arm manipulator with friction implemented in SimMechanics. After performing a motion analysis in the WAM arm we found that the performance of the manipulator is good in general, since the joint position errors are very small when the manipulator is executing motion tasks. However, due to the manipulator is intended to be used in neuroprosthetic applications, we are looking to reduce the motion errors as much as possible and to obtain smoother rotations able to reduce the shaky motion seen during the execution of the experiments. We proposed to use joint PID control to improve the performance of the manipulator. In order to facilitate the tuning process of the seven PID controllers parameters we created a mathematical model of the WAM arm. We took advantage of the mechanical tools available in SimMechanics and its promptness to interact with Simulink to develop a platform to simulate the performance of the real robot manipulator using its dynamic model. The platform included the seven PD/PID controllers, configured exactly as in the real system, used to rotate the joints of the robot arm. The platform also included seven modules that were capable to emulate perfectly the joint reference trajectories used by the real system to feed the joint controllers when executing joint rotations. Although the WAM arm datasheet mentions that the joints of the robot manipulator have almost zero friction, in Section 3.3.2 we observed that the implementation of a model without considering friction was not accurate with respect to the real system. Then, the friction phenomena manifested in each joint of the real manipulator was identified through the execution of several joint rotations at different velocities, in order to find the parameters of the joint friction models using a least-squares minimisation method (this technique is explained in Chapter 4). The mathematical representation of the friction phenomena was added to the dynamic model of the robot manipulator. We compared the performance of the model with friction with the WAM arm in the execution of a motion task. Considering that the real joint trajectory was

completely emulated in the simulations, a close comparison of the performance of the manipulator during simulation with respect to experimental data was possible. The later allowed to test the veracity of the model throughout a set of simulations. In Section 4.4 we showed that the response of the model of the WAM arm with the inclusion of friction was very close to the response of the real system, despite of the disturbances at which the real system is exposed, such as non linearities caused by the joint motors and the inertia of the system. Other sources of disturbance to be considered are the joint actuators and the scheme used for gravity compensation.
The presented dynamic model with friction is prompt to be used as a reference in the design and implementation of different joint control strategies and friction compensation modules for the WAM arm. The platform may be useful to analyse the suitability of the implementations in simulations without compromising the safety of the system.

Chapter 4

Feedforward Friction Compensation in the 7-DOF WAM Arm

4.1 Introduction

In this Chapter we present the implementation of a feedforward compensation technique applied to the joint PD control scheme of the 7-DOF WAM arm. While implementing the mathematical representation of the WAM arm described in Chapter 3, we used a technique to identify the friction phenomena in the manipulator with the aim of obtaining its mathematical model. Since the friction models were pretty accurate with respect to the measured friction phenomena we proposed to implement a feedforward compensation technique, with the objective of improving the performance of the manipulator by reducing the joint position errors and to smooth the motion of the manipulator when performing trajectories. The implementation of the friction compensation module in the joints of the WAM arm would help to analyse the influence of friction force in the accuracy of the joints when executing a motion task.

Friction phenomena in robot manipulators may affect the accuracy of the system

in position control and when moving the manipulator at very low velocities. In control applications, friction compensation may be useful to improve the transient performance and to reduce the steady-state tracking errors, ensuring a smooth control signal (Kermani et al., 2005, 2007). Friction may be described as a function of velocity and other external factors using non linear models.

There are several models used in friction research. Armstrong and colleagues published a survey where all the theory behind friction phenomena is explained. They proposed an integrated friction model based on seven parameters that include the pre-sliding displacement, Columb, viscous and Stribeck curve friction and friction level at breakaway (Armstrong-Helouvry et al., 1994). Canudas de Wit and colleagues proposed a dynamic model considering the contact surfaces as contact between bristles (Canudas de Wit et al., 1995) and illustrated the procedure of identification of friction phenomena to validate the model using experimental data (Canudas de Wit and Lischinsky, 1997). Hensen and colleagues presented two grey-box models and validated the models using experimental data obtained from a rotating arm (Hensen et al., 2000).

The identification of friction phenomena in a manipulator can be achieved by performing motion experiments where the joints of the manipulator are moved at several constant velocities. The data gathered from the experiments is fitted to a friction model using a least squares minimisation method (Borsotto et al., 2009; Calvalho-Bittencourt and Gunnarsson, 2009).

4.2 Friction Identification

Friction force is inherent of the velocity of the joint and it is frequently represented by a mathematical model that includes the sliding friction (or Coulomb friction), the breakdown friction (or stiction) and the viscous friction. In most cases if the manipulator is expected to displace at medium or medium-high velocities, the fric-

tion can be modelled considering the effects of the viscous friction and the Coulomb friction only. This simplified representation of friction is known as classic friction model

$$F = F_c sign(\dot{q}(t)) + \sigma_2 \dot{q}(t) \tag{4.1}$$

where F_c in the parameter for Coulomb friction, σ_2 is the viscous damping coefficient and $\dot{q}(t)$ is the velocity of the link. The value of the parameters for the classic friction model can be obtained from experimental data through the execution of joint rotations at different constant velocities.

The procedure for the identification of the classic friction model parameters for each joint of the 7-DOF WAM arm followed several steps. First a closed loop PD control with gravity compensation was used to move the link following a trapezoidal velocity profile at different positive and negative velocities. During the experiments we kept registers of the joint position and the joint torques, in addition with the corresponding sampling time. The data were used to compute the velocity of the link in each experiment. Afterwards, the values of torque and velocity during the constant velocity stage of the trajectory were averaged and used to construct the friction velocity map, which is basically a torque versus velocity plot. Finally, the parameters of the friction model were estimated to fit the friction velocity map by applying a least-squares minimisation method (Johnson and Lorenz, 1992; Canudas de Wit et al., 1995; Canudas de Wit and Lischinsky, 1997; Olsson et al., 1998) using

$$\sum_{i=1}^{n} [F(v_i) - \hat{F}(v_i)]^2 \tag{4.2}$$

where $F(v_i)$ is the measured torque (i.e. the friction force) at certain constant velocity v_i, and $\hat{F}(v_i)$ is the value estimated by the friction model expressed in Equation 4.2.

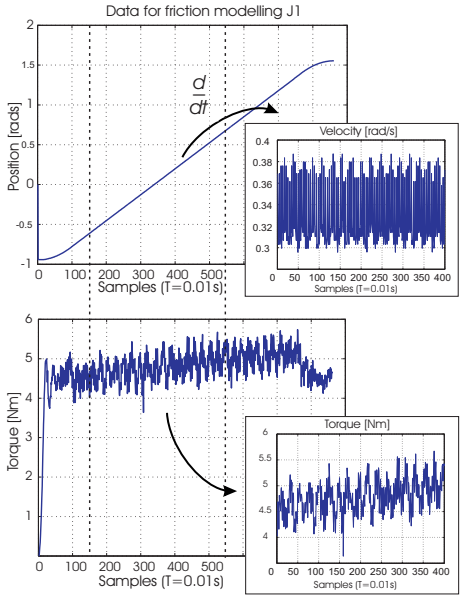


Figure 4.1: Register of position and torque in the Barret WAM when rotating joint 1 by 0.5 rad.

Figure 4.1 shows an example of the measured position and torque in the manipulator when rotating Joint 1. The data segment in the inset of the figure corresponds to the linear change in the position which is used to compute the joint velocity. The averages of the computed velocity v_i and measured torque $F(v_i)$, respectively, yield to ordered pairs $[F(v_i), v_i]$ that shape the friction velocity map.

As mentioned, the construction of the friction velocity map is based on the data gathered when the joint of the manipulator is rotated using both positive and negative velocities. Figure 4.2 and Figure 4.3 show the positive velocity profiles used in the friction identification of Joint 1, the respective joint torque was used together with the velocity value to determine the set of points that construct the positive stage of the friction velocity map. Similarly, Figure 4.4 and Figure 4.5 show the negative velocity profiles used in the friction identification of Joint 1 and the joint torques that construct the negative stage of the friction velocity map. The velocity experiments were conducted for every joint of the manipulator in order to model the friction phenomena in the WAM arm. Figure 4.6 shows the the velocity maps of all the joints of the WAM arm.

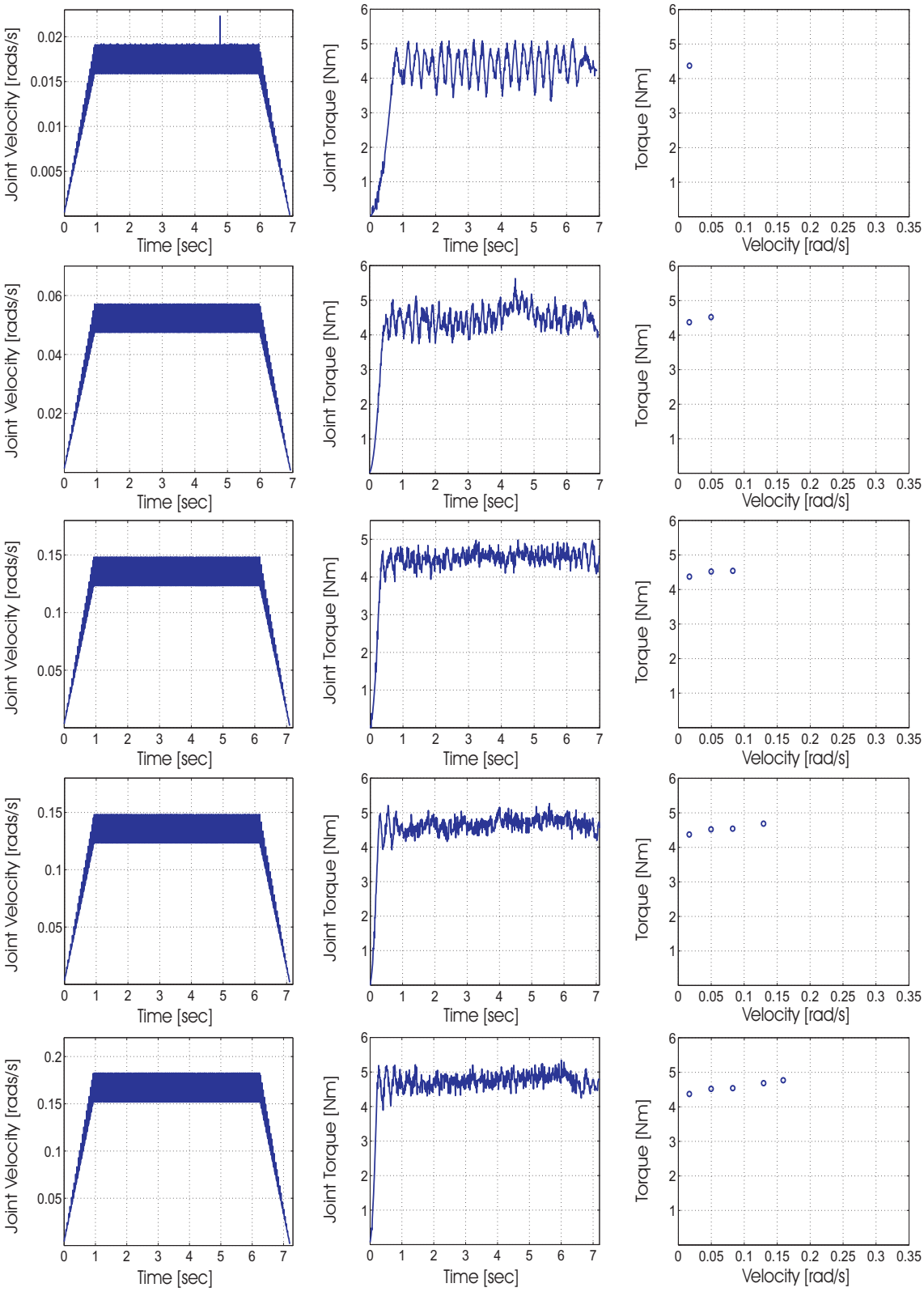


Figure 4.2: Positive velocity profiles used to create the friction velocity map of Joint 1.

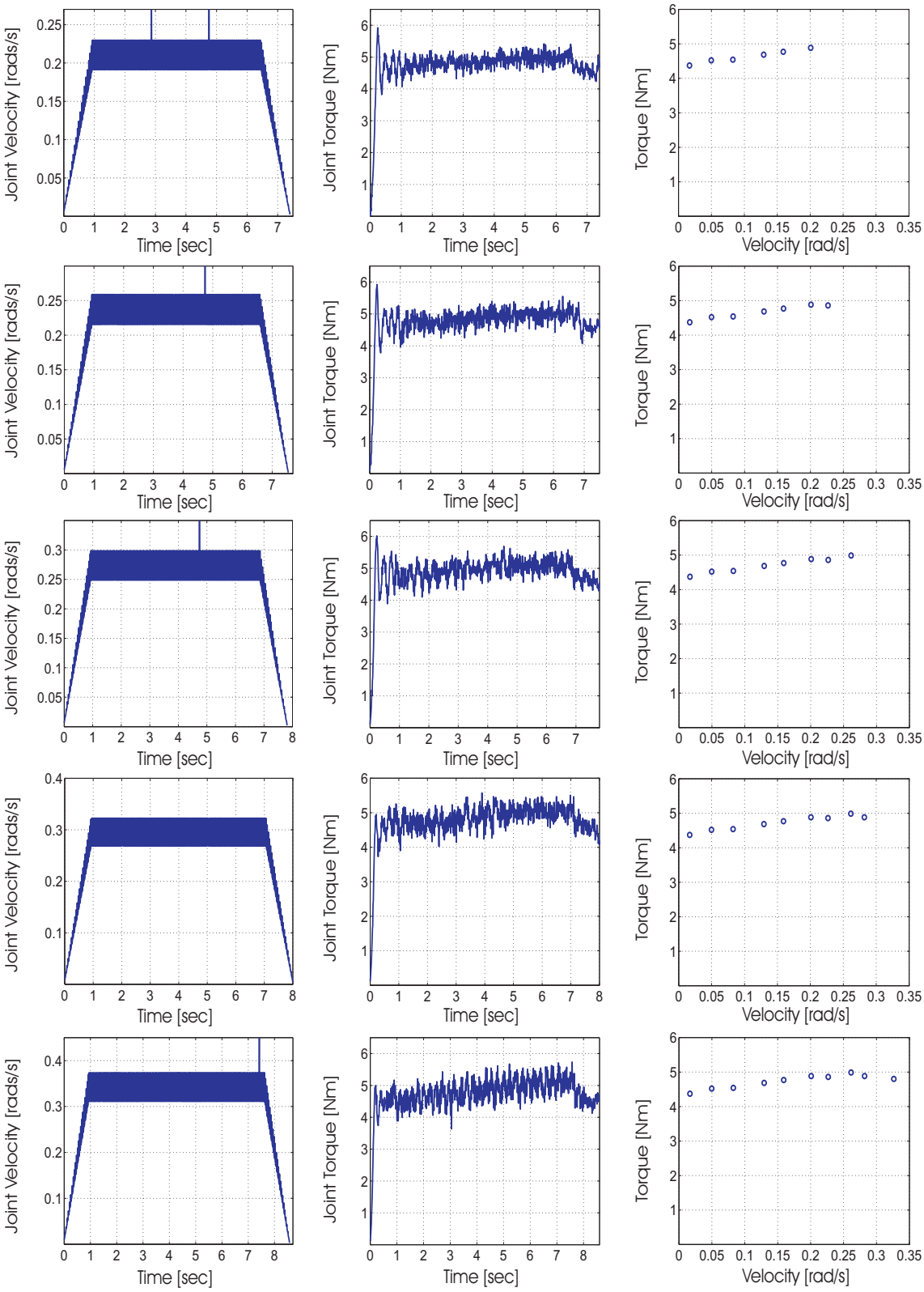


Figure 4.3: Positive velocity profiles used to create the friction velocity map of Joint 1.

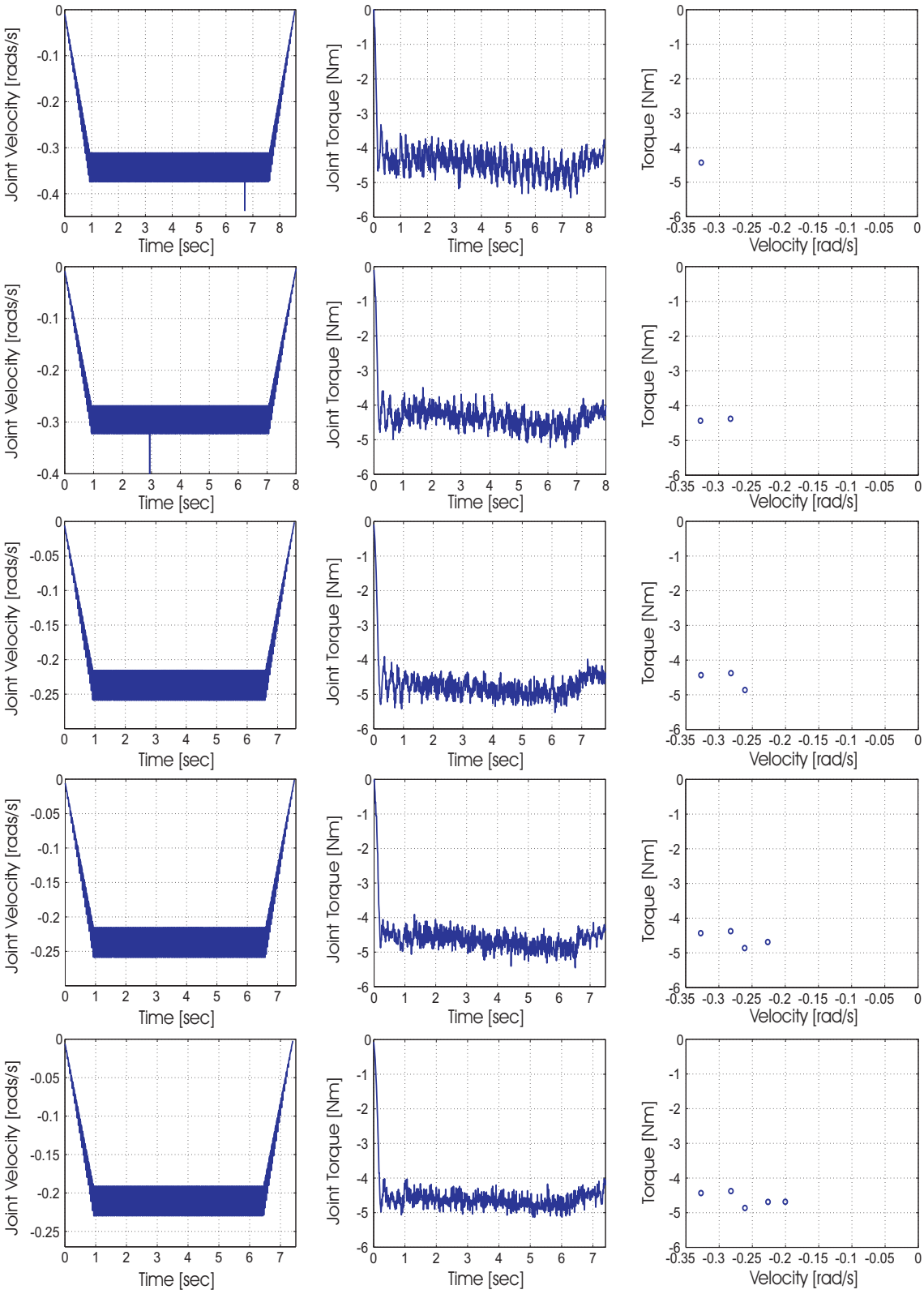


Figure 4.4: Negative velocity profiles used to create the friction velocity map of Joint 1.

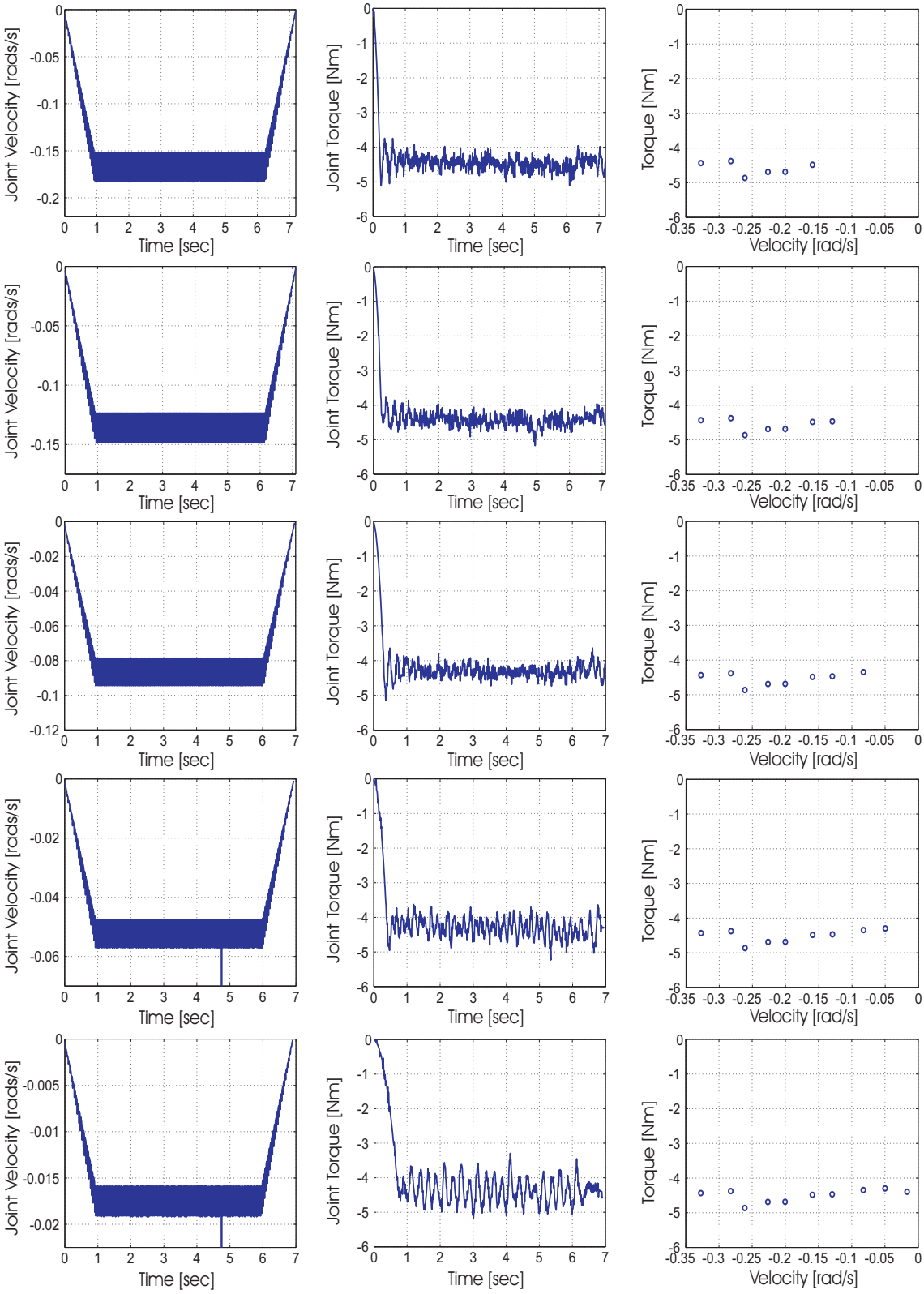


Figure 4.5: Negative velocity profiles used to create the friction velocity map of Joint 1.

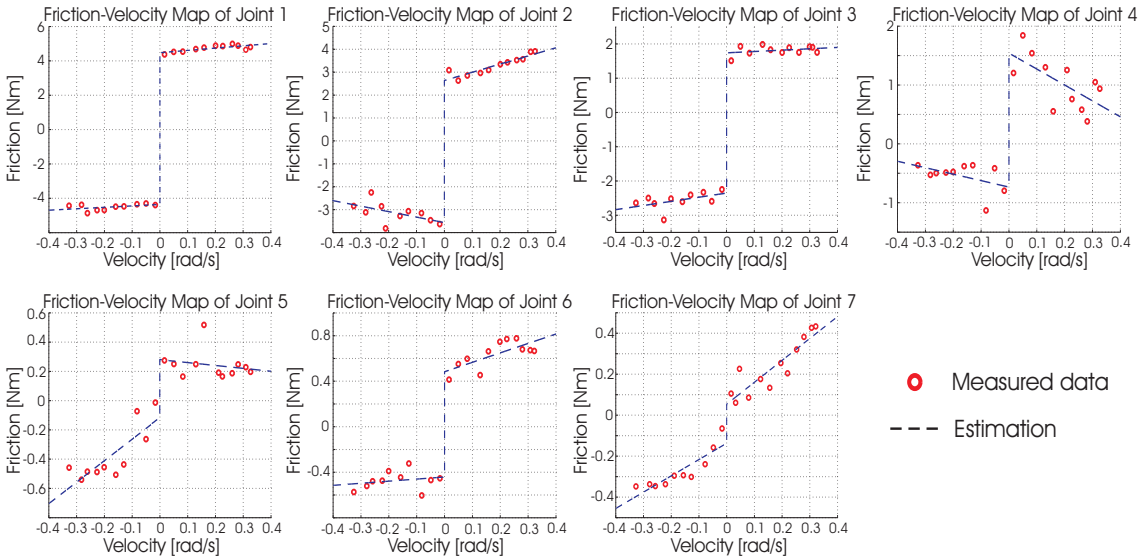


Figure 4.6: Friction velocity maps corresponding to each joint of the 7-DOF manipulator. The measured data was obtained after rotating each joint of the manipulator at different constant velocities whilst the estimated value was computed using a fitted classic friction model.

Table 4.1 summarises the parameters that shape the friction model for all the joints of the 7-DOF WAM Arm. The measured friction suited accurately the shape of the classic friction model.

In general, the estimation of friction is very accurate for all the joints, and it is suitable enough to be considered for future implementations of friction compensation techniques in the joints of the WAM arm. Therefore, the estimated friction could be used as part of the feed-forward compensation block.

Joint	Coulomb friction F_c $(V+/V-)$	Viscous friction σ_2 $(V+/V-)$
1	4.4748/ 4.3609	1.3348/ 0.8266
2	2.6385/ 3.5643	3.5572/-2.3951
3	1.7399/ 2.3529	0.3944/ 1.2075
4	1.5414/ 0.7342	-2.7045/-1.0969
5	0.2798/ 0.1172	-0.1972/ 1.4670
6	0.4834/ 0.4417	0.8291/ 0.1836
7	0.0538/ 0.1370	1.0689/ 0.7954

Table 4.1: Friction model parameters. $V+/V-$ are positive and negative velocities respectively.

4.3 Feed-forward friction compensation

Feed-forward compensation of systems consists in a control scheme working in parallel with the actual controller of the system. It is a desirable option for compensating relatively known disturbances due to the fact that it can be tuned individually without affecting the stability of the on system controller (Golnaraghi and Kuo, 2010; Belanger, 1995; Seborg et al., 2004).

For the present project we designed the feed-forward compensation as a control scheme worked in parallel with the joint PD controllers with gravity compensation originally configured in the WAM arm. The configuration that we used is shown in Figure 4.7 and consists in a feed-forward compensation block $C_f(s)$ added to the joint PD control configuration shown in Figure 2.3. This configuration was easy to implement in software and its response depended on the reference signal only. The function that defined the compensation block $C_f(s)$ is explained as follows. If each link of the manipulator is considered as a single mass in motion, the feed-forward scheme must ensure that the force needed to execute the desired rotation is generated (Su and Zheng, 2009). Then, the actuating force $\tau_{ff}(t)$ necessary to rotate the joint may be expressed as the sum of the mass m of the link multiplied by its reference acceleration $\ddot{q}_r(t)$, the viscous damping coefficient σ_2 multiplied by the reference velocity $\dot{q}_r(t)$ and the friction force as function of the velocity:

$$\tau_{ff}(t) = m\ddot{q}_r(t) + \sigma_2\dot{q}_r(t) + F_c(\dot{q}_r(t)) \tag{4.3}$$

where F_c is the Coulomb friction that affects the joint. Mathematically, the friction and gravity compensated joint PD control is defined as the sum of the contributions of the control actuating force $\tau_c(t)$ and the compensation torque $\tau_{ff}(t)$. The

actuating joint torque $\tau(t)$ is

$$\tau(t) = \tau_c(t) + \tau_{ff}(t) \tag{4.4}$$

After the implementation of the friction compensation module we tested the system in the execution of a motion task, and we compared the results with the performance of the system without friction compensation.

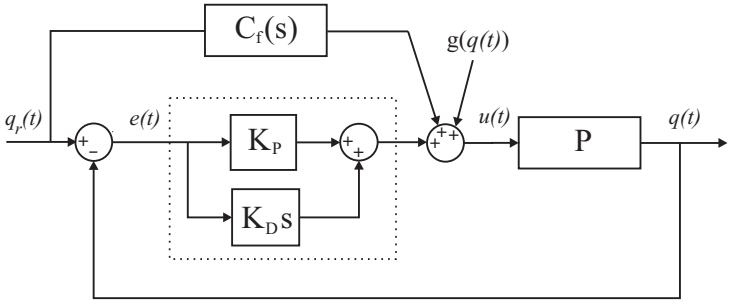


Figure 4.7: Diagram of the PD control with gravity compensation, augmented with a feed-forward compensation block expressed as $C_f(s)$.

4.4 Results

The aim of implementing a friction compensation module in the joints of the WAM arm was to analyse the influence of friction force in the accuracy of the joints when executing a motion task. The task consisted in a joint rotation of 0.5 radian at a velocity of 0.083 rad/s. This rotation was executed for all the joints of the manipulator one at a time. The motion task is the same that was used in Chapter 3 to analyse the performance of the WAM arm under PD control. We chose the same task in order to have a point of comparison among the two control techniques: a) PD control with gravity compensation and b) PD control with feed-forward friction compensation and gravity compensation.

The joint reference trajectory and the joint positions were recorded in all the

experiments in order to compute the joint position errors during the execution of the trajectory and a relative percentage error (RPE) for the final position of the joint and the actual joint trajectories of the manipulator, as described in Section 3.2.2.

The feed-forward friction compensation scheme (as described in Section 4.3) was implemented over the on-system joint PD control with gravity compensation.

Figure 4.8 shows the registers of the reference trajectories and the joint positions during the motion task. By observing the trajectories alone it is difficult to have a fair appreciation of the position errors of Joint 1, Joint 2, Joint 3, Joint 4 and Joint 5 since they are very small, contrary to Joint 6 and to Joint 7 were the position errors are more evident. If we compare these plots with those in Figure 3.3 we can notice how the trajectory becomes smoother after the compensation.

Figure 4.9 shows the comparison of the manipulator's response to the task considering the position errors. Each plot displays the position errors for every joint when executing its respective rotation. The steady state position errors in Joint 1 decreased when the friction compensation module was added to the PD control with gravity compensation. In Joint 2, the steady state position errors did not change significantly when the the friction compensation module was added to the PD control with gravity compensation. In Joint 3, the friction compensation increased slightly the steady state position error in comparison with the response of the system when using the PD controller and gravity compensation alone. The performance of the controllers implemented in Joint 4 was similar to the performance of the controllers implemented in Joint 2, the steady state position errors remain almost the same after adding the friction compensation module to the PD control with gravity compensation. In Joint 5, the steady state position errors decreased when the friction compensation module was added to the PD control with gravity compensation. In Joint 6, the friction compensation increased slightly the transient

state position errors in comparison with the response of the system when using the PD controller and gravity compensation alone. But the steady state position errors were smaller. Finally, in the case of Joint 7 the friction compensation module helped to reduce the position errors considerably in both the transient and the steady state, in comparison with the controller without compensation.

Finally, for a better comparison of the performance of the manipulator with and without the module for friction compensation, the computation of the relative percentage error of the final position of the joint was measured using Equation 3.2.

When using PD control without friction compensation the RPE in Joint 1 was 1.0876 %, in Joint 2 was 0.2342%, in Joint 3 was 0.8666%, in Joint 4 was 0.2912%, in Joint 5 was 1.3592%, in Joint 6 was 3.2848% and in Joint 7 the RPE was 5.4132%. The value of the RPE when the manipulator worked under PD control with feed-forward friction compensation was 1.1198% in Joint 1, in Joint 2 was 0.2244%, in Joint 3 was 0.8988%, in Joint 4 was 0.4398%, in Joint 5 was 1.2426%, in Joint 6 was 1.7974% and in Joint 7 the RPE was 7.2360%. Therefore, considering the final position error, the feed-forward friction compensation module improved the performance of the manipulator in Joint 2, Joint 5 and Joint 6. The final position error slightly increased in Joint 1, Joint 3, Joint 4 and Joint 7.

4.5 Conclusions

In this chapter we described the implementation of a friction compensation module proposed to improve the performance of the WAM arm. The aim of the implementation was to reduce the position errors occurring during the execution of joint rotations, to reduce the end position error and to smooth the joint trajectories. First, the friction phenomena manifested in each joint of the real manipulator was identified through the execution of several joint rotations at different velocities, in order to construct velocity maps. The velocity maps were used to find the parameters of classic friction models using a least-squares minimisation method.

Later, the mathematical representation of the friction was used to implement a feed-forward control scheme over the PD controllers with gravity compensation pre-configured in the WAM arm.

We also compared the performance of the WAM arm in a motion task under the two different control strategies: PD control with gravity compensation and PD control with feed-forward friction compensation and gravity compensation. We analysed the results obtained in the motion task and we found that the implementation of the friction compensation module in the joint controllers improved the performance of the manipulator during the execution of the task by reducing the position errors in six out of the seven joints of the WAM arm. In Joint 3 the compensation slightly increased the errors.

We found the relative percentage error of the final position of the joint using Equation 3.2. The reduction in the end position error was not important considering that the RPE was reduced in only three out of the seven joints of the manipulator. However, the friction compensation module did help to smooth the trajectories of the joints while executing rotations.

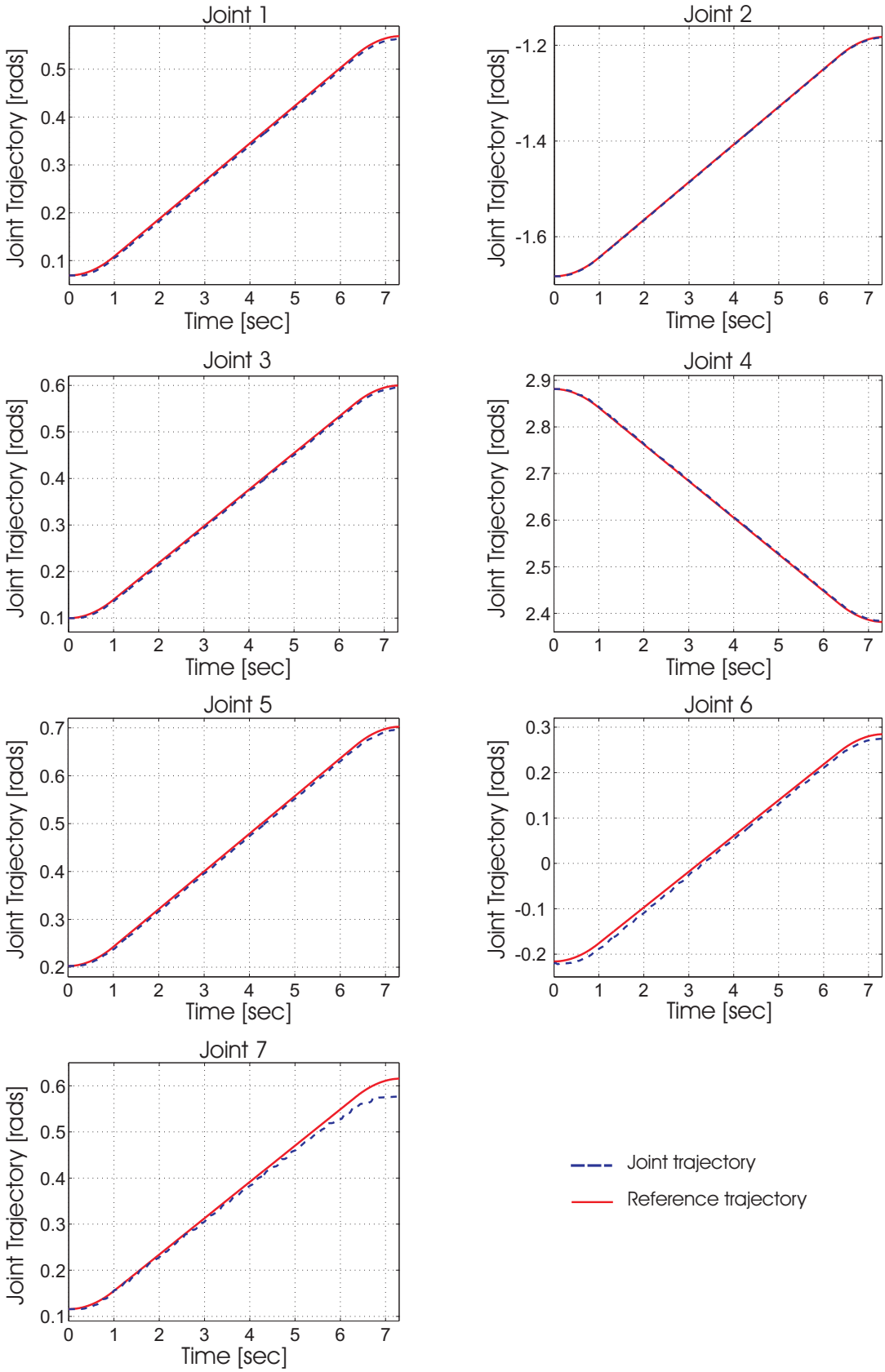


Figure 4.8: Joint trajectories of the 7 joints of the manipulator when executing a rotation of 0.5 rad. The rotations were executed by every joint one at a time. The system worked with joint PD control with gravity compensation augmented with a feed-forward friction compensation module.

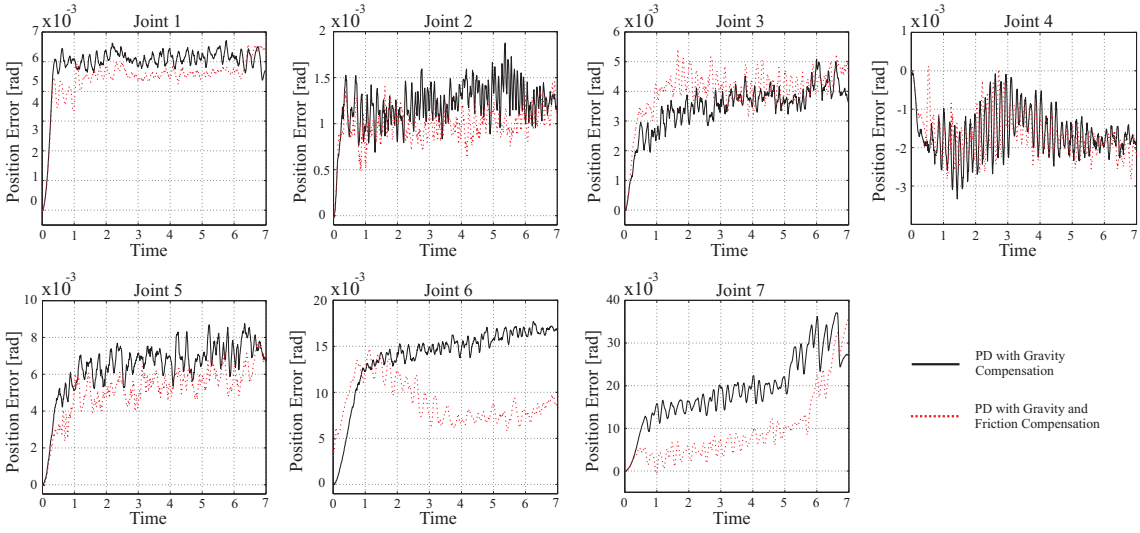


Figure 4.9: Joint position errors of the 7-DOF whole arm manipulator using different control strategies to execute rotations of the joints. The rotations were executed by every joint one at a time. The system was initially configured with joint PD control and gravity compensation and it was later augmented with a feed-forward friction compensation module.

Chapter 5

Iterative feedback tuning in the 7-DOF WAM arm

5.1 Introduction

In a controlled robotic system, gravity compensation with joint Proportional Integral and Derivative (PID) control eliminates the offset error in the response of the system caused for the influence of gravity (Kelly et al., 2005). However, the existence of other disturbances such as friction will not guarantee a zero error in the output response and the effects of friction must be cancelled separately through the implementation of a compensation strategy. In some cases the influence of friction does not affect significantly the performance of the system, so a compensation technique might not be required. PID control strategy is capable of dealing with friction when all its parameters are conveniently selected (Lewis et al., 2004). However, tuning the PID controller parameters might not be a straightforward process. A suitable method for PID controller tuning based on the use of experimental data is the Iterative Feedback Tuning (IFT) (Hamamoto et al., 2003; Hildebrand et al., 2005). The reliability of this method is based on the fact that the response of the system is recorded and used later to estimate a gradient direction for the value of the parameters through several iterations, according to the standard of a determ-

ined performance measure. IFT is a model-free tuning method, however, when the experiments to obtain the gradient are difficult to implement on the system, the use of a model to conduct this procedure seems suitable.

The theory behind IFT was developed for discrete-time linear time-invariant (LTI) systems but its implementation has been proven to be effective on systems with non linearities (Hjalmarsson, 1998; Gunnarsson et al., 2003; Radac et al., 2011; Sjorberg et al., 2003). Based on the later, in order to improve the performance of the 7-DOF WAM arm, we proposed to use the IFT technique to tune the parameters of the PID controllers. In our implementation of the method we used an accurate model to conduct part of the experiments since some of them were difficult to implement on the real system.

During the tuning process we used the response of the real system to estimate the performance measure, while the gradient direction for the parameters was computed using a mathematical model of the system.

As mentioned in Section 3.2.2 we were looking to reduce the end position error to less than 1% in all the joints of the manipulator and to obtain smoother joint trajectories during the execution of tasks. The integral component of the PID control allowed to reduce the steady-state error while the implementation the IFT technique based on the use of experimental data allowed to tune the controllers taking into account all the disturbances acting on the joints during the execution of the trajectories.

After the tuning process ended, we used the results to make a comparison of the performance of the manipulator under three different control strategies: PD control with gravity compensation, PD control with gravity compensation and feedforward friction compensation, and PID control.

5.2 IFT in the joint PID controllers of the WAM arm

As previously mentioned, the 7-DOF WAM arm manipulator was configured by the manufacturer with joint PD control and gravity compensation. However, the system can be configured to perform under joint PID control by properly adjusting the gains of the joint controllers. Despite of the system being non linear, it is possible to iteratively estimate the controller parameters by applying an IFT technique (Hjalmarsson, 2002).

In Section 2.2.3 we described that the IFT technique is based on the realisation of several experiments where the reference signal is changing. The collected data is used to construct a minimising criterion based on the error $\tilde{y}(\rho)$ of the output response.

$$\tilde{y}(\rho) = y(\rho) - y_r \tag{5.1}$$

The variable ρ is a (3×1) vector that contains the controller parameters $\rho = [K_P \quad K_D \quad K_I]^T$. The objective function is defined as a (3×1) vector $J(\rho)$

$$J(\rho) = \frac{1}{2N} \sum_{t=1}^{N} E[\tilde{y}(\rho)^2] \tag{5.1}$$

The objective function is used to calculate iteratively the values of ρ according to

$$\rho_{i+1} = \rho_i - \gamma_i R_i^{-1} \Big[\frac{\partial J(\rho_i)}{\partial \rho} \Big] \tag{5.1}$$

Here R_i^{-1} is a (3×3) positive definite matrix. The value of R_i^{-1} indicates the update direction of the parameter in the next iteration. The value of γ_i gives the iteration step and $\Big[\frac{\partial J(\rho_i)}{\partial \rho} \Big]$ is an estimate of the first derivative of the objective function, known as gradient. The gradient is a (3×1) vector.

The IFT tuning technique was originally developed to tune the parameters of PID

controllers without the need of having a mathematical representation of the system, using a set of experiments based on an closed loop configuration. The data obtained during the experiments were used to estimate the gradient of the objective function $J(\rho)$ and to compute iteratively the controller parameters. However, since some of the experiments were difficult to execute in the WAM arm, we used an accurate model of the system implemented in SimMechanics (described in Chapter 3) to perform part of the experiments. Then, we used data from the response of the real system to estimate the performance measure, while the gradient direction for the parameters was computed using the mathematical model of the robot arm.

The IFT algorithm for controllers with one degree of freedom proposed by Hjalmarsoon to tune the PID controller of every joint of the manipulator independently follows the next steps:

1. Perform a rotation in the real system according to the closed loop configuration described in Figure 2.3 and record the reference trajectory r and the joint positions $y(\rho)$. Compute $\tilde{y}(\rho) = r - y(\rho)$ and define a variable $\tilde{y}^1(\rho) = \tilde{y}(\rho)$. The value of $\tilde{y}^1(\rho)$ represents the joint position error obtained during the first experiment.

2. Perform a second experiment considering the closed loop configuration described in Figure 2.3 applying $\tilde{y}^1(\rho)$ as reference signal and define a variable $y^2(\rho) = y(\rho)$. $y^2(\rho)$ contains the values of the joint positions recorded during the second experiment.

3. Compute $\frac{\partial C(\rho)}{\partial \rho}$ for all the elements in ρ and filter $y^2(\rho)$ in order to obtain $\frac{\partial \tilde{y}(\rho)}{\partial \rho}$ as shown in the diagram of Figure 2.4.

4. Estimate the gradient of the objective function with Equation 2.15.

5. Calculate the Hessian of the objective function using the value of the gradient and update the controller parameters for the next iteration.

As mentioned previously, we implemented the IFT method using both real and simulated data. The first experiment consisted in rotating a joint of the real manipulator while recording the reference joint trajectory and the actual joint positions. The second experiment was made on the model of the manipulator using the data collected during the first experiment while recording the joint positions. The steps followed to implement the IFT technique on the controllers of the WAM arm were:

- Perform step 1 of the IFT algorithm for controllers with 1-DOF in the real manipulator and use the reference joint trajectory and the measured joint positions $y(\rho)$ to estimate the position error $\tilde{y}^1(\rho)$.
- Use $\tilde{y}^1(\rho)$ to perform step 2 of the IFT algorithm for controllers with 1-DOF in the model, and use the values of the joint positions to define $y^2(\rho)$.
- Perform step 3, 4 and 5 of the IFT algorithm for controllers with 1-DOF.

Since the WAM arm is a non linear system we considered to use a Broyden-Fletcher-Goldfarb-Shanno (BFGS) as an alternative algorithm to compute the Hessian as proposed by Hamamoto and colleagues in (Hamamoto et al., 2003). BFGS algorithm is an iterative method for solving non linear optimisation problems. The Hessian is approximated using updates specified by gradient (or an approximation) evaluations. We used two methods to approximate the Hessian, a method based on the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm and the Newton-Gauss method. We implemented both methods separately in order to observe the performance of each method in simulations. Both methods were useful to approximate the Hessian, however we noticed that the BFGS method gave more accurate results in the earlier iterations while the Newton-Gauss method was more accurate near the minimum. Hence, for the implementation of the IFT algorithm, the computation of the Hessian was made using a combination of both methods. The BFGS method was used first to reduce the objective function using a fix step size of $\gamma = 0.5$. Once the objective function was near to the minimum, the Newton-Gauss method was implemented

Joint	Objective function $J(\rho)$					
	Initial	BFGS	Newton-Gauss			
	PID	*iteration 1*	*iteration 2*	*iteration 3*	*iteration 4*	*iteration 5*
1	0.0715	0.0321	0.0296			
2	0.5837		0.6142	0.5933		
3	0.2343	0.0105	0.0132			
4	1.5818	0.6169	0.5765	0.5235	0.4753	0.4585
5	0.3137	0.1705				
6	0.3511	0.1352				
7	2.0517	0.9095	0.8613			

Table 5.1: Objective function $J(\rho)$ for the seven joint controllers of the manipulator at different iterations.

with step sizes of $\gamma = 0.5$ and $\gamma = 0.1$. The values of gamma were varied according to the response of the system during simulations.

The value of the objective function $J(\rho)$ in Equation (2.13) was estimated using the response of the real manipulator, allowing a fair perspective of the performance of the controller in each iteration. During the experiments, every joint of the manipulator was rotated 0.5 radian at a velocity of 0.083 rad/s, one at a time. If we consider that the system uses a trapezoidal velocity profile, at velocities from 0 to 0.083 rad/s the value of the friction force affecting each joint of the manipulator does not change largely and it remains approximately the value of F_c for all the trajectory. Refer to Figure 4.6 to observe the friction velocity maps of all the joints of the 7-DOF WAM arm. Each rotation lasted approximately 7.3 seconds. The sampling period of the data measured in the real system was of 0.01 seconds. The objective function was initially estimated using PID parameters proposed intuitively. This is, for the proportional and derivative gain we used the values of the PD controller parameters pre-configured in the joints of the WAM arm, and for the integral gain we used the value of the derivative gain divided by 10.

5.3 Results

Table 5.1 summarises the IFT process according to the value of the objective function obtained for each joint controller of the manipulator at each iteration. The value of the objective function was considered using 4 digits. In all cases the gradient of the objective function was first computed using the BFGS method, once the value of the objective function stopped decreasing, or it started increasing, the method to compute the gradient changed from the BFGS method to the Newton-Gauss method. We stopped the iterations when the objective function stopped decreasing (considering a 4 digit precision) or it started increasing. In Joint 1 only two iterations were made, one using the BFGS method and one using the Newton-Gauss method. In Joint 2, we made two iterations using the Newton-Gauss method, since the value of the objective function were negative when using the BFGS method; however, according to the value of the objective function, the first choice of parameters had the best performance. In Joint 3 we made one iteration using the BFGS method and one iteration using the newton-Gauss method, but according to the value of the objective function, only the first iteration was needed. In Joint 4 we made five iterations, one iteration using BFGS method and 4 iterations using the Newton-Gauss method. In the case of Joints 5 and 6, only one iteration was needed using the BFGS method. Joint 7 used two iterations, one obtained with the BFGS method and one obtained with the Newton-Gauss method. A total of 15 iterations were needed in order to tune the controllers of the whole system. Considering that two experiments had to be executed in each iteration, one in the real system and one through simulation, a total of 30 experiments were performed in order to tune the parameters of the seven joint controllers of the WAM arm.

For a visual reference of the tuning process, refer to Figure 5.1. The graphs show the position error of the seven joints of the manipulator at different iterations, during the first 3 seconds of the rotation used in the experiments. In Joint 1, the posi-

tion error obtained when using the parameters of the controller chosen intuitively showed an oscillation that disappear after the first iteration, making the response of the controller more stable. The position error was also reduced to values close to zero after the second iteration. In the case of Joint 2, the values chosen intuitively for the controller parameters were the best fit according to the value of the objective function. The later is hard to notice by eye in the plots of the position error obtained between iterations. In Joint 3, the plots of the position error show clearly how the performance of the system improves when using the parameters obtained in the first iteration, but deteriorates at the second iteration. Five iterations were made in Joint 4 but, since the reduction of the objective function occurred very slow after the second iteration, the plots only include the second and the last iteration. These two plots show clearly the reduction in the position error after using the iterative method. In Joint 5 the improvement of the controller performance is not that clear. However, the value of the objective function $J(\rho)$ in Table 5.1 confirms that the position error was reduced after the first iteration. In Joint 6 only one iteration was needed to tune the parameters of the controller. Finally, in the plots of the position error of Joint 7, show the improvement in the performance of the joint controllers through the tuning process with the value of the error decreasing iteration by iteration.

As seen in Figure 5.1, the improvement of the system performance during the tuning process is quite evident through the joint position error plots in most of the cases. However, in the particular case of the controllers of Joints 2 and 5, a better perspective of the improvement of the controller may be obtained by referring to the values of the objective function $J(\rho)$ summarised in Table 5.1.

5.4 Discussion

After successfully tuning the parameters of the PID controllers we decided to compare the performance of the manipulator in the execution of a motion task,

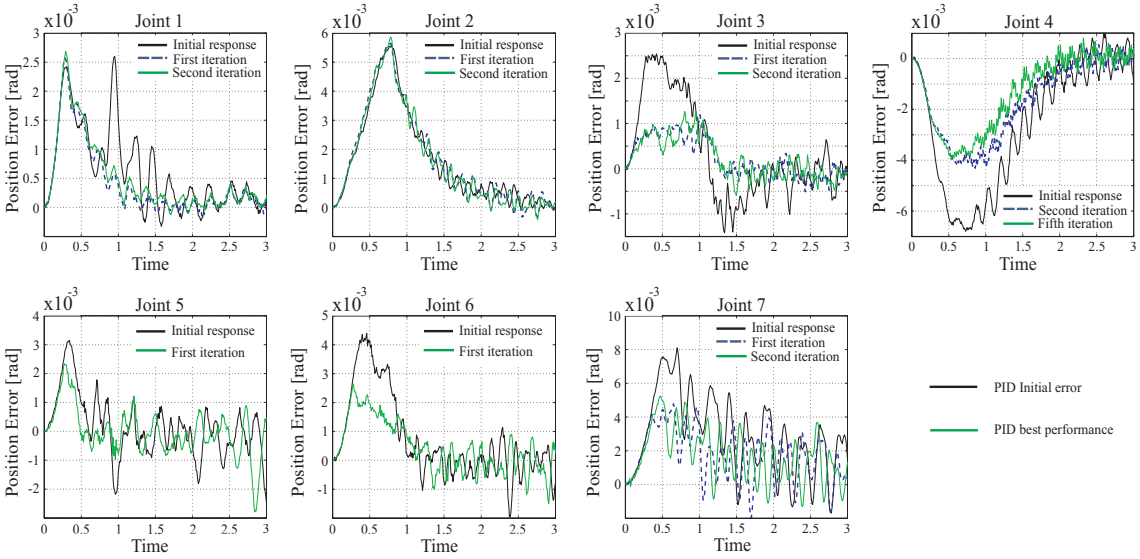


Figure 5.1: Joint position error of the 7-DOF whole arm manipulator with joint PID control. The response error is shown for different iterations of the controllers parameters estimated using IFT.

under three control strategies a) PD control with gravity compensation, b) PD control with feedforward friction compensation and gravity compensation, c) PID control. The WAM arm was configured originally under joint PD control and gravity compensation. The implementation of the joint PD control and gravity compensation and feedforward friction compensation technique is fully explained in Chapter 4.

The task consisted in a joint rotation of 0.5 rad at a velocity of 0.083 rad/s. This rotation was executed for all the joints of the manipulator one at a time. The motion task is the same that was used to analyse the performance of the WAM arm under PD control in Chapter 2. We chose the same task in order to have a point of comparison among the three control techniques. The joint reference trajectory and the joint position of the manipulator were recorded in order to find the joint position errors generated when using each control strategy. The joint reference trajectory and the joint positions were recorded in all the experiments in order to compute the joint position errors during the execution of the trajectory and a relative percentage error (RPE) for the final position of the joint and the actual joint trajectories of the manipulator, as described in Section 3.2.2.

The performance of the PID joint controllers is shown in Figure 5.2. The plots are the registers of the reference trajectories and the joint positions during the motion task described earlier. We can notice that the joint trajectory is very close to the reference trajectory in all the joints of the manipulator, meaning that the joint position errors are very small. If we compare these plots with those in Figure 3.3 and Figure 4.8 we can notice that all the joint trajectories obtained when using PID control are smoother and more accurate with respect to the reference trajectory than those obtained when using PD control with gravity compensation and PD control with feedforward friction compensation and gravity compensation.

Figure 5.3 shows the comparison of the manipulator's response to the motion task considering the plots of the joint position errors. The steady state position errors in Joint 1 decreased when the friction compensation module was added to the PD control with gravity compensation; however, the lowest steady state position errors were obtained with the PID controller. The transient state position error was also smaller with the PID control. In Joint 2, the steady state position errors did not change significantly when the the friction compensation module was added to the PD control with gravity compensation. The lowest steady state position errors were obtained with the PID controller, although the transient state position errors were larger that those obtained when using the other two control strategies. In Joint 3, the friction compensation increased slightly the steady state position error in comparison with the response of the system when using the PD controller and gravity compensation alone. Opposite to the PID controller, in which case even the transient state position errors were smaller. The performance of the controllers implemented in Joint 4 was similar to the performance of the controllers implemented in Joint 2, the steady state position errors remain almost the same after adding the friction compensation module to the PD control with gravity compensation. The lowest steady state position errors were registered with the PID controller, but the transient state position errors were larger. In Joint 5, the

steady state position errors decreased when the friction compensation module was added to the PD control with gravity compensation, but decreased the most with the PID controller. The transient state position errors were also smaller with the PID control. Although small oscillations were present during the linear stage of the joint trajectory, these oscillations did not compromise the stability of the system. In Joint 6, the friction compensation increased slightly the transient state position errors in comparison with the response of the system when using the PD controller and gravity compensation alone. But the steady state position errors were smaller. The position errors in both the transient and the steady state were smaller when using the PID controller among the three control strategies implemented in the joint. Finally, in the case of Joint 7 the friction compensation module helped to reduce the position errors considerably in both the transient and the steady state, in comparison with the controller without compensation. The transient state errors obtained with the PID controller were the same as those obtained when the friction compensation module was added to the PD controller with gravity compensation. However, the steady state position errors obtained with the PID controller were smaller.

The relative percentage error (RPE) of the final position of the joint was measured using Equation 3.2. When using PID control the value of the RPE in Joint 1 was 0.0003%, in Joint 2 was 0.0%, in Joint 3 was 0.0004%, in Joint 4 was 0.0003%, in Joint 5 was 0.0003%, in Joint 6 was 0.0003% and in Joint 7 was 0.0007%. When using PD control with feed-forward friction compensation the value of the RPE in Joint 1 was 1.1198%, in Joint 2 was 0.2244%, in Joint 3 was 0.8988%, in Joint 4 was 0.4398%, in Joint 5 was 1.2426%, in Joint 6 was 1.7974%, in Joint 7 was 7.2360%. In comparison with the performance of the manipulator under PD control without friction compensation the RPE in Joint 1 was 1.0876 %, in Joint 2 was 0.2342%, in Joint 3 was 0.8666%, in Joint 4 was 0.2912%, in Joint 5 was 1.3592%, in Joint 6

was 3.2848%, in Joint 7 was 5.4132%. Considering the final position error, the use of joint PID control improved the performance of the manipulator in every joint of the manipulator when executing a motion task.

5.5 Conclusions

In this chapter we described the implementation of an Iterative Feedback Tuning technique to tune the parameters of the 7-DOF WAM arm joint PID controllers. The Iterative Feedback Tuning technique used experimental data to estimate the gradient direction for each parameter, as described in (Hjalmarsson, 2002). However, since some experiments were difficult to execute in the real system, we used a model of the system to implement part of the iterative technique. Then, we used both experimental data and model simulations in the tuning process and we considered two different approaches for the approximation of the gradient: a Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm and the Newton-Gauss method. The BFGS based method was used to approach the minimum of the performance measurement using a fixed step size, while the Newton-Gauss method was used to get closer to the minimum using two different step sizes. The estimates of the parameters obtained through the tuning process were used to configure the joints of the WAM arm to work under PID controllers, as shown in Section 5.3. The implementation of the PID controllers allowed to perform motion tasks in the manipulator without the need to activate the gravity compensation module pre-configured in the WAM arm. We also compared the performance of the WAM arm in a motion task under the three different control strategies: PD control with gravity compensation, PD control with gravity compensation and feed-forward friction compensation, and PID control. According to the results obtained in the motion task, and using the performance of the WAM arm under the pre-configured PD control with gravity compensation as the standard for comparison, the use of joint PID controllers improved the most the performance of the system. This is, the addition of the feed-forward scheme used

for friction compensation to the PD control with gravity compensation, did help to reduce the position errors in all the joints of the manipulator, except of Joint 3, where the compensation slightly increased the errors. However, the reduction in the position error was not important in most of the joints that showed improvement, except for Joint 6 where the the value of the position error was reduced to almost half. The results obtained when using joint PID controllers without gravity compensation showed how the steady state position errors in all the joints of the manipulator dropped to values very close to zero, reducing the end position error to less that 0.007%. Also, the joint trajectories obtained with PID control were smoother than the ones obtained with PD control and gravity compensation and with PD control with friction compensation and gravity compensation. The response of the WAM arm under PID control during the motion task, also helped to support the results obtained during the iterative process that we used to tune the parameters of the joint controllers.

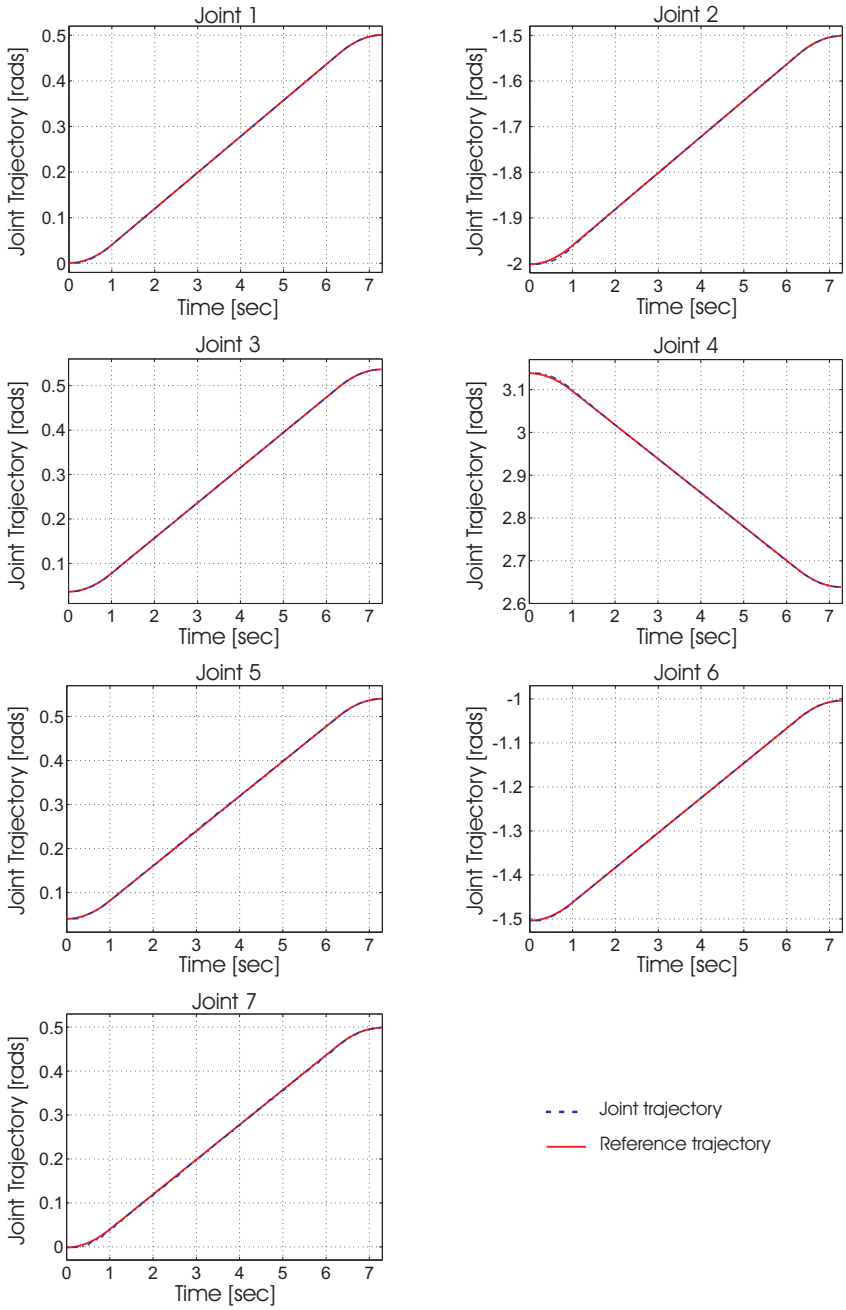


Figure 5.2: Joint trajectories of the 7 joints of the manipulator using joint PID control. Each joint is executing a rotation of 0.5 rad at a velocity of 0.083 rad/s, one joint at a time.

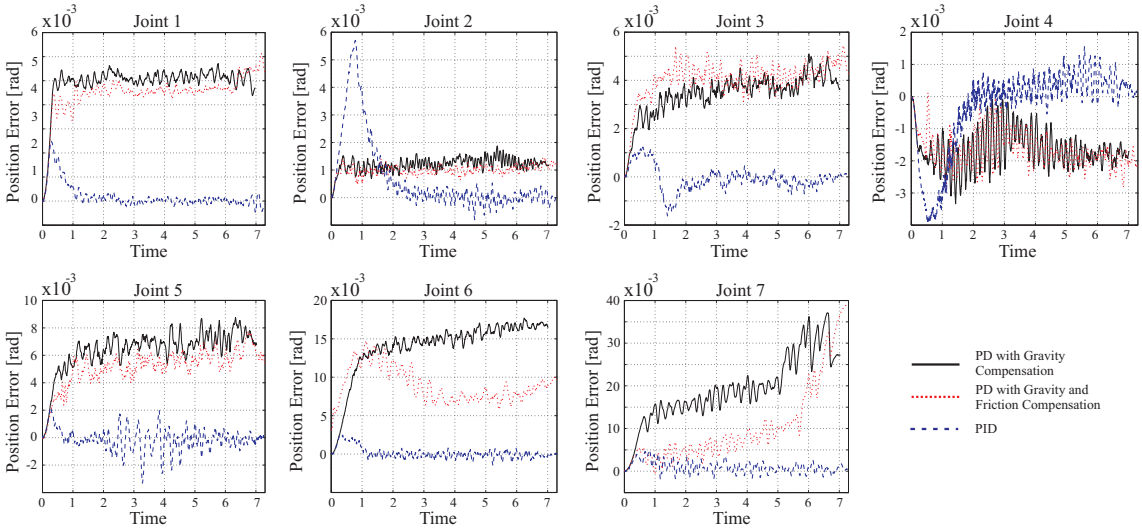


Figure 5.3: Joint position error of the 7-DOF whole arm manipulator using different control strategies to execute rotations of the joints: joint PD control with gravity compensation, joint PD control with gravity compensation and feed-forward friction compensation, joint PID control

Chapter 6

Human Machine Interface based on Gaze Tracking used to control a robot manipulator

6.1 Introduction

In this chapter we describe a Human Machine Interface (HMI) that allows the user to control the direction of a robot manipulator in order to grasp objects located on a table. The user wears a head mounted gaze tracking device that sends data to a computer that measures the gaze point with respect to a scene.

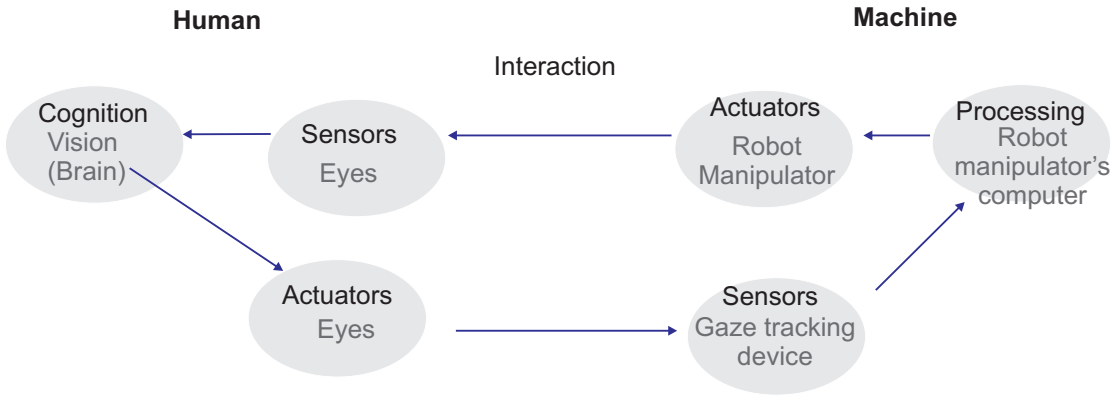


Figure 6.1: Structure of the proposed human machine interface used to control the position of a robot manipulator through gaze tracking.

The gaze point is then sent to another computer that holds the controls of the robot manipulator where it is processed and used to move the manipulator in real time. The elements that conform the HMI are described in the diagram of Figure 6.1. The cognition process takes place in the brain through the visual system and the eyes are both the *human*'s sensor and actuator. Among the elements of the *machine* the gaze tracker is the sensor, the robot manipulator is the actuator and the robot manipulator's computer serves as the *machine* processor.

The *machine* is a device composed by two main elements: the head mounted gaze tracker and the robot manipulator. Both devices interact as shown in Figure 6.2, the head mounted device measures the position of the eyes and sends this data to a laptop where the gaze point is computed. This value is sent through the serial port to the robot manipulator's computer where the data is processed and converted to control signals that are used to perform two main actions: give directions of movement or give a grasping instruction.
The user is able to control the position of the robot manipulator in a bidimensional space (workspace) considering its visual space, as given by the eye tracker's frontal camera. The subject must place its head in a position where the workspace is fully covered by the visual space and has to move the eyes in order to give directions to the manipulator while keeping a mental plot of the position of the object. The system allows the user to glimpse in order to find the actual position of the manipulator and the object during the execution of reaching and grasping tasks.

6.1.1 The Robot Manipulator

The robot manipulator is a 7 degrees of freedom (DOF) WAM arm from Barrett Technologies (refer to Section 3.2 for a more complete description of the system), with a three finger grasper BH-280 barrett hand. The manipulator is allowed to move in a two dimensional space (XY plane) limited by the kinematics of the arm

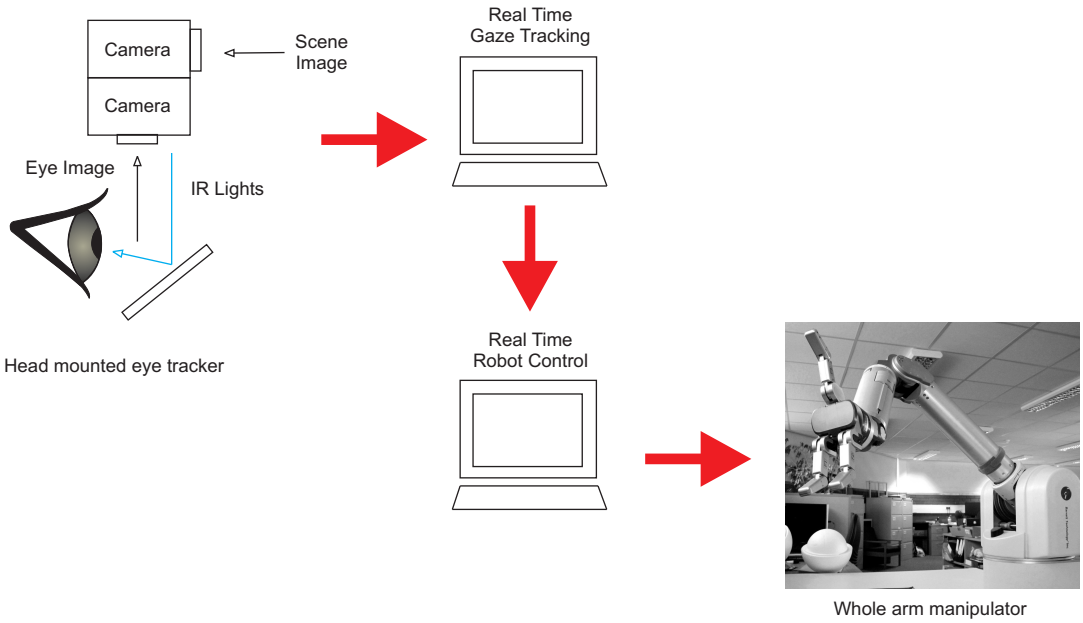


Figure 6.2: Diagram of the HMI based on gaze tracking. The elements of the system are a head mounted with laptop gaze tracking device, a robot manipulator and the robot manipulator's computer.

with the palm of the hand facing the XY plane. The orientation of the hand allows grasping objects from above. An image of the arm workspace with the arm in its initial working pose is shown in Figure 6.3. When the system is moving the hand preserves the palm-facing-table orientation with moderate directional rotations.

6.1.2 The Eye Tracking System

The eye tracking system is a head mounted device from Applied Science Laboratories (ASL). The system, the Mobile Eye, uses corneal reflections and pupil detection to compute the gaze by superimposing images from two cameras that record the eye and the scene simultaneously.

Three infra red lights are projected onto the eye and their reflections are used as reference point to find the angle and distance of the pupil while rotating the eye. The angles are synchronised with the images of the scene camera in order to

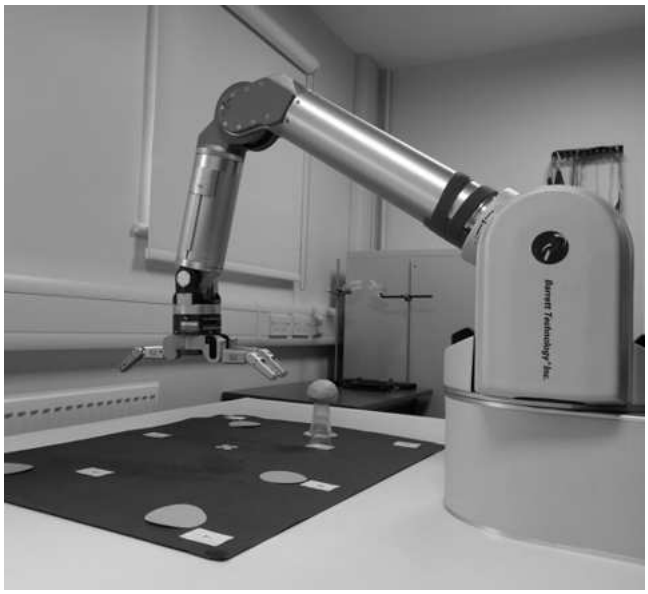


Figure 6.3: Initial pose of the robot arm; its XY working space is limited by the kinematics of the manipulator.

compute the gaze point in real time (Applied Science Laboratories, 2008).

The mobile eye system has the option to send the gaze point as a ordered pair through the serial port at a rate of 30 Hz. The coordinates are given in pixels referenced to the images of the scene which have a size of 768×576 pixels. When the information regarding to the corneal reflections or the pupil is not reachable, eye-closing for example, the ordinate pair (-2000,-2000) is given. In Figure 6.4 a plot of 50 seconds of raw data as it is sent to the serial port is shown. The figure displays separately the horizontal and vertical position of the gaze, the occurrence of saccades, blinking and corrupted data all along the recording are noticeable in this plots. For example, by looking into the first 12 seconds of data: blinking is present in seconds 4 and 10, there is an horizontal long saccade at second 9 and corrupted data can be seen right before the 10th second. Blinking and corrupted data may be easily filtered from the data segment by removing the ordinate pairs (-2000,-2000) and any value that is out of the space limits (768,576).

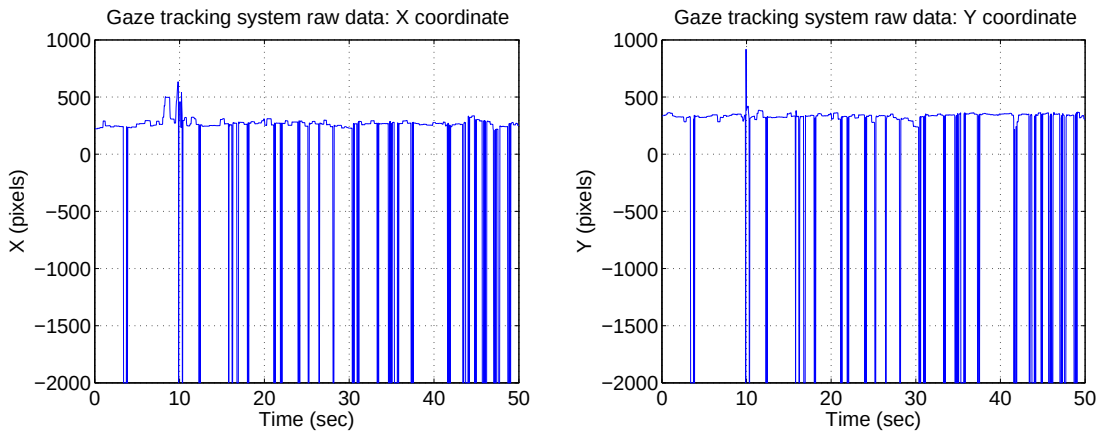


Figure 6.4: Gaze tracking raw data as it is being sent through the serial port of the mobile eye. The negative value -2000 represents blinking.

6.2 Interfacing the Gaze Tracking system with the WAM robot arm

The communication between the gaze tracking device and the robot manipulator is made through the serial ports of the gaze tracker laptop and the manipulator's computer. The diagram in Figure 6.5 represents the main data processing steps all the way since the acquisition of the gaze point, to the final execution of the user's instructions through the manipulator. Data processing, such as filtering and blinking detection, and the control strategy were programmed in Python. Python is a programming language composed by standard libraries and special packages used in several applications, such as scientific and numeric computing, software development, desktop GUI's, Networks programming, Internet applications and game developing (Python, 2013). This language can integrate with other objects and runs on Windows and Linux, which makes it perfect to interact with the WAM arm due to the fact that its development platform is written in C++ language and that it runs under a Linux based operating system.

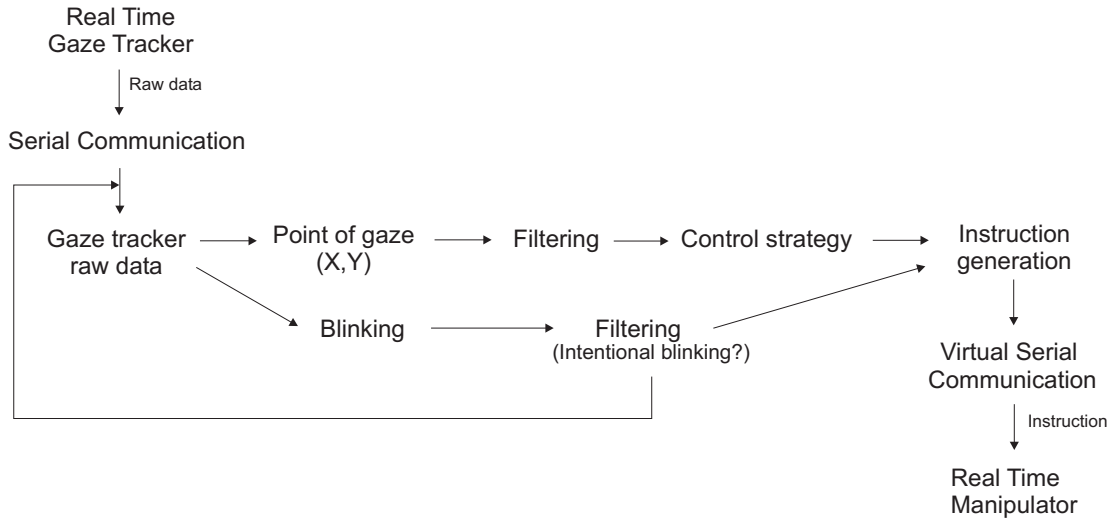


Figure 6.5: Main flow diagram showing the data process from the acquisition of the gaze point to the final execution of the user's instructions to control the manipulator.

6.2.1 Control signals

The HMI based on gaze tracking uses the gaze point and eye blinking as control signals. The gaze point determines the robot manipulator's direction of movement while intentional blinking indicates the execution of a grasping action. As explained in a previous section, the raw data vector from the gaze tracker contains continuous information of the measured gaze point, represented as an ordinate pair, altogether with blinking information seen as a (-2000,-2000) value. Then, the vector containing the raw data needs to be filtered first in order to separate the two control signals (Figure 6.6).

Only voluntary blinks are used as control signal. A blink is considered voluntary if its length is more than 0.7 seconds (Krolak and Strumillo, 2012). Since the data rate of the gaze tracking system is an approximating of 1 sample every 0.033 seconds, a voluntary blink will consist in a count of 22 samples of continuous blinking data (-2000,-2000).

After removing the blinking data from the raw data vector a continuous vector containing only the gaze point is generated.

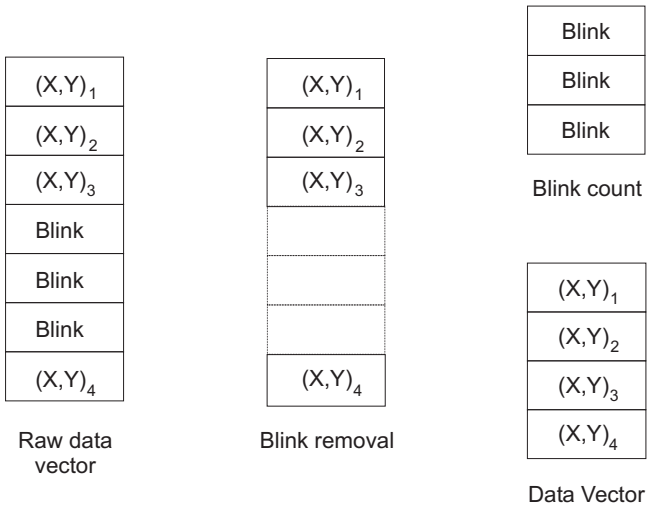
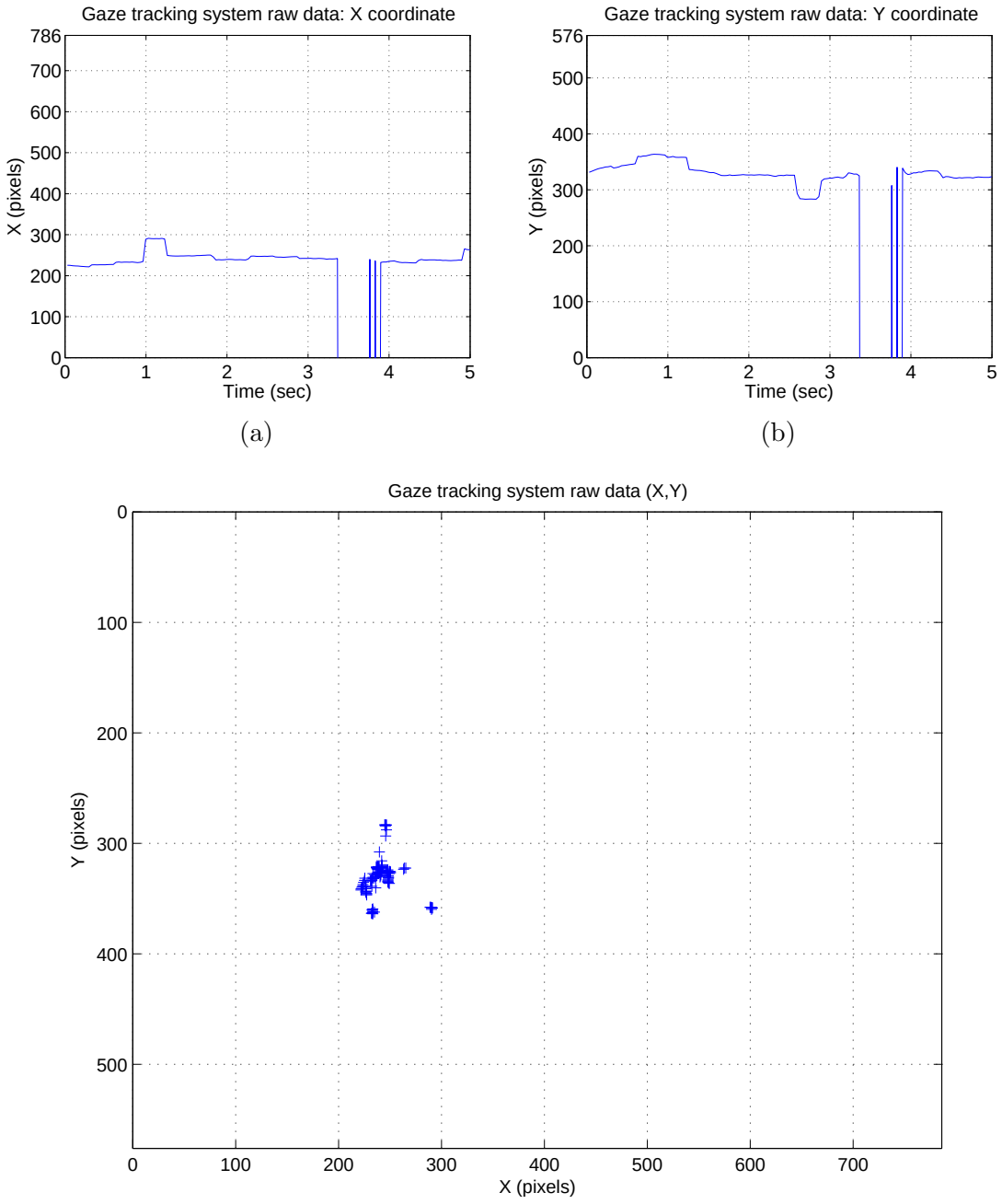


Figure 6.6: Gaze tracking raw data presentation as it is being sent through the serial port of the mobile eye. The blinking action is represented in the data vector with a numerical value of (-2000,-2000), while the gaze point is referred as a ordinate pair given in pixels according to the scene image.

Figure 6.7 shows 5 seconds of data containing the gaze point of a subject fixating in a location referenced to the scene image. The graphs in a) and b) show the individual coordinates X and Y versus time, the irregular shape of the signals indicates the presence of the involuntary saccadic movement of the eye that appear during the fixation process. Between second 3 and 4 there is large discontinuity in the signals, this discontinuity corresponds to the ordinate pair (-2000,-2000) that indicates blinking, or intentional eyes closing. For this particular set of data, the discontinuity represents blinking, but it is possible to identify if the eyes closure was involuntary or intentional in any set of data by measuring the duration of the discontinuity. The data showed in c) is the Cartesian plot of the gaze point after removing blinking. This type of graph is more suitable to visualise the fixation point with respect to the scene image and to appreciate the influence of the involuntary saccadic movement during the fixation. If we look at the X and Y plots, large involuntary saccades are present between 0.5-1.3 seconds and 2.5-3.0 seconds. Due to the saccadic noise, the data obtained from the gaze tracking device may need further filtering. In the next section several filtering techniques are proposed in order to smooth the signal.



(a) (b) (c)

Figure 6.7: Gaze tracking raw data presentation as it is being sent through the serial port of the mobile eye.

6.2.2 Eye control strategies

Two main strategies are proposed to control the direction of movement of the robot manipulator using the gaze point. The first one is a directional control that instructs the manipulator to follow five commands considering the position of the gaze in the XY plane: stop, right, left, up, down; and one instruction to control the grasp. The grasp control is used to open and to close the grasp and it is based on the user's intentional blinking. The conditions and methods used to determine the control commands are explained in Section 6.3. The general flow diagram of the four directional strategy is described in Figure 6.8. The arm is fixed in an initial position with the hand open. Once the control receives an instruction the system responds by moving the arm or by closing or opening the hand. The instruction for grasping works for both opening and closing the hand, so the system has to determine first the present state of the hand in order to execute the grasping order accordingly: if the hand was open and the control received the grasping order, the hand will close, but the hand will open at the occurrence of a grasping order if it had being closed previously. The four directional control allows the user to control the velocity of the manipulator through the implementation of a velocity profile that is also based on the position of the gaze point in the XY plane. The characteristics of the velocity profile will be fully explained in Section 6.3.2.

The second control strategy is a vectorial control that considers the gaze point as a vector referenced to the origin of the XY plane and which angle of direction will be used to guide the manipulator. The angle of direction goes from 0 to 360 degrees allowing diagonal motion, in addition to the vertical and horizontal motion offered by the four directional strategy. The vectorial control considers the magnitude and the direction of the gaze point vector to instruct the robot to stop or to give the direction that it has to follow; and uses intentional blinking as a condition to grasp, or release an object through the grasping command. The functioning of the grasping action

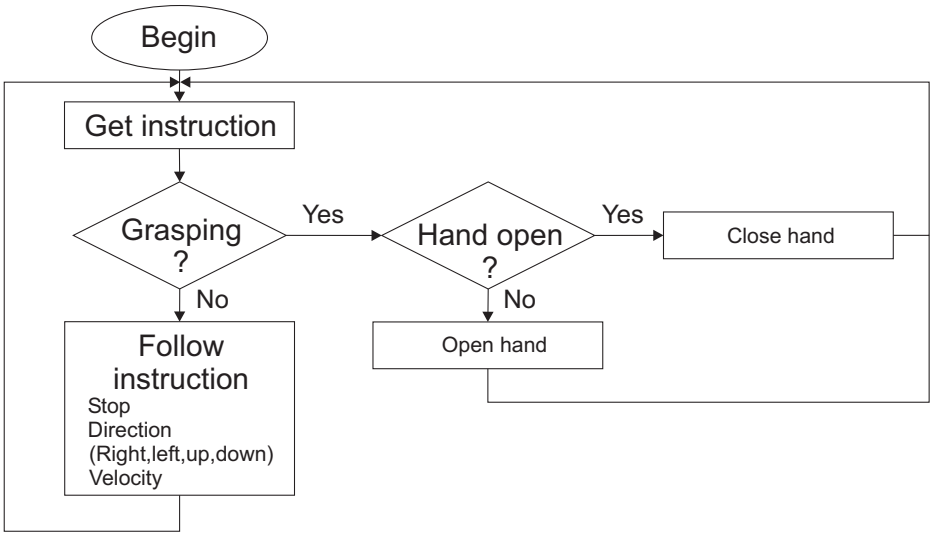


Figure 6.8: Flow diagram of the four directional control strategy. The strategy allows to change the position of the manipulator using vertical and horizontal steps and to grasp or to release objects. Seven commands can be sent to the robot arm: a stop order, four directions (right, left, up, down), grasp open, grasp close and motion velocity.

to open and to close the hand works the same as described for the four directional control. As seen in the general flow diagram of the vectorial control shown in Figure 6.9, the commands available to control the manipulator are stop, angle of direction, open the grasp or close the grasp accordingly. The vector directional control also allows to control the velocity of the manipulator through the implementation of a velocity profile that is based on the magnitude of the gaze point vector, as explained in Section 6.4.2.

A full description on the design and implementation of the two control strategies is given in Section 6.3 and Section 6.4.

6.3 Four directional control

The directional control instructs the manipulator to follow six commands: stop, right, left, up, down, open grasp or close grasp. The assignation of commands are made according to the position of the gaze point in the Cartesian space marked by the view scene image. The scene image can be considered as a Cartesian XY_{gt} plane with size 768×576 pixels, with the origin of the axes located at the upper

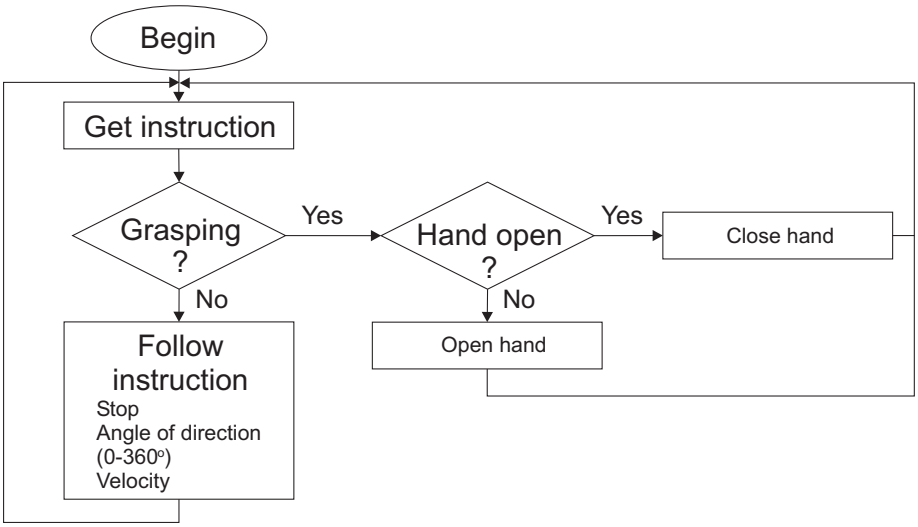


Figure 6.9: Flow diagram of the vectorial control strategy. Using this strategy it is possible to change the position of the manipulator through the execution of vertical, horizontal and diagonal steps by giving the angle of direction for the desired movement; and to grasp or release objects. Five commands are used in this strategy: the stop order, angle of direction (from 0 to 360 degrees), grasp open, grasp close and motion velocity.

left corner of the plane. In order to have a more versatile space, the gaze point is translated from the XY_{gt} space to another XY space with the origin located at $(384,288)_{gt}$ as shown in Figure 6.10.

Using the new Cartesian arrangement two important measurements are considered in the design of the control strategy: the *x ratio*, the *y ratio*. The values of the *x ratio* and the *y ratio* are used to identify the proportional distance of the gaze point with respect to the axes. The signs of the *x ratio* and the *y ratio* help to determine the direction of the vector drawn from the origin to the gaze point with respect to the origin. The arrangement of the conditions used to generate the direction instructions is explained in Table 6.1. Considering these conditions the XY plane is divided in five zones as shown in Figure 6.11.

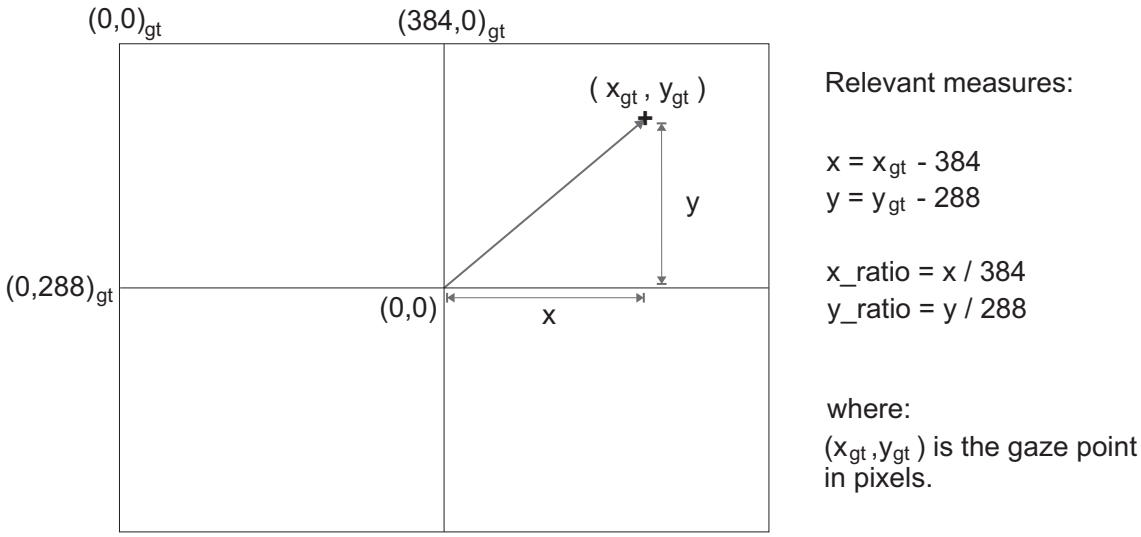


Figure 6.10: Map of translation from XY_{gt} space as given by the gaze tracking system to reference the gaze point, to the Cartesian XY space used in the design of the control strategies. The diagram shows the relevant measurements used in the Four Directional Control.

Instruction	Conditions		
Right	$x\ ratio > y\ ratio$	&	$x\ ratio > 0$
Left	$x\ ratio > y\ ratio$	&	$x\ ratio < 0$
Up	$x\ ratio < y\ ratio$	&	$y\ ratio > 0$
Down	$x\ ratio < y\ ratio$	&	$y\ ratio < 0$
Stop	$x\ ratio < 0.28$	&	$y\ ratio < 0.26$

Table 6.1: Generation of the direction instruction according to the position of the gaze point in the XY Cartesian space used in the four directional control

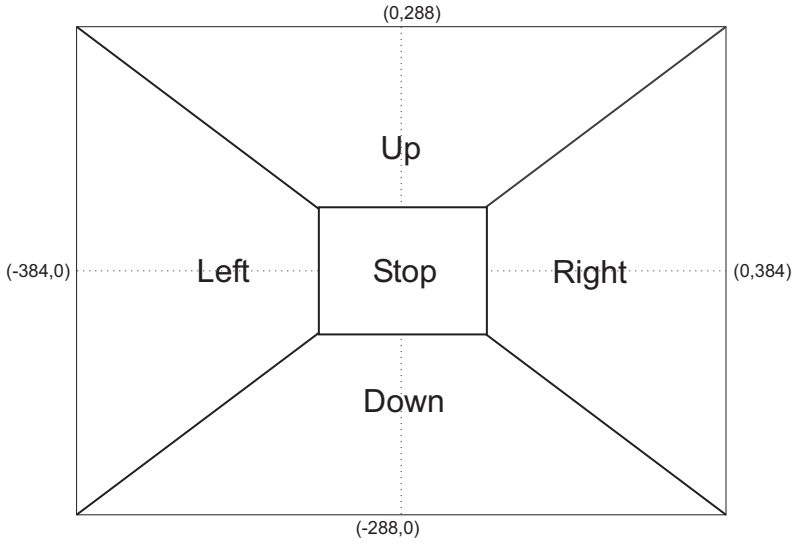


Figure 6.11: Map of instructions in the XY plane according to the conditions given in Table 6.1.

6.3.1 Filtering

The signal also needs to be filtered in order to compensate for the saccadic noise. For the design of the control strategy the signal is preferred to be as smooth and stable as possible, but no too slow so it allows a real time interaction. Two filtering techniques were implemented: a simple moving average filtering and a weighted moving average filtering.

Moving average (MA) filtering is a simple solution in digital applications, it works as a low pass filter by cutting high frequencies while keeping a sharp step response (Analog Devices Inc., 2013). The output of the filter y is the computation of the mean of the previous M values of the signal x

$$y[i] = \frac{1}{M} \sum_{j=-(M-1)}^{0} x[i+j] \qquad i = 1, 2..n \tag{6.1}$$

where M can be considered as a time window of variable size. A different version of the moving average filter is the weighted moving average (WMA) filter where the output y is the computation of the mean of the previous M values multiplied by a constant variable:

$$y[i] = \frac{M(M+1)}{2} \sum_{j=-(M-1)}^{0} (M+j)x[i+j] \qquad i = 1, 2..n \tag{6.2}$$

We considered three different time windows to filter the gaze point data: 150 ms, 300 ms and 500 ms, using both simple moving average and weighted moving average filtering. In order to analyse the effects of filtering for the three different time windows we took 5 seconds of data of a subject performing a fixation. Figure 6.12 and Figure 6.13 show the results of filtering the data segment using moving average. Plots a) and b) show the gaze point expressed in its X and Y coordinates, the signal obtained after filtering with the three different time windows is a delayed lowpass filtered version of the original signal. For both MA and WMA filters, when using

the 500 ms window, the signal is slower and the amplitude of the large saccades is noticeable reduced in comparison to the response of the signal when using the 150 ms and 300 ms windows. Plots c), d), e) and f) show the effect of lowpass filtering the gaze point in the cartesian space. The non filtered original signal is scattered around the fixation point but after filtering, a more continuous pattern was developed. MA filter with windows of 150 ms and 300 ms performed similar to WMA filter with 150 ms and 300 ms, respectively. MA filter with a 500 ms window had the most drastic smoothing effect and also presented the largest delay among the filtering strategies implemented.

6.3.2 Velocity profiles

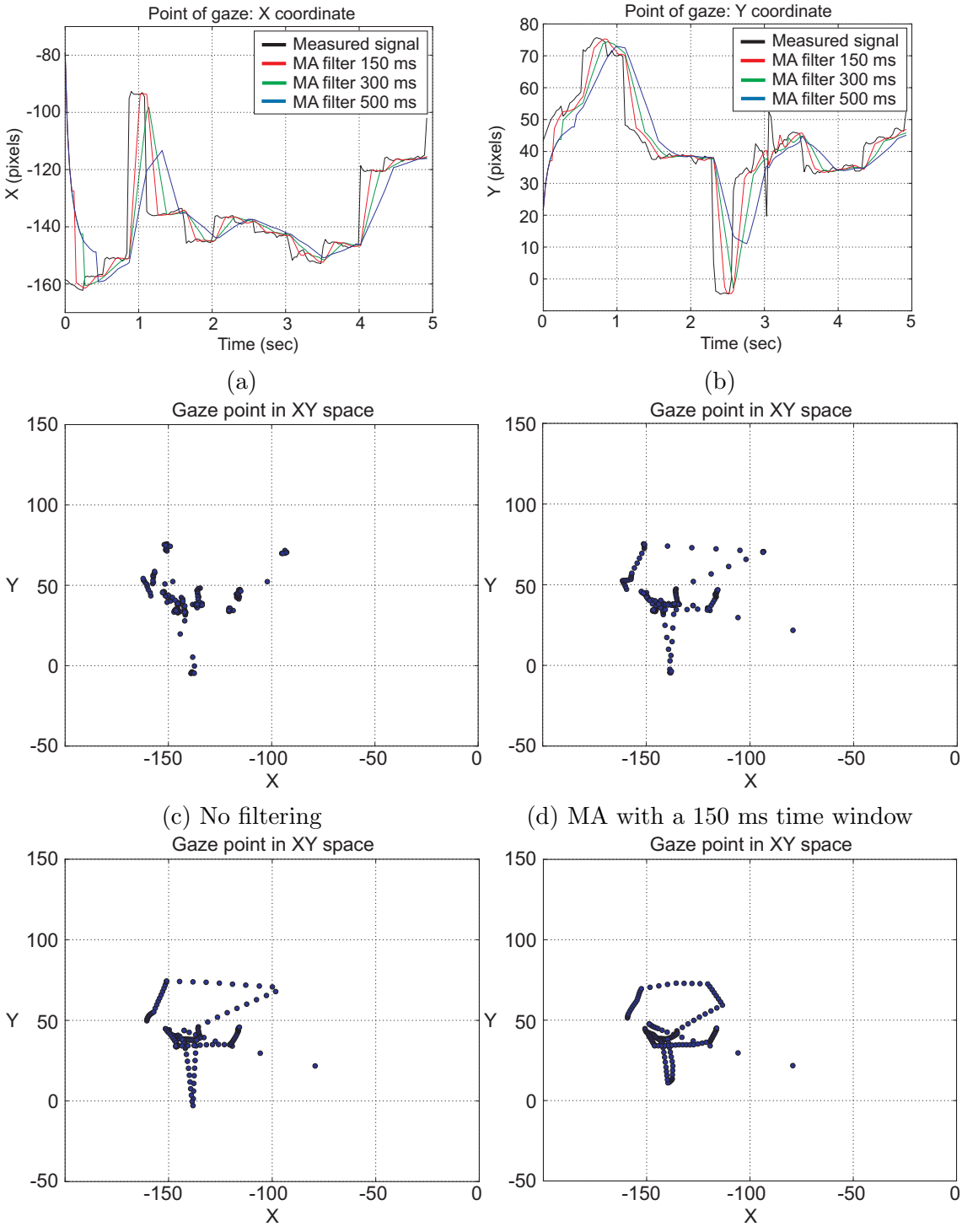
The position of the gaze point with respect to the Cartesian space XY was used to design the velocity profiles. According to this, the space is divided in several regions that represent the velocity value. Using the gaze point the user is able to change the velocity of the robot manipulator when executing a task. Three different velocity profiles were implemented: Single Velocity profile, Dual Velocity profile and Cubic Velocity profile.

Single Velocity profile

The Single Velocity profile allows to move the manipulator at a constant velocity only. The speed is slow so the user can easily make small corrections in the position of the hand with accuracy specially when grasping an object. As seen in Figure 6.14, for this strategy the XY plane is divided only in two regions: constant velocity and stop.

Dual Velocity profile

The Double Velocity profile consist in dividing the XY plane in three regions. One region is assigned to stop and the other two are slow velocity and fast velocity. Fast

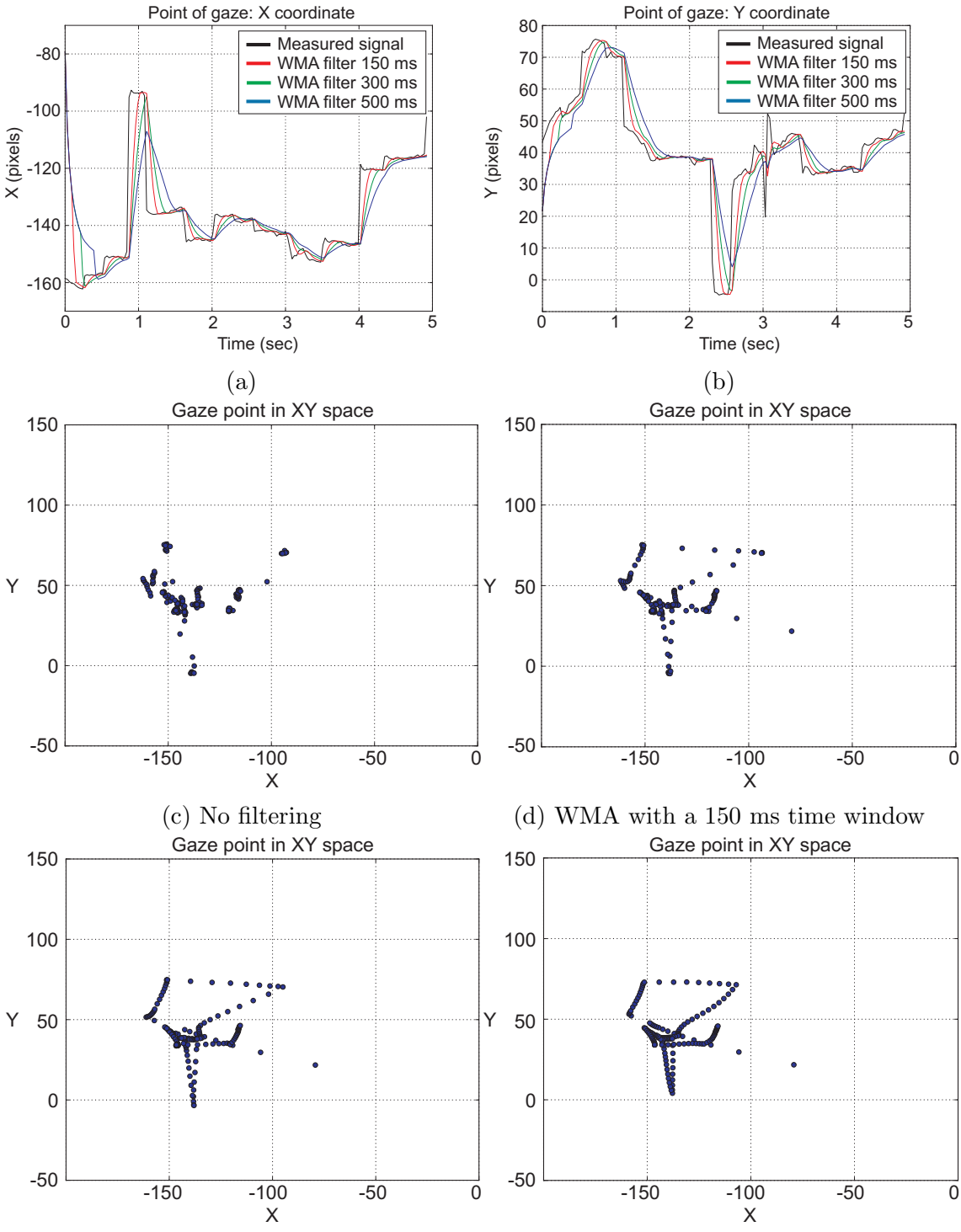


(a) (b)

(c) No filtering (d) MA with a 150 ms time window

(e) MA with a 300 ms time window (f) MA with a 500 ms time window

Figure 6.12: Response of a simple moving average filter when applied to the gaze point signal using different time windows. a) and b) show the X and Y coordinates of the gaze point during filtering. c), d), e) and f) show the gaze point in the XY Cartesian space.



(a) (b)

(c) No filtering (d) WMA with a 150 ms time window

(e) WMA with a 300 ms time window (f) WMA with a 500 ms time window

Figure 6.13: Response of a weighted moving average filter when applied to the gaze point signal using different time windows. a) and b) show the X and Y coordinates of the gaze point during filtering. c), d), e) and f) show the gaze point in the XY Cartesian space.

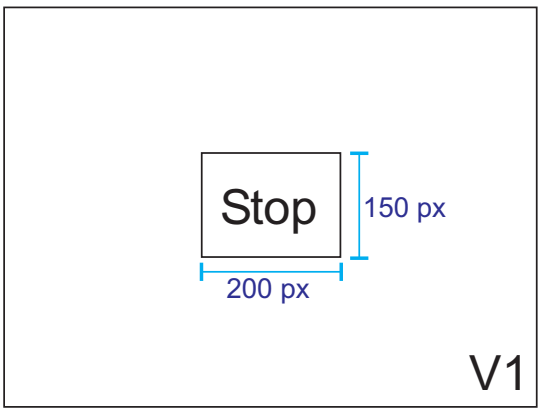


Figure 6.14: Single Velocity profile. The gaze point can fall only in two regions being one of them the Stop instruction. The size of each region is given in pixels.

velocity (V2) may be used to get close to the object while Slow velocity (V1) helps to position the hand more accurately. The Fast velocity is 2.5 times faster than the Slow velocity and it is activated far from the center of the XY plane, as explained in Figure 6.15. If we divided the positive X axis in the XY plane: the Stop region is located within 0 to 100 pixels, V1 is between 100 and 200 pixels, and V2 goes from 200 to 384 pixels. The positive Y axis is divided similarly, with the Stop region located within 0 to 75 pixels, the V1 is between 75 and 150 pixels, and V2 is between 150 to 288. Along the axes, Stop and V1 have the same length and V2 is slightly larger. This is because in practice, the user will have a tendency to maintain the gaze close to the Stop region when aiming for small adjustments in the position of the arm, so a bigger Slow region is not needed.

Cubic Velocity profile

The Cubic Velocity profile is shaped with the quantisation of a cubic function. The first design was chosen so that the velocity of the manipulator would change considering: slow increments near the stop region, linear increments in the mid area, a slow increments to the maximum speed and vice versa. During the first implementation of the cubic velocity profile we noticed that it was not convenient for the manipulator to keep changing velocities that often so we decided to enlarge the quantisation

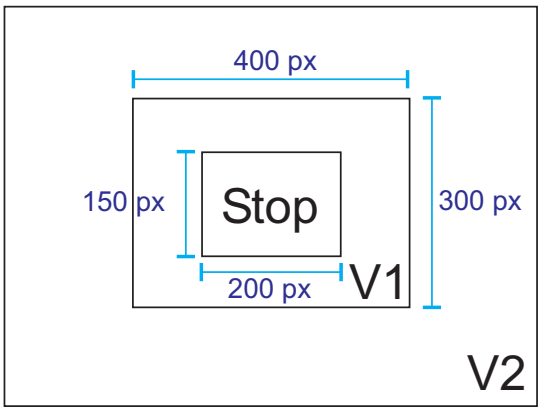


Figure 6.15: Double Velocity profile. The slow velocity V1 is activated in the area close to the stop region while the fast velocity V2 is located far from the center. The size of each region is given in pixels.

step size resulting in a velocity control that increased almost linearly. Figure 6.16 shows the quantisation of the cubic function (Equation (6.3.2)) used to divide the positive X axis of the XY plane in several velocity regions. The resulting Stop area of 50 pixels was too small, so we disposed from the region assigned to velocity 2 to enlarge it, velocity 2 is too slow anyway to move the robot arm. Then, the velocity arrangement in the positive X axis in pixels is as follows: Stop is between 0 to 100, V1 is between 101 to 161, V2 is between 162 to 215, V3 is between 216 to 269, V4 is between 270 to 323 and V5 is between 324 to 384. Similarly, the positive Y axis was divided as follows: Stop is between 0 to 75, V1 is between 76 to 125, V2 is between 126 to 161, V3 is between 162 to 197, V4 is between 198 to 233 and V5 is between 234 to 288. The resulting set of velocities let the velocities XY plane looking as shown in Figure 6.18. The regions assigned to the slow velocity V1 and fast velocity V5 are slightly larger than the regions given to the velocities in between. V2 is larger due to the arrangement used to enlarge the Stop region and in the case of V5, this happened due to the shape of the cubic function.

$$y = -1.7 \times 10^{-7}x^3 + 9.7 \times 10^{-5}x^2 + 7.8 \times 10^{-3}x - 0.011 \tag{6.2}$$

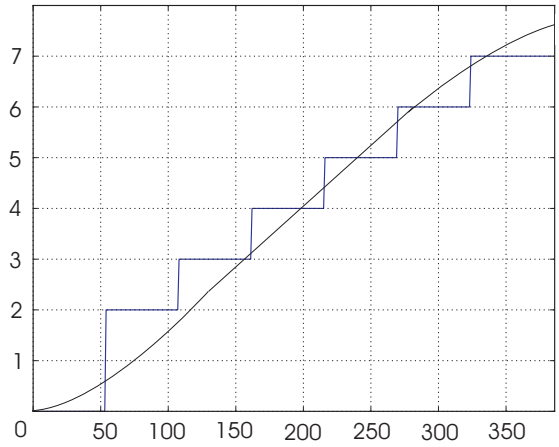


Figure 6.16: The Cubic Velocity profile is based on the quantisation of a cubic function.

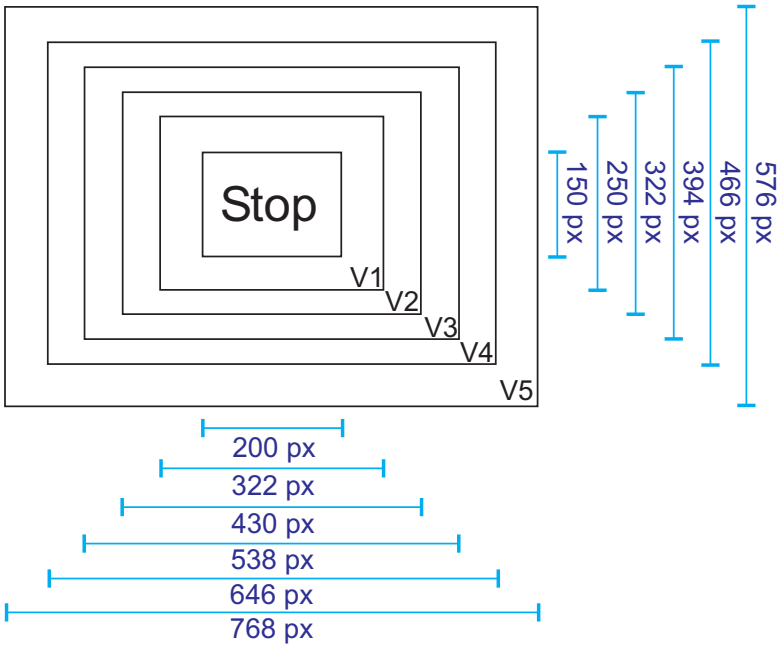


Figure 6.17: The Cubic Velocity profile offers five different velocities. The regions to activate velocity V1 and velocity V5 are slightly larger. The size of each region is given in pixels.

6.3.3 Experiments

The experiments were conducted on one single subject in order to find the suitability of the filtering techniques and the velocity profiles, analysing the implications that their implementation have in the real time response of the HMI based on gazed tracking. The setup for all the experiments consists in the subject wearing the head mounted gaze tracking positioned at the side of a table where the robot manipulator is fixed. The workspace for the experiments was a delimited area of the robot's kinematic workspace within the table, as shown in Figure 6.18. The workspace was marked to create fixed spots to be used as reference positions to place an object that the user intended to grasp during the experiments. The marked positions are shown in Figure 6.19.

For each experiment an object is placed in one of the 8 numbered marked positions, from 1 to 8, where the user tries to grasp it in 10 attempts for each position, giving a total of 80 trials per experiment. At the beginning of each trial the arm is positioned with the palm of the hand facing to the table right over the cross marked in the workspace, as shown in Figure 6.20. When the object is grasped it is considered a hit and we measured the time that took to grasp the object from the starting point (execution time).

The performance was measured in terms of the average number of hits and the average execution time.

Filtering

Five experiments were conducted in order to test each of the filtering techniques proposed for the four directional control: a) no filtering, b) MA with a window of 150 ms, c) MA with a window of 300 ms, c) MA with a window of 500 ms, d) WMA with a window of 150 ms and e) WMA with a window of 300 ms. The implementation of WMA with a window of 500 ms made the response of the system

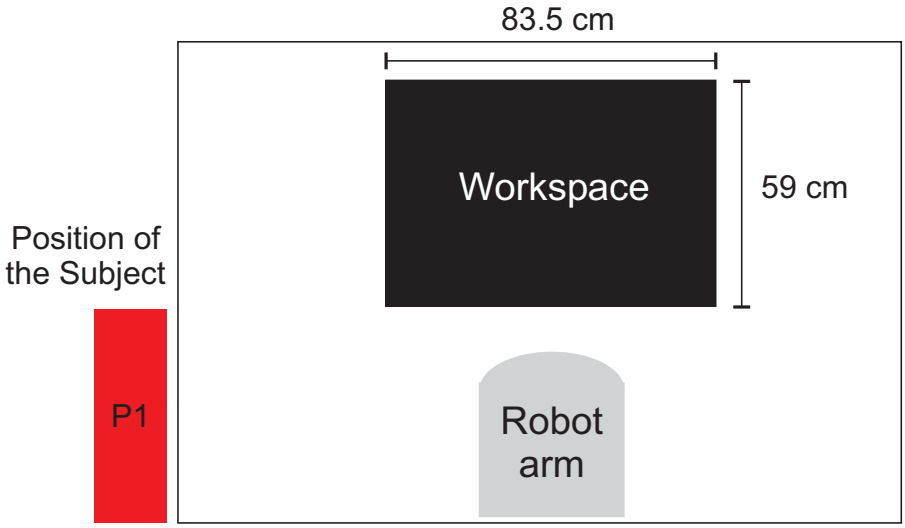


Figure 6.18: Setup for experiments.

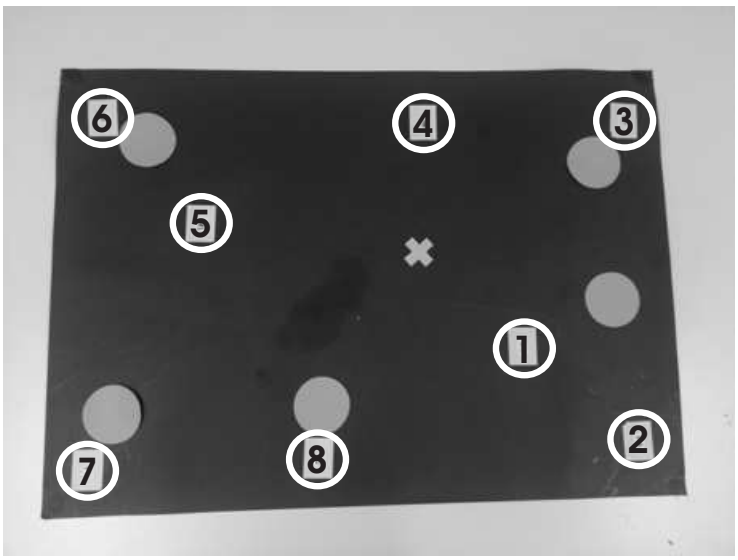
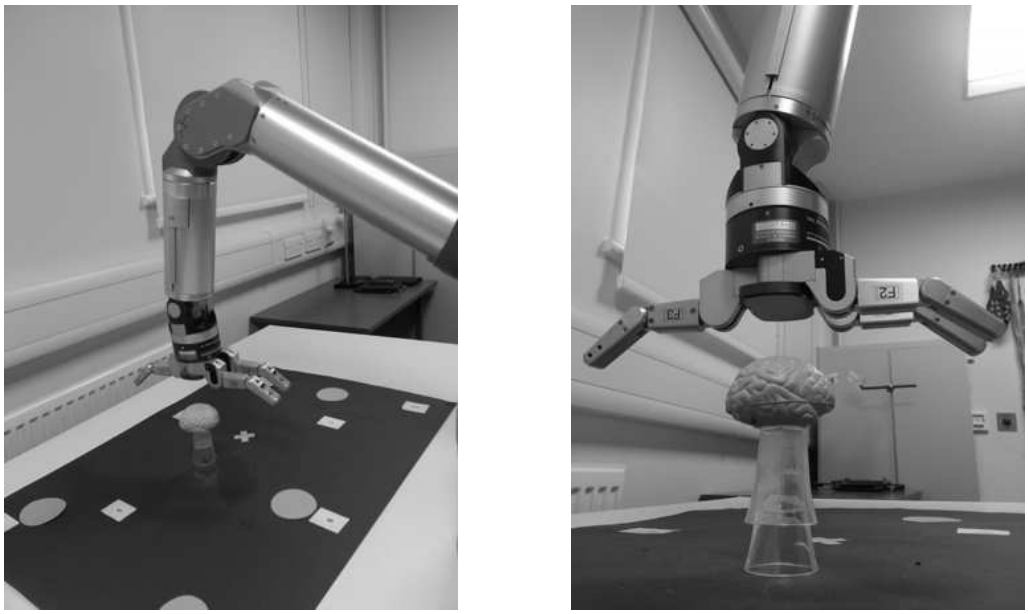


Figure 6.19: Workspace showing the eight numbered marked positions used in the experiments. The plain circles are used in a different setup of experiments, as explained later on, and the cross indicates the manipulator's initial position.



(a) (b)

Figure 6.20: a) Orientation of the hand while moving the robot manipulator in order to grasp the object from above. b) The target object used in the experiments.

too slow and therefore it did not meet the conditions to work in real time. For all the experiments the system was programmed to work under the Single Velocity profile. The results of the experiments are listed in Table 6.2 and Table 6.3. Table 6.2 shows the percentage of hits obtained according to the filtering technique used in each experiment. The highest percentages of hits were obtained with the moving average filter with a 300 ms window, having 93.7% of hits; the moving average filter with a 500 ms window, had 90.0% of hits; and the the weighted moving average with the 150 ms window, had 92.5% of hits. The lowest rates of hits were obtained when the signal was not filtered, resulting in 87.5% of hits; the moving average filter with a 150 ms window, had 85.0% of hits; and the weighted moving average filter of 300 ms, had 88.7% of hits.

Table 6.3 shows the execution time obtained according to the filtering technique used in each experiment. The shortest execution times were obtained with the moving average filter with a 300 ms window, having an average of 18.17 seconds time; the moving average with a 500 ms window had an average of 18.79 seconds time; and the weighted moving average with a 300 ms window had an average of 16.22 seconds

Percentage of Hits						
		MA			WMA	
Location	No filtering	150 ms	300 ms	500 ms	150 ms	300 ms
P1	90	80	90	90	100	90
P2	70	80	90	90	90	80
P3	100	90	90	90	90	90
P4	100	90	100	100	100	100
P5	80	90	100	80	100	70
P6	80	90	100	100	70	100
P7	80	90	90	90	90	90
P8	100	90	90	80	100	90
Average	87.5	87.5	93.7	90	92.5	88.7
SEM	11.64	4.62	5.17	7.55	10.35	9.91

Table 6.2: Performance of the system using different techniques to filter the gaze point signal. The test experiment consisted in placing an object in 8 different positions where the subject tried to grasp it, in 10 attempts for each position. The table shows the percentage of hits when executing the experiment without filtering the signal, when using moving average filters with time windows of 150 ms, 300 ms and 500 ms, and using weighted moving average filters with time windows of 150 ms and 300 ms.

time. The longest execution times were obtained when the signal was not filtered, with an average of 22.8 seconds time; the moving average filter with a 150 ms window had an average of 19.53 seconds time; and the weighted moving average filter of 150 ms had an average of 21.67 seconds time.

Velocity profiles

Three experiments were conducted in order to test each of the velocity profiles proposed for the four directional control: a) Single Velocity profile, b) Dual Velocity profile and c) Cubic Velocity profile.

The results of the experiments are listed in Table 6.4, the highest percentage of hits was obtained with the Single Velocity profile having 93% of hits. The Cubic Velocity profile had 81% of hits. The lowest percentage of hits was obtained when using the Dual Velocity profile, with 78% of hits. The shortest execution times corresponded to the Cubic Velocity profile, with an average time of 15.2 seconds. The Single Velocity profile had an average execution time of 18.4 seconds. The Dual Velocity profile had the largest execution time, with an average of 19.4 seconds time.

Execution Time

		MA			WMA	
Location	No filtering	150 ms	300 ms	500 ms	150 ms	300 ms
P1	16.7	12.8	11.3	12.2	16.4	14.2
P2	27.2	28.6	22.6	20.8	25.7	20.5
P3	26.3	22.0	22.8	22.3	26.3	19.5
P4	18.0	11.8	8.4	10.9	12.0	9.3
P5	26.1	18.4	16.8	17.5	23.9	17.8
P6	23.5	22.6	21.5	21.0	27.8	20.3
P7	27.1	24.8	26.3	26.7	26.4	28.0
P8	17.3	14.8	15.4	18.7	14.7	16.8
Average	22.8	19.53	18.17	18.79	21.67	16.22
SEM	4.65	6.02	6.21	5.22	6.23	5.40

Table 6.3: Performance of the system using the four directional control with different techniques to filter the gaze point signal. The test experiment consisted in placing an object in 8 different positions where the subject tried to grasp it, in 10 attempts for each position. The table shows the execution time of the experiment when the signal is not filtered, when using moving average filters with time windows of 150 ms, 300 ms and 500 ms, and using weighted moving average filters with time windows of 150 ms and 300 ms.

Velocity Profiles

	Percentage of Hits			Execution Time		
Location	Single	Dual	Cubic	Single	Dual	Cubic
P1	100	100	90	11.6	13.2	10.6
P2	90	90	80	21.7	22.1	19.1
P3	100	80	90	22.1	27.1	19.3
P4	100	70	100	8.1	14.8	7.8
P5	70	40	60	18.2	23.7	19.5
P6	90	70	70	22.6	18.0	15.7
P7	100	80	70	25.8	20.0	18.5
P8	100	100	90	17.4	16.4	11.4
Average	93.7	78.7	81.2	18.4	19.4	15.2
SEM	10.60	19.59	13.56	5.98	4.72	4.65

Table 6.4: Performance of the system using the four directional control with different velocity profiles. The test experiment consisted in placing an object in 8 different positions where the subject tried to grasp it, in 10 attempts for each position. The table shows the percentage of hits and the execution time of the experiments.

ANOVA

		Sum of Squares	df	Mean Square	F	Sig.
Hits	Between Groups	0.0275	5	0.0055	0.74	0.5984
	Within Groups	0.3125	42	0.00744		
	Total	0.34	47			
Execution Time	Between Groups	147.63	5	29.5265	0.92	0.4763
	Within Groups	1344.95	42	32.0226		
	Total	1492.58	47			

Table 6.5: Results of the one way analysis of variance (ANOVA) of the percentage of hits and execution time obtained with the implementation of different filters for the gaze point.

Summary

The results of the experiments during the initial optimisation of the HMI based on gaze tracking were used to perform an analysis of variance (ANOVA) in order to decide which filter was more suitable to implement in the Four Directional Control. A one way analysis of variance of the results is shown in Table 6.5. The results of the analysis on the percentage of hits was not significant, $F(5,42)= 0.74$, $p= 0.5984$. Moving average with a 300 ms window had the highest percentage of hits (Mean=0.973, SD=0.05), and the lowest percentage of hits was obtained when the gaze point was not filtered (Mean=0.875, SD=0.11). The results of the analysis on the execution time were also not significant, $F(5,42)= 0.92$, $p= 0.4763$. Weighted moving average with a 300 ms window had the best execution time (Mean=16.22, SD=5.40), while moving average with a 300 ms window had the second best execution time (Mean=18.17, SD=5.22). The longest execution time was obtained when the gaze point was not filtered (Mean=22.8, SD=4.65). Moving average with a 300 ms window was implemented to filter the gaze point in the Four Directional Control.

The results obtained when testing the three velocity profiles proposed for the Four Directional Control were also used to perform an analysis of variance (ANOVA), as shown in Table 6.6. The results of the analysis on the percentage of hits was not significant, $F(2,21)= 2.28$, $p= 0.1272$. The higher percentage of hits was obtained when using the Single Velocity profile (Mean=0.93, SD=0.1), while the subject had

ANOVA

		Sum of Squares	df	Mean Square	F	Sig.
Hits	Between Groups	0.10333	2	0.05167	2.28	0.1272
	Within Groups	0.47625	21	0.02268		
	Total	0.57958	23			
Execution Time	Between Groups	76.323	2	38.1617	1.44	0.2604
	Within Groups	558.326	21	26.587		
	Total	634.65	23			

Table 6.6: Results of the one way analysis of variance (ANOVA) of the percentage of hits and execution time obtained with the implementation of the three different velocity profiles proposed for the Four Directional Control.

the lowest rate of hits when using the Dual Velocity profile (Mean=0.78, SD=0.19). The results of the analysis on the execution time were also not significant, $F(2,21)= 1.44$, $p= 0.2604$. The best execution time was obtained when using the Cubic Velocity profile (Mean=15.2, SD=4.65) and the poorest execution time was obtained with the Dual Velocity profile (Mean=19.4, SD=4.72).

6.4 Vector directional control

The gaze point in the Vector directional control is considered as a vector referenced to the origin of the XY plane, as shown in Figure 6.21. The angle of direction of the gaze point vector is translated onto the robot manipulator's space on a way that the user is able to guide the end effector position using horizontal motion, vertical motion, diagonals and curves. The magnitude of the vector can be considered to generate three different velocity profiles, as described later.

6.4.1 Filtering

The implementation of the Vector directional control requires a more sophisticated filtering strategy than the ones suggested for the four directional control due to the saccadic movement of the eye, which causes instability in the direction of the gaze point vector, thus affecting the desired motion instruction. We proposed a

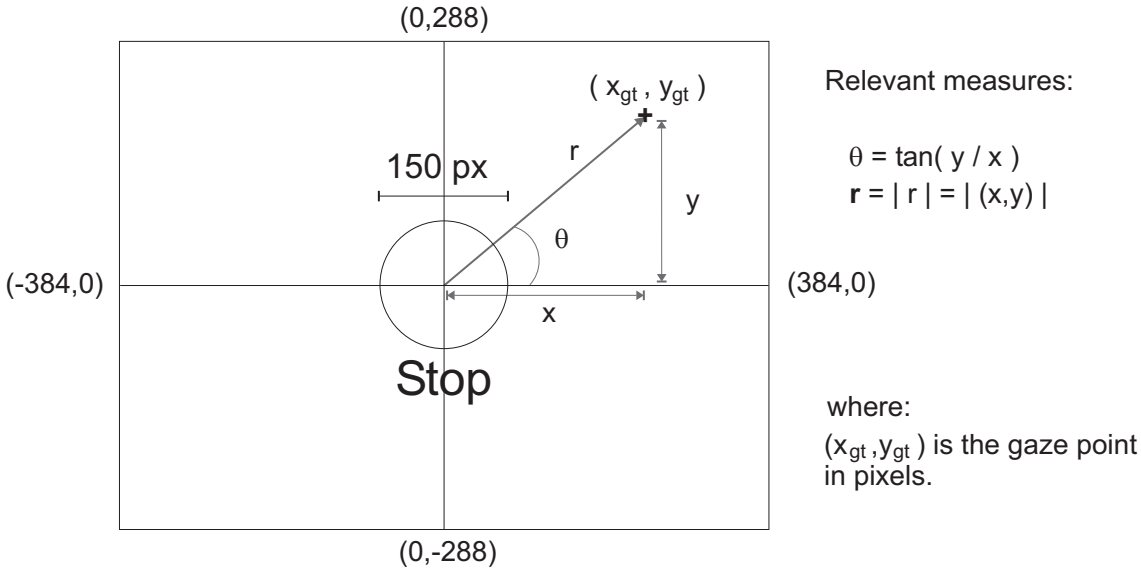


Figure 6.21: Relevant measurements used in the Vector Directional Control expressed in the XY Cartesian space.

Kalman filter as a filter strategy for the Vector directional control due to its ability of smoothing the input signal while predicting future incomes. Kalman filters work as predictor-corrector estimators using a specific set of equations based on the covariance of the error and a previous measurement of the variable to estimate (Welch and Bishop, 2013). For a given controlled process represented as a linear system

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_k + w_{k-1} \\ z_k &= Hx_k + v_k \end{aligned}$$

the process noise w_k and measurement noise v_k have respective covariances of values Q and R.

The estimating process of a discrete Kalman filter is according to the diagram in Figure 6.22, the filter is fed with initial values of the variable to estimate $\hat{\mathrm{x}}_k$ and the error covariance P_k, this information is used to make a prediction that is later corrected through a measurement z_k. The value of the error covariance is updated and used to feedback the filter.

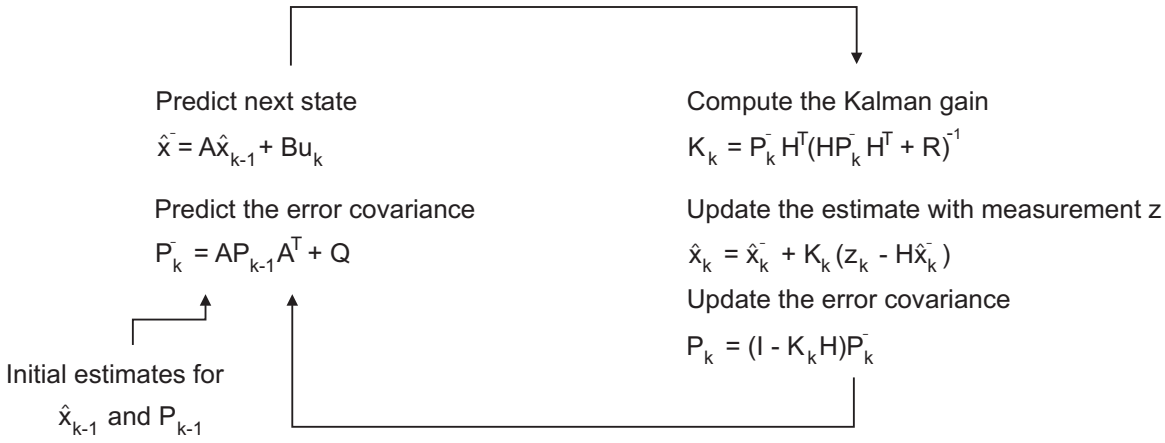
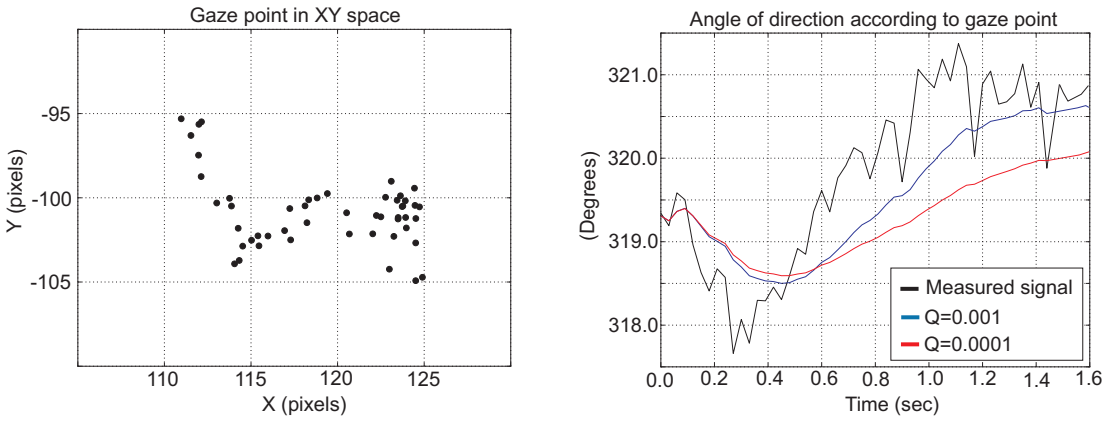


Figure 6.22: Diagram of operation of the discrete Kalman filter, the filter takes initial conditions in order to make a prediction of the signal and uses a measurement value to correct the prediction in order to update the process (Welch and Bishop, 2013).

The implementation of the Kalman filter with small process noise covariances works as a low pass filter, helping to smooth the noisy signal due to the saccadic movement of the eye. The filter was implemented for two different values of the process noise covariance Q=0.001 and Q=0.0001. Figure 6.23 shows the results of filtering 1.6 seconds of the angle of direction given by the gaze point vector when the subject is fixating. The plot in a) is the cartesian representation of the gaze point in the XY plane. The angle of direction of the gaze point vector is shown in b). Kalman filter with noise covariance Q=0.001 removes the noise in the signal while Kalman filter with noise covariance Q=0.0001, not only removes the noise but makes the signal slower.

6.4.2 Velocity profiles

Three different velocity profiles were implemented on the Vector directional strategy: Single Velocity profile, Dual Velocity profile and Cubic Velocity profile. The magnitude of the gaze point vector was used as reference in the assignation of the velocity value. According to the profile, the larger the magnitude of the vector the fastest the manipulator would move. The theory behind the velocity profiles is the same as explained in Section 6.3.2, only that the regions are now defined by concentric circles instead of rectangles. Figure 6.24 shows the velocity regions assigned for the



(a) (b)

Figure 6.23: a) Gaze point in the XY plane and b) its angle of direction. The angle of direction is filtered through a Kalman filter using a process noise covariance value of Q=0.001 and Q=0.0001.

Single Velocity profile based on the XY plane, Figure 6.25 shows the velocity regions assigned for the Double Velocity profile, and Figure 6.26 shows the velocity regions assigned for the Cubic Velocity profile.

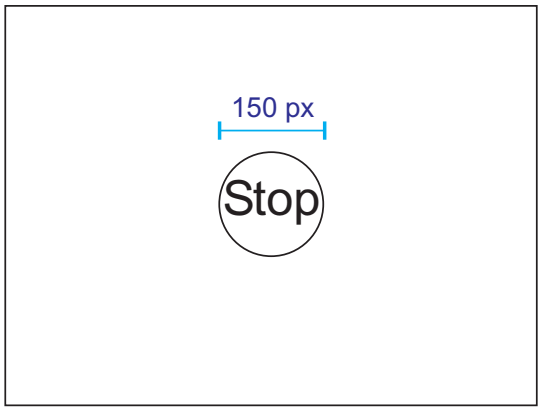


Figure 6.24: Single Velocity profile. The gaze point can fall only in two regions being one of them the Stop instruction.

6.4.3 Experiments

As for the Four Directional Control, several experiments were conducted on one single subject in order to find the suitability of the filtering techniques and the velocity profiles proposed for the Vector Directional Control. The setup used in the experiments is the same as described in Section 6.3.3.

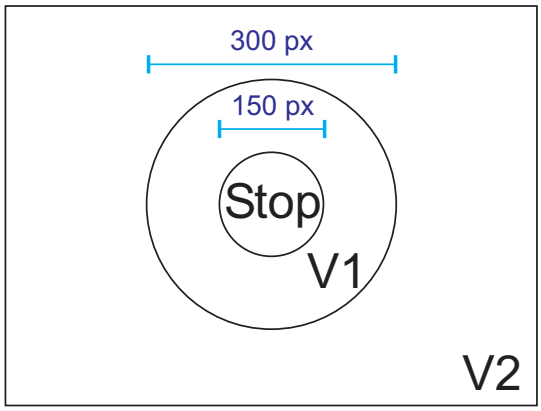


Figure 6.25: Double Velocity profile. The slow velocity V1 is activated in the area close to the stop region while the fast velocity V2 is located far from the center. V2 is 2.5 times V1.

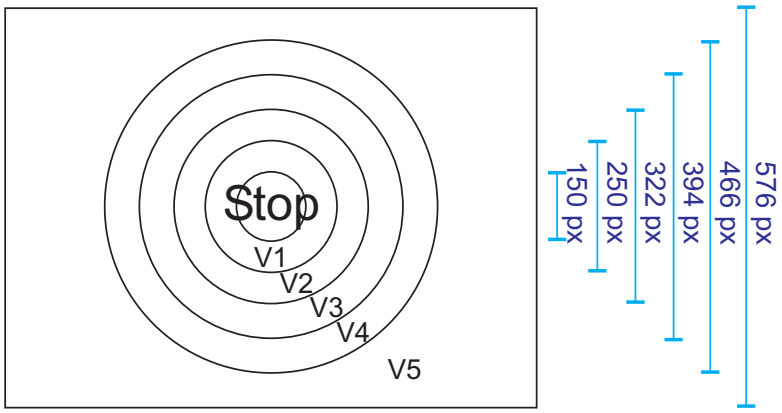


Figure 6.26: The Cubic Velocity profile is based on the quantisation of a cubic function. The profile offers five incremental velocities starting from the center towards the limits of the space.

Kalman Filtering				
	Percentage of Hits		Execution Time	
Location	Q=0.001	Q=0.0001	Q=0.001	Q=0.0001
P1	100	80	8.7	14.2
P2	80	100	19.6	21.8
P3	100	60	18.2	26.0
P4	100	90	9.2	12.1
P5	70	60	16.0	24.5
P6	100	100	17.8	15.7
P7	100	100	24.5	20.6
P8	90	90	10.7	11.6
Average	92.5	85	14.2	16.8
SEM	11.64	16.90	5.60	5.63

Table 6.7: Performance of the system with vector directional control using Kalman filters with covariance noise of Q=0.001 and Q=0.0001, respectively, to filter the direction of the gaze point vector. The test experiment consisted in placing an object in 8 different positions where the subject tried to grasp it, in 10 attempts for each position. The table shows the percentage of hits and the execution time of the experiments, obtained when using the Single Velocity profile.

Filtering

Two experiments were conducted: a) Kalman filter with noise covariance Q=0.001 and b) Kalman filter with noise covariance Q=0.0001. The implementation of the filter with less than Q=0.0001 made the response of the system too slow and therefore it did not meet the conditions to work with the system in real time. On the other hand, the system could not be tested without filtering due to the instability of the control signal.

In all the experiments the system was programmed to work under the Single Velocity profile. The results of the experiments are shown in Table 6.7. Kalman filter with noise covariance Q=0.001 obtained the highest percentage of hits and also the shortest execution time with an average time of 14.2 seconds. Kalman filter with noise covariance of Q=0.0001 obtained 86% of hits with an average execution time of 16.8 seconds.

Velocity profiles

Three experiments were performed to test the velocity profiles proposed for the vector directional control: a) Single Velocity profile, b) Dual Velocity profile and c)

Velocity Profiles						
	Percentage of Hits			Execution Time		
Location	Single	Dual	Cubic	Single	Dual	Cubic
P1	100	90	90	8.7	9.6	7.1
P2	80	90	70	19.6	16.0	14.1
P3	100	100	100	18.2	14.8	13.5
P4	100	90	100	9.2	9.0	6.5
P5	70	90	60	16.0	12.7	11.8
P6	100	80	100	17.8	11.3	13.9
P7	100	100	100	24.5	15.5	14.4
P8	90	90	90	10.7	10.4	12.6
Average	92.5	91.2	88.7	15.6	12.4	11.7
SEM	11.64	6.40	15.52	5.60	2.75	3.26

Table 6.8: Performance of the system using the vector directional control with different velocity profiles. The test experiment consisted in placing an object in 8 different positions where the subject tried to grasp it, in 10 attempts for each position. The table shows the percentage of hits and the execution time of the experiments.

Cubic Velocity profile.

The results of the experiments are listed in Table 6.8, the highest percentage of hits was obtained when using the Single Velocity profile, having 92%; the Dual Velocity profile had 91% of hits, while the Dual Velocity profile had the lowest percentage of hits, with 88%. The shortest execution times corresponded to the Cubic Velocity profile with an average of 11.7 seconds time. The Single Velocity profile had an average execution time of 15.6 seconds, while the Dual Velocity profile had an average execution time of 12.4 seconds.

Summary

According to the results of the experiments using the Vector Directional Control, Kalman filtering with a process noise covariance of Q=0.001 had the highest percentage of hits (Mean=0.92, SD=0.11), against (Mean=0.86, SD=0.16) obtained when using a Kalman filter with process noise covariance Q=0.0001. We proposed a Kalman filter with Q=0.001 to be implemented in order to filter the gaze point in the Vector Directional Control. Considering the execution time, the subject was faster when using Kalman filter with process noise covariance Q=0.001 (Mean=14.2, SD=5.60) in comparison with the time obtained when

ANOVA

		Sum of Squares	df	Mean Square	F	Sig.
Hits	Between Groups	0.00583	2	0.00292	0.21	0.8127
	Within Groups	0.2925	21	0.01393		
	Total	0.29833	23			
Execution Time	Between Groups	67.623	2	33.8117	2.07	0.1509
	Within Groups	342.716	21	16.3198		
	Total	410.34	23			

Table 6.9: Results of the one way analysis of variance (ANOVA) of the percentage of hits and execution time obtained with the implementation of the three different velocity profiles proposed for the Vector directional control.

using Kalman filter with process noise covariance Q=0.0001 (Mean=16.8, SD=5.63).

The results obtained when testing the three velocity profiles proposed for the Vector Directional Control were used to perform an analysis of variance (ANOVA). The results of the analysis are shown in Table 6.9. On the implementation of different velocity profiles, the results of the percentage of hits were not significant, $F(2,21)= 0.21$, $p= 0.8127$. The higher percentage of hits was obtained when using the Single Velocity profile (Mean=0.92, SD=0.11), while the subject had the lowest rate of hits when using the Cubic Velocity profile (Mean=0.88, SD=0.15). The results of the analysis on the execution time were also not significant, $F(2,21)= 2.07$, $p= 0.1509$. The best execution time was obtained when using the Cubic Velocity profile (Mean=11.7, SD=3.26) and the poorest execution time was obtained with the Single Velocity profile (Mean=15.6, SD=5.60).

6.5 Performance of the System on different subjects

The performance of the HMI based on gaze tracking was tested through the execution of 6 experiments in 9 different subjects. The objective of testing the system in different subjects was to analyse the suitability of using the two proposed controls: the Four Directional Control and the Vector Directional Control. Also, the perform-

ance of the subjects during the experiments allowed to test the implementation of the three velocity profiles under each control strategy.

6.5.1 Experiments

The workspace used in the experiments with the subjects is shown in Figure 6.27. The workspace was modified considering that the subjects were completely naive to the HMI based on gaze tracking and some of the positions used in the experiments performed during the optimisation stage were quite challenging. Specially those positions that were too close to the kinematic limits of the manipulator or those that fell out of the visual space of the subject, like the position marked as 2 which view was obstructed by the base of the manipulator.

For each experiment the object was placed in one of the 8 numbered marked positions, from 1 to 8, and the user tried to grasp it in one attempt, giving a total of 8 trials per experiment. The subject was positioned at the left side of the robot manipulator.

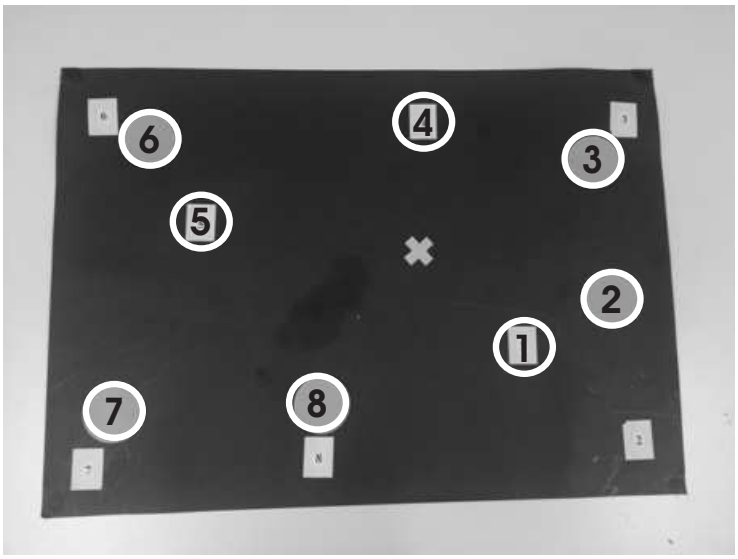


Figure 6.27: Workspace showing the eight numbered positions marked by white circles that were used in the experiments. The numbered squares were part of the setup used in the initial optimisation of the system and the cross marked the initial position of the arm.

The order and description of the control strategy used in each of the six experiments as executed by the subjects were:

1. Four directional control with single velocity profile
2. Four directional control with dual velocity profile
3. Four directional control with cubic velocity profile
4. Vector directional control with single velocity profile
5. Vector directional control with dual velocity profile
6. Vector directional control with velocity velocity profile

6.5.2 Results

The results of the experiments are listed in Table 6.10. Subject 1 had its highest percentage of hits of 75% when using the Four Directional Control with Dual Velocity profile, and obtained its shortest execution time of 13 seconds when using the Four Directional Control with Cubic Velocity profile. Subject 2 had its highest percentage of hits of 87% when using the Four Directional Control with Dual Velocity profile, and obtained its shortest execution time of 13 seconds when using the Vector Directional Control with Cubic Velocity profile. Subject 3 had its highest percentage of hits of 62% when using the Vector Directional Control with Dual Velocity profile, and obtained its shortest execution time of 17.3 seconds when using the Vector Directional Control with Cubic Velocity profile. Subject 4 had its highest percentage of hits of 75% when using the Four Directional Control with Cubic Velocity profile, and obtained its shortest execution time of 13.7 seconds when using the Vector Directional Control with Cubic Velocity profile. Subject 5 had its highest percentage of hits of 87% when using the Four Directional Control with Single Velocity profile and with Dual Velocity profile, and obtained its shortest execution time of 13 seconds when using the Four Directional Control with Dual Velocity profile. Subject 6 had its highest percentage of hits of 75% when using the Four Directional Control with Dual Velocity profile and the

Vector Directional Control with Single Velocity profile, and obtained its shortest execution time of 17.2 seconds when using the Vector Directional Control with Dual Velocity profile. Subject 7 had its highest percentage of hits of 75% when using the Four Directional Control with Dual Velocity profile and the Cubic Velocity profile, and obtained its shortest execution time of 13.3 seconds when using the Four Directional Control with Cubic Velocity profile. Subject 8 had its highest percentage of hits of 75% when using the Four Directional Control with Dual Velocity profile and the Vector Directional Control with Single Velocity profile, and obtained its shortest execution time of 19 seconds when using the Four Directional Control with Cubic Velocity profile. Subject 9 was not able to complete any experiment that involved velocity control and could only move the arm when the system worked under the single velocity profile. For subject 9, the highest percentage of hits of 62% was obtained when using the Vector Directional Control with Single Velocity profile, and had its shortest execution time of 27.3 seconds when using the Four Directional Control with Single Velocity profile.

Figure 6.28 shows a plot with the mean of the results of the experiments considering all the subjects. According to the graphs, the highest percentage of hits was obtained when the subjects used the Four Directional Control with the Double Velocity profile, while the lowest rate of success happened when using the Vector Directional Control with Cubic Velocity profile. The mean of the results also shows that the longest execution time was obtained when using the Four Directional Control and the Vector Directional Control with Single Velocity profile, and the shortest execution times occurred with the Vector Directional Control configured with Dual Velocity profile and Cubic Velocity profile.

If we considered the median of the results of the experiments shown in Figure 6.29, the highest percentage of hits was obtained when the subjects used the Four Directional Control with the Double Velocity profile, while the lowest rate of

success happened when using the Vector Directional Control with Cubic Velocity profile and the Four Directional Control with Single Velocity profile. The median of the execution time shows that the subjects took more time when using the Four Directional Control with Single Velocity profile, while the shortest execution times occurred with the Vector Directional Control configured with Dual Velocity profile and Cubic Velocity profile.

A one way analysis of variance of the results of the experiments on subjects is shown in Table 6.11. The results of the analysis on the percentage of hits was significant, $F(5,42)= 2.86$, $p= 0.0262$. Four Directional Control with Dual Velocity profile had the highest percentage of hits (Mean=0.71, SD=0.18). The second best strategy was the Four Directional Control with Cubic Velocity profile. Vector directional control with Cubic Velocity profile showed the poorest performance (Mean=0.48, SD=0.14). The results of the analysis on the execution time were not significant, $F(5,42)= 2.24$, $p= 0.068$. Vector directional control with Dual Velocity profile had the best execution time (Mean=18.0, SD=4.02). The second best rate was obtained with the Vector directional control with Cubic Velocity profile (Mean=20.4, SD=9.48). Four directional control with Single Velocity profile presented the longest execution time (Mean=29.6, SD=6.64).

6.6 Discussion

The HMI based on gaze tracking was successfully used by nine subjects to control a robot manipulator. The users were able to accomplish desired tasks such as reaching and grasping objects in one single session, testing two different control strategies with three velocity profiles. The proficiency in using each control strategy varied among subjects. Most of them were able to achieve more than 50% of hits using two or more strategies, except for subject 7 and subject 9. Subject 9 could

Subject	Percentage of Hits						Execution Time					
	E1	E2	E3	E4	E5	E6	E1	E2	E3	E4	E5	E6
1	37	75	62	37	50	25	27.3	25.3	13.0	18.6	15.7	16.5
2	32	87	75	75	25	62	28.8	25.2	24.6	22.3	11.0	10.0
3	50	37	37	50	62	37	26.7	18.0	17.6	17.5	23.8	17.3
4	50	50	75	50	62	50	42.5	35.5	39.3	17.5	16.6	13.7
5	87	87	62	50	25	62	30.7	17.1	25.2	55.7	17.5	39.2
6	37	75	62	75	50	62	38.0	17.3	26.6	38.6	17.2	21.4
7	62	75	75	62	62	50	24.6	16.1	13.3	21.0	20.6	16.2
8	50	87	75	87	62	37	21.2	25.1	19.0	37.5	22.0	29.3
9	37	-	-	62	-	-	27.3	-	-	31.8	-	-
Average	49.1	71.6	65.3	60.8	49.7	48.1	29.6	22.4	23.6	28.9	18.0	20.4
SEM	17.09	18.55	13.14	15.84	16.14	13.97	6.64	6.63	8.63	13.07	4.02	9.48

Table 6.10: Results of the experiments used to test the control strategies on different subjects. The rate of success and the execution time were computed for each subject and for all the experiments.

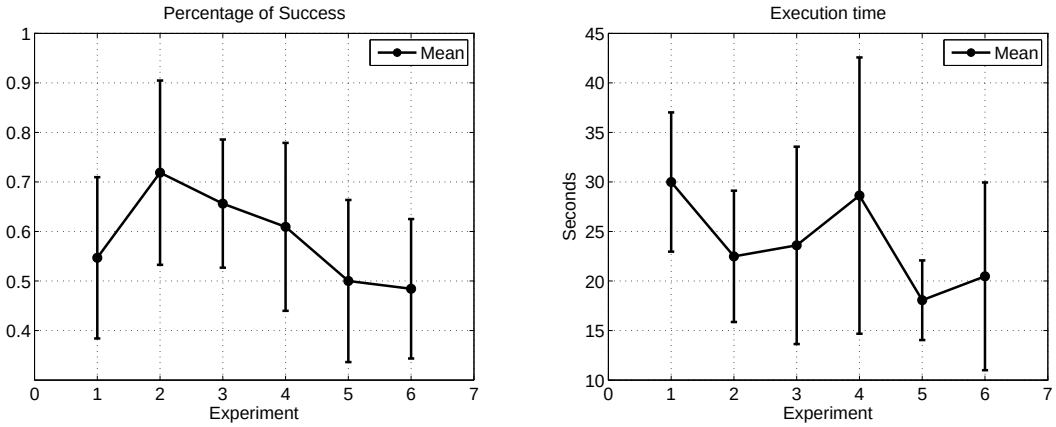


Figure 6.28: Computation of the mean of the percentage of hits and the execution time obtained in the six experiments used to test the control strategies on different subjects.

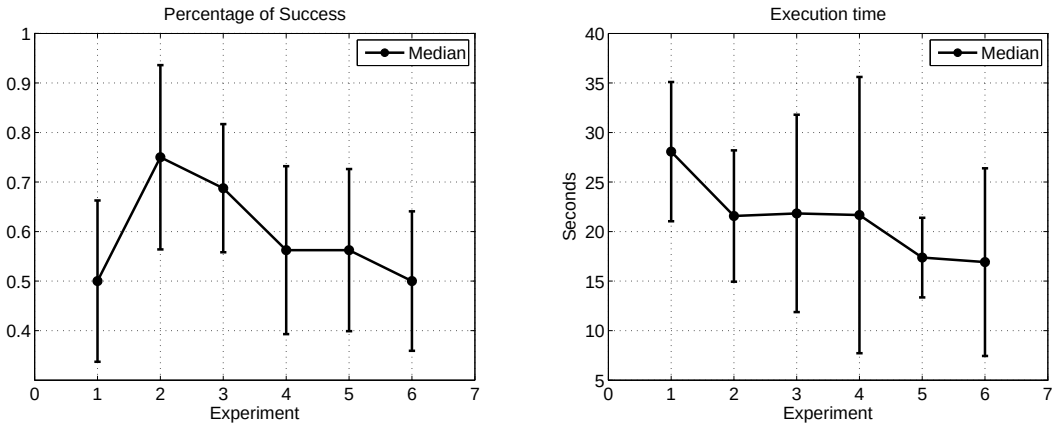


Figure 6.29: Computation of the median of the percentage of hits and the execution time obtained in the six experiments used to test the control strategies on different subjects.

ANOVA

		Sum of Squares	df	Mean Square	F	Sig.
Hits	Between Groups	0.37364	5	0.07473	2.86	0.0262
	Within Groups	1.09915	42	0.02617		
	Total	1.47279	47			
Execution Time	Between Groups	873.44	5	174.687	2.24	0.068
	Within Groups	3277.04	42	78.025		
	Total	4150.47	47			

Table 6.11: Results of the one way analysis of variance (ANOVA) of the percentage of hits and execution time obtained in the six experiments used to test the control strategies on different subjects.

only execute the reaching and grasping tasks in the experiments that did not involve velocity control, but still registered a percentage of hits of 62% using the Vector Directional Control with Single Velocity profile. The highest percentage of hits of Subject 1 was 75% with the Four Directional Control and Dual Velocity profile. Subject 2 had 87% of hits with the Four Directional Control and Dual Velocity profile. Subject 3 had 62% of hits using the Vector Directional Control with Cubic Velocity profile. Subject 4 had 75% of hits with Four Directional Control and Cubic Velocity profile. Subject 5, 6, 7 and 8 reached their highest percentage of hits with two different strategies. Subject 5 registered 87% of hits using the Four Directional Control with Single and Dual Velocity profiles, respectively. Subject 6 had 75% of hits with the Four Directional Control and Dual Velocity profile and the Vector Directional Control with Single Velocity profile. Subject 7 had 75% of hits using the Four Directional Control with Single and Dual Velocity profiles. Subject 8 had 87% of hits using the Four Directional Control and Dual Velocity profile and the Vector Directional Control with Single Velocity profile.

Considering the average of the results, the highest percentage of hits was 71%, obtained when using the Four Directional Control with Dual Velocity profile. The Vector Directional Control with Cubic Velocity profile registered the lowest average percentage of hits with 48%.

The results obtained in the experiments with subjects put our HMI based on gaze tracking under consideration to be compared to past implementations of interfaces

based on brain activity, or Brain Computer Interfaces (BCI), that controlled robot manipulators in the execution of similar reaching and grasping tasks.

For example, Velliste and colleagues (Velliste et al., 2008) used cortical activity patterns on monkeys to control a robot arm in a self feeding task. They used intracortical microelectrode arrays implanted in the motor cortex of the monkeys. The monkeys underwent several training sessions to operate the robot arm and to learn the self-feeding task. After the initial training period the monkeys executed the self-feeding task continuously for several days. One of the monkeys performed the task for 2 days with a success rate of 61%, while the other monkey executed a simpler version of the task for 13 days with a success rate of 78%.
In a similar implementation on people with tetraplegia Hochberg and colleagues presented in (Hochberg et al., 2012) a neural interface-system based control of a robotic arm. The patients were implanted with a 96-channel microelectrode array used to record neural activity in the motor cortex and had sessions on a near-weekly basis to learn how to control a computer cursor. The participants were asked to control two different right handed robot arms to reach and grasp objects in three dimensional movements. One of the subjects obtained success rates of 21.3% and 46.2%, using each of the robot arms in four sessions. The other subject grasped the object 62% of the times in one session.

It might seems unfair to compare the performance of the HMI based on gaze tracking to control a robot manipulator presented in this work with the BCI implementations mentioned above, since there are remarkable differences in the setup of the systems. For instance, our HMI is used to control the robot manipulator using two dimensional movements while Velliste and Hochberg presented controls for three dimensional motion. However, the merit of our implementation resides in the fact that the system is completely not invasive which means that the user would

not have to undergo any surgery for sensor's implantation, reducing the health risks that accompany the surgery itself, open wounds treatment, foreign object rejection, etcetera.

As mentioned, the successful control of cortical neural prosthetics involve long periods of training and the use of complex algorithms. The initial period of training for the presented HMI based on gaze tracking takes advantage of the user's own understanding of the eye movements and their relation with the directional control strategy implemented on the system. In our experience with subjects during the validation process, a simple description of how the gaze tracking system works is enough for the subject to understand the concept of the visual space given by the scene image and how it is used to reference the control strategy. This step takes less than half an hour even in subjects who are completely naive to the eye/gaze tracking concept.

The HMI based on gaze tracking presented in this work seems suitable to be used as an alternative to control a robot manipulator and to be considered for future implementations of prosthetics intended to aid paralysed people. The present system is completely prompt to be extended in order to implement a three dimensional motion control able to fulfil the necessities of the patients. Three dimensional motion may be accomplished by taking advantage of blinking sequences using them as binary control signals, or by introducing the concept of depth on the gaze point signal processing when designing the directional control strategy.

6.7 Conclusions

In this chapter we described a Human Machine Interface (HMI) based on gaze tracking to control the direction of movement of a robot manipulator. The interface allows the user to open and to close a robotic hand attached to the end effector of the manipulator using intentional blinking. The initial optimisation of the Human Machine Interface based on gaze tracking was made through the results obtained after several experiments based on the execution of a reaching and grasping task, performed by one single subject. We evaluated the performance of the system according to the percentage of hits obtained in the task and the execution time. After the first optimisation was complete we tested the interface in 9 different subjects. The setup of the experiments used in the initial optimisation was explained in Section 6.3.3. The test experiment consisted in placing an object in 8 different positions where the subject tried to grasp it.

Two different control strategies based on the location of the gaze point were proposed: a Four directional Control and a Vector directional control.

The signal from the gaze tracker had to be pre-filtered in order to remove the noise generated by the non intentional saccadic movement of the eye and to smooth the signal, we proposed and tested several filters according to the control strategy. For the Four Directional Control we tested: moving average filters with windows of 150 ms, 300 ms and 500 ms; and weighted moving average filters with windows of 150 ms and 300 ms. The suitability of the filters was evaluated in the execution of a reaching and grasping task during the initial optimisation of the HMI. The highest percentage of hits and the second best execution time record were reached when using a moving average filter with a 300 ms windows. According to this, a moving average filter with a 300 ms window was implemented in the Four Directional Control. For the Vector Directional Control we tested a Kalman filter with two different values for the process noise covariance Q=0.001 and Q=0.0001. The performance of the filters was also evaluated in the execution of reaching and grasping tasks during the initial

optimisation of the HMI. A Kalman filter with process noise covariance of Q=0.001 showed to be the most suitable to be implemented in the Vector Directional Control, obtaining both the highest percentage of hits and the shortest execution time.

The control strategies also allowed to control the velocity of the manipulator under three different velocity profiles: a Single Velocity profile, a Dual Velocity profile and a Cubic Velocity profile. During the initial optimisation, a subject performed several reaching and grasping tasks in order to test the velocity profiles under the two control strategies. The subject had the highest percentage of hits when using the Four Directional Control with Single Velocity profile, and carried out the task faster when using the Vector Directional Control with Cubic Velocity profile. On the other hand, Vector Directional Control with Dual Velocity profile had the poorest performance, obtaining the lowest percentage of hits; while Vector Directional Control with Dual Velocity profile registered the longest execution time. Regarding to the performance of the subject when testing the control strategies with different velocity profiles, the results obtained in the initial optimisation differed from the results obtained when other subjects performed the same tasks.

The HMI based on gaze tracking was tested by nine different subjects in the execution of a reaching and grasping task using the two proposed control strategies under the three velocity profiles, respectively. The results of the experiments were analysed in Section 6.5.2. A one way analysis of variance was significant for the percentage of hits, but not significant for the execution time. According to the mean and median plots of the percentage of hits, the Four Directional Control with Dual Velocity profile performed better. But the Vector Directional Control with Dual and Cubic Velocity profile helped the user to execute the task faster. On the other hand, Vector Directional Control with Cubic Velocity profile had the poorest performance, while Vector Directional Control with Single Velocity profile registered the longest execution time.

Chapter 7

General conclusions and future work

In this thesis we presented the development of a human machine interface (HMI) based on gaze tracking that was used to control a 7 degrees of freedom (DOF) WAM arm manipulator in several reaching and grasping tasks. We analysed the performance of the manipulator in order to determine its suitability for the HMI implementation. We proposed two alternative control schemes for the manipulator and we compared the performance of the WAM arm under three different control strategies. The conclusions related to this work are presented in Section 7.1. The results obtained during the development of this thesis encouraged us to suggest some implementations for future work that are presented in Section 7.2.

7.1 Conclusions

.

A robot manipulator can be considered as a good representation of the human arm and might be used as a platform to develop control strategies that can be used in prototypes of future artificial prosthetics. Considering this, in Chapter 2 we analysed the performance of the 7 DOF Wam arm from Barrett Technologies in

order to determine its accuracy in the execution of specific trajectories. The system was configured to work with joint Proportional Derivative (PD) control and gravity compensation. We used three performance measurements: the joint position errors during the execution of the trajectory, a relative percentage error (RPE) for the final position of the joint and the actual joint trajectories of the manipulator. The joint position errors obtained during the execution of the joint trajectory helped to analyse the behaviour of the joint controllers in the execution of the joint rotation. The RPE was used to measure how accurate the joint motion was in reaching the desired end position. The actual joint trajectory showed the quality of the manipulator's motion during the task giving that an irregular trajectory was equivalent to a shaky motion of the joint.

After analysing the system in the execution of joint rotations of 0.5 rad at a velocity of 0.083 rad/sec, we found that during the execution of the task the joint position errors were less than 0.02 rad; the RPE varied among joints, from less than 1% to up to an approximate of 7%. The motion of the manipulator was slightly shaky, which was appreciable in the joint trajectory plots where it could be noticed that the curves were not completely smooth.

Since the manipulator was intended to be used in future applications of neuroprosthetics, we proposed to use Proportional Integral and Derivative (PID) control in the WAM arm with the aim of reducing the RPE to less than 1% in all the joints of the manipulator. Our second objective was to obtain smoother joint trajectories during the execution of tasks. However, tuning the parameters of the PID controllers was not a straightforward process since the system was non linear and due to the number of the manipulator's degrees of freedom. In order to have a better understanding of the dynamics of the manipulator we worked on a mathematical model of the system that included friction. The mathematical model was implemented in MATLAB/Simmechanics. Mathematical modelling is fundamental in the designing process depending of the objectives of the implementation,

the constraints of the task to be executed, and the desired robot's performance. The difficulty in obtaining these models varies according to the complexity of the kinematics of the mechanical structure and the number of degrees of freedom (DOF) (Khalil and Dombre, 2002). According to the Barrett WAM arm datasheet the system has zero backlash and near-zero friction. The initial design of the model did not include friction, but after running some simulations and comparing the behaviour of the model with respect to the real system we concluded that this model was not accurate enough. We proceeded to identify the friction phenomena in the joints of the manipulator with the aim of finding a mathematical representation that could be added to the WAM arm model. The model of the WAM arm included: seven joint PD/PID controllers configured as in the real system, seven modules used to emulate the joint reference trajectory as computed in the real system, and the mathematical models of the friction manifested in all the joints of the manipulator. The accuracy of the model was evaluated according to the results obtained in the execution of several joint rotations performed by both the model and the real system. We used the position errors registered in the experiments to compare the performance of the model and the WAM arm. Considering that in practice the real system is affected by many external disturbances such as: non linearities due to the joint motors, the joint actuators, and the inertia of the system; the results of the experiments showed that the response of the model was a fair representation of the performance of the real system. Since the model was implemented in SimMechanics/Simulink, we were able to implement a platform for simulations that was useful in future applications. For example, the accuracy of the model allowed us to implement an iterative method for estimating the parameters of the PID controllers of the seven joints of the manipulator described in Chapter 5.

The friction identification was achieved by performing motion experiments where

the joints of the manipulator were moved at several constant velocities. During the experiments, the joint torques and the velocities were registered and they were used later to construct friction velocity maps. The friction velocity maps helped to find the parameters of the classic friction model by fitting the data using a least squares minimisation method. The classic friction model is a good representation of friction when the system works from medium to medium-high velocities, which was the case in our objective applications. The classic friction model suited satisfactorily the friction phenomena in the joints of the WAM arm. Considering this, we implemented a feedforward scheme to be added to the joint PD controllers of the manipulator as an alternative to improve the performance of the manipulator. The control scheme for friction compensation helped to reduce the position error in six out of the seven joints; the RPE was reduced in three out of the seven joints of the manipulator; and all the joint trajectories were smoother than those observed when using the joint PD control scheme without the friction compensation module. The friction identification process and the implementation of the feedforward friction compensation are described in Chapter 4.

The implementation of joint PID control in the WAM arm involved a tuning process based on an Iterative Feedback Tuning (IFT) technique for linear systems proposed by Hjalmarsson. The technique had been implemented with positive results in simulated non linear systems (Hjalmarsson, 1998, 2002). In Chapter 5 we showed a different implementation of the Iterative Feedback Tuning in a real non linear system using both real and simulated data. The IFT technique used experimental data to find the gradient direction of the estimation of the controller parameters in each iteration. However, since some of the experiments could not be executed in the WAM arm, we used the model of the system described in Chapter 3 to perform part of the iterative process. We also used two different approaches to approximate the gradient: a Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm and the

Newton-Gauss method. The process of estimation of the controller parameters was completed after 15 iterations, which consisted in the execution of 30 experiments, for all the joints of the manipulator. After tuning the joint PID controllers we analysed the performance of the WAM arm in the execution of a motion task. The PID control helped to reduce the position error in all the joints of the manipulator to values very close to zero; the RPE was reduced to less than 0.007%; and the joint trajectories were smoother than those obtained when using the joint PD control scheme without the friction compensation module.

Chapter 5 also includes a discussion section where we compared the performance of the WAM arm under the three different control strategies described in this thesis: PD control with gravity compensation, PD control with gravity compensation and feedforward friction compensation, and PID control. According to the results, out of the two alternative control strategies that were tested on the WAM arm, PID control improved the performance of the manipulator by reducing the joint position errors during the rotation task, obtaining the lowest RPE and executing the smoothest trajectories.

In Chapter 6 we described a Human Machine Interface based on gaze tracking used to control the position a robot manipulator in a 2D space. The system used the gaze point to give directions of movement to the robot arm and used intentional blinking to open and close a robotic hand attached to the end effector of the manipulator, in order to grasp objects.

The gaze tracking system used in the project consisted in a head mounted device that measures the gaze point of the user with respect to a scene view, given by a camera positioned in the front of the head. The manipulator was a 7 DOF WAM arm form Barrett technologies. We filtered the gaze point signal using different filtering techniques in order to feed two different control strategies that worked

in real time: a Four directional Control and a Vector Directional control. The Four Directional Control was used to move the robot manipulator using vertical and horizontal steps. The Vector Directional Control considered the gaze point as a vector, which angle of direction was translated into the manipulator's space, allowing the robot arm to move with vertical, horizontal and diagonal steps.

The two control strategies also included an implementation of velocity control. We proposed and tested three different velocity profiles in the execution of reaching and grasping tasks: A Single Velocity profile, a Dual Velocity profile and a Cubic Velocity profile.

We performed an initial optimisation of the system that helped us to find the suitability of a gaze point filtering technique and to test the coherence of the velocity profiles. The experiments used in the optimisation process consisted in placing an object on a table in 8 different locations where a subject tried to grasp it. The manipulator was placed in an initial position at the beginning of the reaching and grasping task and came back to the initial position after the user closed the grasp three times using intentional blinking. We measured the percentage of hits and the execution time in each experiment. During the optimisation stage all the experiments were conducted using a single velocity profile. Based on the results of the optimisation process, the technique implemented in the system to filter the gaze point for the Four Directional Control was a moving average filter with a 300 ms window. The filtering technique implemented in the system in the Vector Directional Control was a Kalman filter with process noise covariance of Q=0.001. The optimisation process of the system is described in Section 6.3.3 and Section 6.4.3. After the initial optimisation we tested the system in 9 different subjects in reaching and grasping tasks using the two control strategies working under the three velocity profiles, respectively. All the subjects were able to complete the tasks using both of the proposed control strategies, even when the subjects were completely naive to the system at the beginning of the

experiments. The proficiency in using each velocity profile varied according to the user as described in Section 6.5.2. The strategy with the highest average rate of hits was the Four Directional Control with Dual Velocity profile, with 71% of hits.

The results of the experiments allowed us to consider the use of gaze tracking as an alternative to control robotic prosthetics intended to aid paralysed people. Our system was able to perform reaching and grasping tasks similar to those accomplished with motor neuroprosthetics implemented in monkeys (Velliste et al., 2008) and in people with tetraplegia (Hochberg et al., 2012), without the need to undergo surgery or having several training sessions. It took an average of 15 minutes for the eye tracker to be fully operational in the user, this time included the introduction of the devices to the user and the explanation of how the system worked. Most of the subjects took an extra 5 minutes time to practice before starting the first experiment. The subjects did not have problems in understanding the relation of their visual space with the gaze tracker visual space and they were able to successfully rely in mental plots of the surrounding space and the location of the object while moving the eyes to control the manipulator. At the beginning of the experiments the users slightly struggle to find the correct position of the eyes that allowed to pause the manipulator's control using the *Stop* region but they were able to master it during the practice time or within the execution of the first experiment, respectively. All the experiments in the subjects were made in one session that lasted an approximate of 2 hours time. At the end of the experiments most of the subjects grow tired of making fixations for this period of time. Despite of the later, the HMI based on gaze tracking described in this work exhibited a good performance in a real application and it was successfully tested in different subjects. The signal processing behind the design of the control strategies was computationally cheap and easy to implement.

The results presented in this work showed that it is suitable to use a HMI based on gaze tracking in the execution of reaching and grasping tasks with a high percentage of success. However, further experiments have to be conducted in order to establish which combination of control strategy and velocity profile is the best out of the ones proposed.

7.2 Future work

The friction phenomena affecting the WAM arm did not compromise the stability of the system during the implementation of the IFT technique described in Chapter 5. However, a better response can be expected if we completely compensate for the non linearities of the system.

The aim of the future work proposed in this thesis is to find a method able to work in parallel with the IFT tuning that can help to compensate for the friction affecting the system without a previous friction identification process.

After some consideration we propose the implementation of an observer.

Observers are helpful when estimating unknown states of a dynamic system. They consist in the implementation of the mathematical model of the system and its addition to the operational process by working in parallel with the real system. When the model is accurate with respect to the real system the behaviour of both is expected to be exactly the same. However, this expectation is not likely to occur due to all the external disturbances that affect a process in practice.

In order to design an observer a measurable variable in the real system needs to be found first. This variable will be used for the observer as a correction factor in the estimation of the future states. Figure 7.1 shows the general scheme of a state estimator. In the block diagram the measurement variable is the output of the process y, the state variable is x and the control signal is u. The observer estimates the output variable y_e through the mathematical model of the system. The difference

between y and y_e is used then to update the estimate of a future state (Ogata, 2010).

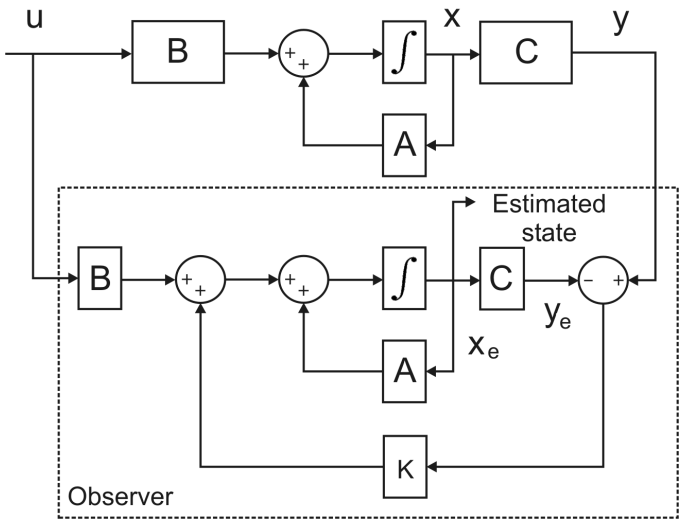


Figure 7.1: General scheme of a state estimator. The variable x is the state of the system driven by signal u and C represents the sensor for the output y. The values of x_e and y_e are the estimated state and output, respectively.

In regards to the HMI based on gaze tracking presented in Chapter 6, we showed that the system exhibited good performance when it was tested by 9 different subjects in reaching and grasping tasks. However, further experiments need to be conducted and several analyses need to be considered in order to improve the performance of the system.

First, we suggest to analyse the robot manipulator's trajectory when it approaches the object, in order to determine the suitability to implement a strategy to predict the grasping point. During the experiments the subjects had to rely in mental plots of the surrounding space and the location of the object while moving the eyes to control the position of the manipulator. If a prediction technique is used to find the location of the object based on the fixations of the user while controlling the position of the robot arm, we expect an improvement in the hit rates and a reduction of the execution time.

We also need to find the best combination of control strategy and velocity profile

so further experiments with subjects are needed. In order to accomplish that, the subjects will need to undergo an initial training in the use of all the strategies and velocity profiles proposed in this work. After the training process, the subjects would perform a series of grasping and reaching tasks using various combination of control strategy and velocity profile presented randomly, in several trials. The data obtained from the experiments will be useful to find the best performance considering each strategy based on the percentage of hits and the execution time. The design of the HMI should consider to add control signals that can be used to automatically start the control of the manipulator such as those used to stop the control (using intentional blinking three times), since so far the control starts only with the help of a supervisor that activates the control for safety reasons.

The current implementation of the HMI based on gaze tracking allows to control the robot manipulator in a two dimensional space. However the future implementation of three dimensional motion is completely feasible, we propose several approaches that can be considered:

- Binary coding using blinking to switch between two planar spaces XY and XZ.
- The use of image signal processing to find the depth considering the gaze point and the scene image given by the gaze tracking system.
- The use of eye tracking techniques to find the three-dimensional angle of rotation of the eye to compute the depth.

Appendices

Appendix A

Denavit-Hartenberg Convention

The Denavit-Hartenberg convention is a systematic method that helps to establish a body attached frame to each link of an articulated chain (Verma and Gor, 2010). The Denavit-Hartenberg convention is widely used to find the relationship between the joints of the robot manipulator and the position of the end-effector (Spong et al., 2006) using four specific transformations:

1. Rotation about z_{i-1} axis by an angle θ_i.
2. Translation along z_{i-1} axis by distance d_i.
3. Translation along x_{i-1} axis by distance a_i.
4. Rotation about x_{i-1} axis by an angle α_i.

where θ_i, a_i, d_i and α_i are the Denavit-Hartenberg parameters of link i. Defining the parameters:

- a_i: distance measured along x_i axis from the point of intersection of x_i axis with z_{i-1} to the origin of frame i.
- α_i: angle between z_{i-1} and z_i axes measured about x_i axis in the right hand sense.

- d_i: distance measured along z_{i-1} axis from the origin of frame $(i-1)$ to the intersection of x_i-axis with z_{i-1} axis.

- θ_i: angle between x_{i-1} and x_i axes measured about the z_{i-1} axis in the right-hand sense.

the assignation of the coordinate systems for each link that builds the articulated chains can be established following the algorithm mentioned by Barrientos et al. (1997).

- Assign a number to each link of the chain from 1 to n, using 0 for the fixed base of the robot.

- Assign a number to each joint starting with 1 (corresponding to the first degree o freedom) and ending with n.

- Identify the rotation axis for revolute joints, and the displacement axis for prismatic joints.

- For the i joint from 0 to $(n-1)$, place the z_i axis over the $(i+1)$ joint axis.

- Place the origin of the base of the system $x_0y_0z_0$ over the z_0 axis.

- For the i joint from 1 to $(n-1)$, place the system $x_iy_iz_i$ in the intersection of the z_i axis with the common normal line of z_{i-1} and z_i. If the axes are parallel then the system must be placed in the $(i+1)$ joint.

- Place x_i over the common normal line of z_{i-1} and z_i.

Appendix B

Newton-Euler Recursive Algorithm

The Newton-Euler recursive algorithm as mentioned by Siciliano et al. (2009) refers all vectors to the current frame on a generic link i, using the following parameters:

- m_i mass of the link.
- $\bar{I}_i$ inertia tensor of link i.
- r_{i-1,C_i} vector from origin of frame $(i-1)$ to centre of mass C_i of link i.
- r_{i,C_i} vector from origin of frame (i) to centre of mass C_i of link i.
- $r_{i-1,i}$ vector from origin of frame $(i-1)$ to origin of frame i.
- $\dot{p}_{C_i}$ linear velocity of centre of mass C_i of link i.
- $\dot{p}_i$ linear velocity of origin of frame i.
- ω_i angular velocity of link i.
- $\ddot{p}_{C_i}$ linear acceleration of centre of mass C_i of link i.
- $\ddot{p}_i$ linear acceleration of origin of frame i.

- $\dot{\omega}_i$ angular acceleration of link i.
- g_0 gravity acceleration.
- f_i force exerted by link $(i-1)$ on link i.
- $-f_{i+1}$ force exerted by link $(i+1)$ on link i.
- μ_i force exerted by link $(i-1)$ on link i with respect to origin of frame $(i-1)$.
- $-\mu_{i+1}$ force exerted by link $(i+1)$ on link i with respect to origin of frame i.

the recursive algorithm can be listed as

$$
\begin{aligned}
\omega_i^i &= (R_i^{i-1})^T \omega_{i-1}^{i-1} \quad \text{For a } \textit{prismatic} \text{ joint} \\
&= (R_i^{i-1})^T (\omega_{i-1}^{i-1} + \dot{\theta}_i z_0) \quad \text{For a } \textit{revolute} \text{ joint} \\
\dot{\omega}_i^i &= (R_i^{i-1})^T \dot{\omega}_{i-1}^{i-1} \quad \text{For a } \textit{prismatic} \text{ joint} \\
&= (R_i^{i-1})^T (\dot{\omega}_{i-1}^{i-1} + \ddot{\theta}_i z_0) + \dot{\theta}_i \omega_{i-1}^{i-1} \times z_0 \quad \text{For a } \textit{revolute} \text{ joint} \\
\ddot{p}_i^i &= (R_i^{i-1})^T (\ddot{p}_{i-1}^{i-1} + \ddot{d}_i z_0) + 2\dot{d}_i \omega_i^i \times (R_i^{i-1})^T z_0 \\
&\quad + \dot{\omega}_i^i \times r_{i-1,i}^i + \omega_i^i \times (\omega_i^i \times r_{i-1,i}^i) \quad \text{For a } \textit{prismatic} \text{ joint} \\
&= (R_i^{i-1})^T \ddot{p}_{i-1}^{i-1} + \dot{\omega}_i^i \times r_{i-1,i}^i + \omega_i^i \times (\omega_i^i \times r_{i-1,i}^i) \quad \text{For a } \textit{revolute} \text{ joint} \\
\ddot{p}_{C_i}^i &= \ddot{p}_i^i + \dot{\omega}_i^i \times r_{i,C_i}^i + \omega_i^i \times (\omega_i^i \times r_{i,C_i}^i) \\
f_i^i &= R_{i+1}^i f_{i+1}^{i+1} + m_i \ddot{p}_{C_i}^i \\
\mu_i^i &= -f_i^i \times (r_{i-1,i}^i + r_{i,C_i}^i) + R_{i+1}^i \mu_{i+1}^{i+1} + R_{i+1}^i f_{i+1}^{i+1} \times r_{i,C_i}^i + \bar{I}_i^i \dot{\omega}_i^i + \omega_i^i \times (\bar{I}_i^i \omega_i^i) \\
\tau_i &= (f_i^i)^T (R_i^{i-1})^T z_0 \quad \text{For a } \textit{prismatic} \text{ joint} \\
&= (\mu_i^i)^T (R_i^{i-1})^T z_0 \quad \text{For a } \textit{revolute} \text{ joint}
\end{aligned}
$$

where z_0 is a unit vector $z_0 = [0\ 0\ 0]^T$; $\omega_0^0, \ddot{p}_0^0 - g_0^0$ and $\dot{\omega}_0^0$ are the initial conditions; f_{n+1}^{n+1} and μ_{n+1}^{n+1} are the terminal conditions; and quantities $\bar{I}_i$ and r_{i,C_i}^i are constant.

Bibliography

Analog Devices Inc. `http://www.analog.com`, 2013.

(ASL) Applied Science Laboratories. Mobile eye SDK manual version 1.01. *Applied Science Group Inc.*, 2008.

K. Arai and R. Mardiyanto. Eye-based human computer interaction allowing phoning, reading e-book/e-comic/e-learning, internet browsing, and tv information extraction. *International Journal of Advanced Computer Science and Applications*, 12:26–32, 2011.

B. Armstrong, O. Khatib, and J. Burdick. The explicit dynamic model and inertial parameters of the PUMA 560 arm. *Proccedings of the 1986 IEEE International Conference on Robotics and Automation*, pages 510–518, 1986.

B. Armstrong-Helouvry, P. Dupont, and C. Canudas de Wit. A survey of models, analysis tools and compensation methods for the control of machines with friction. *Automatica*, 30:1083–1138, 1994.

H. Asada and J-J.E. Slotine. *Robot Analysis and Control*. John Wiley and Sons, 1986.

R. Barea, L. Boquete, M. Mazo, and E. Lopez. System for assisted mobility using eye movements based on electrooculography. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10:209–2018, 2002.

R. Barea, L. Boquete, S. Ortega, E. Lopez, and J. M. Rodriguez-Ascariz. EOG-based

eye movements codification for human computer interaction. *Expert Systems with Applications*, 39:2677–2683, 2012.

Barrett Technology Inc. *WAM Arm User's Manual*. 2008a.

Barrett Technology Inc. *WAM Arm Datasheet*. 2008b.

A. Barrientos, L.F Penin, and C. Balaguer. *Fundamentos de Robotica*. McGraw-Hill, 1997.

P.R. Belanger. *Control Engineering: A Modern Approach*. Oxford University Press, 1995.

N. A. Bompos, P. K. Artemiadis, A. S. Oikonomopoulos, and K. J. Kyriakopoulos. Modeling, full identification and control of the MITSUBISHI PA-10 robot arm. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 1–6, 2007.

D. Borghetti, A. Bruni, M. Fabbrini, L. Murri, and F. Sartucci. A low-cost interface for control of computer functions by means of eye movements. *Computers in Biology and Medicine*, 37:1765–1770, 2007.

B. Borsotto, E.and Beauvois D. Godoy, and E. Devaud. An identification method for static and Coulomb friction coefficients. *International Journal of Control, Automation and Systems*, 7:305–310, 2009.

BrainGate. Braingate neural interface system. `http://www.braingate.com/`.

A. Bulling, J. A. Ward, H. Gellersen, and G. Troster. Eye movement analysis for activity recognition using electrooculography. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

T. Bushmann, S. Lohmeier, H. Ulbrich, and F. Pfeiffer. Dynamics simulation for a biped robot: modeling and experimental verification. *IEEE Proccedings of the*

2006 International Conference on Robotics and Automation., pages 2673–2678, 2006.

G. Calafiore, Indri M., and B. Bona. Robot dynamic calibration: optimal excitation trajectories and experimental parameter estimation. *Journal of robotic systems*, 18:55–68, 2001.

A. Calvalho-Bittencourt and S. Gunnarsson. Static friction in a robot joint modeling and identication of load and temperature effects. *Journal of Dynamic Systems, Measurement, and Control*, 134:10, 2009.

C. Canudas de Wit and P. Lischinsky. Adaptive friction compensation with partially known dynamic friction model. *International Journal of Adaptive Control and Signal Processing*, 11:65–80, 1997.

C. Canudas de Wit, H. Olsson, K.J. Astrom, and P. Lishinsky. A new model for control of systems with friction. *IEEE Transactions On Automatic Control*, 40: 419–425, 1995.

P.I. Corke. A robotics toolbox for MATLAB. In *IEEE Robotics and Automation Magazine*, 1996.

F. De Bruyne, B. D. O. Anderson, M. Gevers, and N. Linard. Iterative controller optimization for nonlinear systems. *Proceedings of the 36th Conference on Decision and Control*, pages 3749–3754, 1997.

A. De Satis and D. Iacoviello. Robust real time eye tracking for computer interface for disabled people. *Computer Methods and Programs in Biomedicine*, 96:1–11, 2009.

L. Y. Deng, C-L. Hsu, T-C. Lin, J-S. Tuan, and S-M. Chang. EOG-based human computer interface system development. *Expert Systems with Applications*, 37: 3337–3343, 2010.

R. L. Drake, A. W. Vogl, and A. W. M. Mitchell. *Gray's Anatomy for Students*. Churchill Livingstone Elsevier, second edition, 2010.

F. Golnaraghi and B.C. Kuo. *Automatic Control Systems*. John Wiley & Sons, 2010.

S. Gunnarsson, V. Collignon, and O. Rousseaux. Tuning of decoupling controller for a 2x2 system using iterative feedback tuning. *Control Engineering Practice*, 11:1035, 2003.

H. B. Guo and H. R. Li. Dynamic analysis and simulation of a six degree of freedom stewart platform manipulator. *IMechE, J. Mechanical Engineering Science*, 220: 61–72, 2005.

K. Hamamoto, T. Fukuda, and T. Sugie. Iterative feedback tuning of controllers for a two-mass spring system with friction. *Control Engineering Practice*, 11: 10611068, 2003.

R. H. A. Hensen, G. Z. Angelis, M. J. G. van de Molengraft, A. G. de Jager, and J. J. Kok. Grey-box modeling of friction: an experimental case-study. *European Journal of Control*, 6, 2000.

R. Hildebrand, A. Lecchini, G. Solari, and M. Gevers. Asymptotic accuracy of iterative feedback tuning. *IEEE Transactions on Automatic Control*, 50:1182, 2005.

H. Hjalmarsson. Control of nonlinear systems using iterative feedback tuning. *Proceedings of the American Control Conference*, 50:2083, 1998.

H. Hjalmarsson. Iterative feedback tuning: an overview. *International Journal of adaptive control and signal processing*, 16:373–395, 2002.

H. Hjalmarsson, S. Gunnarsson, and M. Gevers. A convergent iterative restricted complexity control design scheme. *Proceedings of the 33th Conference on Decision and Control*, 16:1735, 1994.

L. R. Hochberg, M. D. Serruya, G. M. Friehs, J. A. Mukand, M. Saleh, A. H. Caplan, A. Branner, D. Chen, R. D. Penn, and J. P. Donoghue. Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, 442:164–171, 2006.

L. R. Hochberg, D. Bacher, B. Jarosiewicz, N. Y. Masse, J. D. Simeral, J. Vogel, S. Haddadin, J. Liu, S. S. Cash, P. van der Smagt, and J. P. Donoghue. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485:372–375, 2012.

J. Hua, Y. Cui, W. Ding, H. Li, Y. Wang, and N. Xi. Modeling and control of wheeled mobile robot in constrained environment based on hybrid control framework. *IEEE Proccedings of the 2009 International Conference on Robotics and Biomimetics.*, pages 75–79, 2009.

T. E. Hutchinson, K. P. White, W. N. Martin, K. C. Reichert, and Frey L. A. Human computer interaction using eye gaze input. *IEEE Transactions on Systems, Man, and Cybernetics*, 19:1527–1534, 1989.

M. Indri. Control of manipulators subject to unknown friction. In *45th IEEE Conference on Decision and Control*, 2006.

A. Izadbakhsh. Closed-form dynamic model of the PUMA 560 robot arm. *Proccedings of the 4th International Conference on Autonomous Robots and Agents*, pages 675–680, 2009.

R. J. K. Jacob. The use of eye movements in human computer interaction techniques: What you look is what you get. *ACM Transactions on Information Systems*, 9: 152–169, 1991.

C.R Johnson and R.D Lorenz. Experimental identification of friction and its compensation in precise, position controlled mechanisms. *IEEE Transactions on Industry and Applications*, pages 1392–1398, 1992.

M. A. Johnson and M.H. Moradi. *PID control: New Identification and Design Methods*. Springer, 2005.

R. Kelly, V. Santibanez, and A. Loria. *Control of Robot Manipulators in Joint Space*. Springer, 2005.

C. W. Kennedy and J. P. Desai. Model-based control of the MITSUBISHI PA-10 robot arm: application to robot-assisted surgery. *Proccedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 2523–2528, 2004.

M. R. Kermani, R. V. Patel, and M. Moallem. Friction identification in robotic manipulators: Case studies. *IEEE Conference on Control Applications*, 2005.

M. R. Kermani, R. V. Patel, and M. Moallem. Friction identification and compensation in robotic manipulators. *IEEE Transactions on instrumentation and measurement*, 56, 2007.

W. Khalil and E. Dombre. *Modeling Identification and Control of Robots*. Kogan Page Science, 2002.

S-P. Kim, J. D. Simeral, L. R. Hochberg, J. P. Donoghue, G. M. Friehs, and M. J. Black. Point-and-click cursor control with an intracortical neural interface system by humans with tetraplegia. *IEEE Transactions on neural systems and rehabilitation engineering*, 19:193–203, 2011.

D. Kostic, B. de Jager, M. Steinbuch, and R. Hensen. Modeling and identification for high performance robot control: An rrr-robotic arm case study. In *IEEE Transactions on Control System Technology*, 2004.

A. Krolak and P. Strumillo. Eye blink detection system for human computer interaction. *Universal Access in the Information Society*, 11:409–419, 2012.

J. R. LaCourse and F. C. Jr. Hludik. An eye movement communication-control

system for the disabled. *IEEE Transactions on Biomedical Engineering*, 37:1215–1220, 1990.

H. Y. K. Lau and L. C. C. Wai. A jacobian-based redundant control strategy for the 7-DOF WAM. *Seventh International Conference on Control, Automation, Robotics and Vision*, pages 1060–1065, 2002.

F. L. Lewis, D. M. Dawson, and C. T. Abdallah. *Robot Manipulator Control: Theory and Practice*. Marcel Dekker, 2004.

C. Lightcap and S. Banks. Dynamic identification of a MITSUBISHI PA10-6CE robot using motion capture. *Proccedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3860–3865, 2007.

MathWorks. *MATLAB/SimMechanics Documentation*. 2011.

J.L Meriam and L.G Kraige. *Engineering Mechanics: Dynamics*. John Wiley and Sons, sixth edition, 2007.

C. P. Neuman and J. J. Murray. The complete dynamic model and customized algorithms of the PUMA robot. *IEEE Transactions on systems, man, and cybernetics*, 17:635–644, 1987.

M. A. L Nicolelis. Action from thoughts. *Nature*, 409:403–407, 2001.

B. Noureddin, P. D. Lawrence, and C. F. Man. A non-contact device for tracking gaze in a human computer interface. *Computer Vision and Image Understanding*, 98:52–82, 2005.

K. Ogata. *Modern Control Engineering*. Pearson, fifth edition, 2010.

M. M. Olsen and H. G. Petersen. A new method for estimating parameters of a dynamic robot model. *IEEE Transactions on Robotics ans Automation*, 17, 2001.

H. Olsson, K.J Astrom, C. Canudas de Wit, M. Gafvert, and P. Lischinsky. Friction models and friction compensation. *European Journal of Control*, pages 176–195, 1998.

D. E. Orin. Application of robotics to prosthetic control. *Annals of Biomedical Engineering.*, 8:305–316, 1980.

G. Patil Parag and D. A Turner. The development of brain-machine interface neuroprosthetic devices. *The Journal of the American Society for Experimental NeuroTherapeutics.*, 5:137–146, 2008.

Python. `http://www.python.org`, 2013.

M. B. Radac, R. M. Precup, E. M. Petriu, S. Preitl, and C. A. Dragos. Convergent iterative feedback tuning of state feedback-controlled servo systems. *Informatics in Control Automation and Robotics*, page 99, 2011.

V. Rantanen, T. Vanhala, O. Tuisku, P-H. Niemenlehto, V. Vernho, J. Surakka, M. Juhola, and Lekkala J. A wearable, wireless gaze tracker with integrated selection command source for human-computer interaction. *IEEE Transactions on Information Technology in Biomedicine*, 15:795–801, 2011.

M. J. Reale, S. Canavan, L. Yin, K. Hu, and T. Hung. A multi-gesture interaction system using a 3-D iris disk model for gaze estimation and an active appearance model for 3-D hand pointing. *IEEE Transactions in Multimedia*, 13:474–486, 2011.

A. B. Schwartz. Cortical neural prosthetics. *Annual Review of Neuroscience*, 27: 487–507, 2004.

A. B. Schwartz, D. M. Taylor, and S. I. Helms Tillery. Extraction algorithms for cortical control of arm prosthetics. *Current Opinion in Neurobiology*, 11:701–707, 2001.

A. B. Schwartz, X. T. Cui, D. J. Weber, and D. W. Moran. Brain-controlled interfaces: movement restoration with neural prosthetics. *Neuron*, 52:205–220, 2006.

D.E Seborg, T.F. Edgar, and D.A. Mellichamp. *Process Dynamics and Control*. Wiley, 2004.

A Sesin, M. Adjouadi, M. Ayala, M. Cabrerizo, and A. Barreto. Eyeing a real time human computer interface to assist those with motor disabilities. *IEEE Potentials*, 27:19–25, 2008a.

A Sesin, M. Adjouadi, M. Cabrerizo, M. Ayala, and A. Barreto. Adaptive eye-gaze tracking using neural-network-based user profiles to assist people with motor disability. *Journal of Rehabilitation Research and Development*, 45:801–818, 2008b.

B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer, 2009.

J. D. Simeral, S-P. Kim, M. J. Black, J. P. Donoghue, and L. R. Hochberg. Neural control of cursor trajectory and click by a human with tetraplegia 1000 days after implant of an intracortical microelectrode array. *Journal of neural engineering*, 8: 1–24, 2009.

J. Sjoberg and M. Agarwal. Model-free repetitive control design for nonlinear system. *Proceedings of the 35th Conference on Decision and Control*, pages 2824–2829, 1996.

J. Sjorberg, F. De Bruyne, M. Agarwal, B. D. O. Anderson, M. Gevers, F. J. Kraus, and N. Linard. Iterative controller optimization for nonlinear systems. *Control Engineering Practice*, 11:1079, 2003.

R. S. Snell and M. A. Lemp. *Clinical Antomy of the Eye*. Blackwell Science, second edition, 1998.

C. Sousa, R. Cortesao, and P. Queiros. Compliant co-manipulation control for medical robotics. *Proceedings of the 2nd conference on Human System Interactions*, pages 262–268, 2009.

O. Spakov. Comparison of eye movement filters used in HCI. *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 281–284, 2012.

M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling Control*. John Wiley and Sons, 2006.

Y. Su and C. Zheng. A simple nonlinear proportional-derivative controller for friction compensation. *Proceedings of the 2009 IEEE International Conference on Robotics and Biomimetics*, 2009.

A. Tall, A. Alapetite, J. San Agustin, H. H. T. Skovsgaard, J. P. Hansen, D. W. Hansen, and E. Mollenbach. Gaze controlled driving. *CHI Conference on Human Factors in Computing Systems*, pages 4387–4392, 2009.

D. M. Taylor, S. I. Helms Tillery, and A. B. Schwartz. Direct cortical control of 3D neuroprosthetic devices. *Science*, 296:1829–1832, 2002.

D. M. Taylor, S. I. Helms Tillery, and A. B. Schwartz. Information conveyed through brain-control: cursor versus robot. *IEEE Transactions on neural systems and rehabilitation engineering*, 11:195–199, 2003.

J. J. Tecce, J. Gips, C. P. Olivieri, L. J. Pok, and M. R. Consiglio. Eye movement control of computer functions. *International Journal of Psychophysiology*, 29:319–325, 1998.

M. J. Tovee. *An Introduction to the Visual System*. Cambridge University Press, 1996.

A. A. Transeth and K. Y. Pettersen. Developments in snake robot modeling and

locomotion. *IEEE 9th International Conference on Control, Automation, Robotics and Vision, 2006. ICARCV '06.*, pages 1–8, 2006.

A. A. Transeth, , L.C. Glocker, K. Y. Pettersen, and P. Liljeback. Developments in snake robot modeling and locomotion. *IEEE Transactions on robotics.*, 24:88–104, 2008.

J. Varona, C. Manresa-Yee, and F. J. Perales. Hands-free vision-based interface for computer accessibility. *Journal of Rehabilitation Research and Development*, 31: 357–374, 2007.

M. Velliste, S. Perel, M. C. Spalding, A. S. Whitford, and A. B. Schwartz. Cortical control of a prosthetic arm for self-feeding. *Nature*, 453:1098–1101, 2008.

A. Verma and M. Gor. Forward kinematics analysis of 6-DOF arc welding robot. *International Journal of Engineering Science and Technology*, 2:4682–4686, 2010.

N. J. Wade and B. W. Tatler. *The Moving Tablet of the Eye.* Oxford University Press, 2005.

G. Welch and G. Bishop. The kalman filter. `http://www.cs.unc.edu/ welch/kalman/`, 2013.

J. Wolpaw and E. W. Wolpaw. *Brain Computer Interfaces: Principles and Practice.* Oxford University Press, 2012.

A. L. Yarbus. *Eye Movements and Vision.* Plenum Press, 1967.

J. S. Yeon, E. J. Kim, S-H. Lee, J. H. Park, and J-S. Hur. Development of inverse dynamic controller for industrial robots with HyRoHILS system. *International Conference on Control, Automation and Systems*, pages 1060–1065, 2005.

Q. Zhao, W. Xu, Y. Wang, and X. Shi. Using head poses to control a virtual robot walking in a virtual maze. *Optics Communications*, 295:84–91, 2012.