TÉCNICO LISBOA

A Position-based Approach for Force/ Moment Control of an Industrial Manipulator

Manuel Esteves de Mendonça

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisor: Prof. Jorge Manuel Mateus Martins

Examination Committee

Chairperson: Prof. Paulo Jorge Coelho Ramalho Oliveira
Supervisor: Prof. Jorge Manuel Mateus Martins
Member of the Committee: Prof. João Carlos Prata dos Reis

December 2019

Dedicated to my parents.

Acknowledgments

Approaching the end of my academic journey I could not fail to express my gratitude to everyone who have contributed to its success, especially those who were present during the entire process of my thesis development. Such accomplishments have never been possible without them.

First of all, I would like to thank my supervisor Prof. Jorge Martins for providing me the opportunity to work in the robotic manipulation field, for always being patient and willing to guide me throughout this work.

I would also like to thank to everyone in the ACCAII laboratory for the technical support and kindness during this work.

My special thanks are extended to my brother João Mendonça and my good friend Eng. Rafael Lucas for their unconditional availability and constant encouragement, especially given all the brainstorming sessions and endless late discussions that were incredibly fruitful.

I would like to thank Madelena Cid for all the reasons in the world and for constantly being there, mainly in times of most pressure.

Last but not the least, I would like to express my immense gratitude to all my family members and close friends, specially my parents, for all the love, wisdom and motivation not only regarding this thesis, but most importantly throughout my whole life.

Resumo

As estratégias de controlo de força fornecem uma forma eficaz de lidar com aplicações robóticas que envolvam interacção com o ambiente. Nesta tese é apresentada uma abordagem de controlo de força e momento em manipuladores robóticos baseada na posição, considerando interações de 6-GL não mínimos. Esta abordagem permite utilizar todas as medições disponíveis do sensor numa acção de controlo em todo o espaço dimensional sem recorrer a matrizes de selecção. As acções de controlo de força e momento foram projectadas para prevalecer sobre a acção de controlo de movimento, o que garante pequenos desvios na trajectória de força desejada. As variáveis posição/ orientação e força/ momento devem ser especificadas em todas as direcções do referencial da tarefa. A estratégia de controlo desenvolvida coaduna com a fase de impacto no material e com o movimento livre do manipulador. O ambiente é modelado como sendo complacente e de forma simplificada, pelo que se considera para ambas as acções de controlo um controlador PI. Projectou-se uma arquitectura de *hardware* e *software* onde o manipulador é controlado de forma remota sem acesso à linguagem de baixo nível do controlador industrial. Esta arquitectura contém um manipulador ABB® IRB140 dotado de um controlador industrial IRC5 e um sensor de força e momento de 6-GL JR3®. A comunicação entre o IRC5 e o computador externo é feita através de uma aplicação de controlo remoto entre o Simulink Real-Time™ e o RAPID segundo o protocolo de comunicação RS-232 a uma taxa de amostragem de $33Hz$. De forma a validar e comprovar a eficácia da abordagem, realizaram-se vários testes representativos de aplicações reais. As trajectórias utilizadas foram definidas por um planeador de trajectórias analítico.

Palavras-chave: Controlo de força, Controlador industrial padrão, Controlo de força/ momento baseado na posição, Planeamento de trajectória

Abstract

Force control strategies can provide an effective framework to deal with tasks that involve robotic-environment interaction. In this thesis, a position-based approach to force and moment control robotic manipulators is proposed while considering non-minimal 6-DOF interactions. Such approach allows a complete use of the available sensor measurements by operating the control action in a full-dimensional space without resorting to selection matrices. The force and moment control actions are designed to prevail the motion control loop, therefore ensuring limited deviations from the prescribed force trajectory. Position/ orientation and force/ moment must be specified along each direction of the task frame. A strategy to overcome the hurdle related to the non-contact to contact transition with the environment is considered, assuming a simplified compliant model of the environment and a PI controller law for both controllers' action. It relies on a hardware-software architecture for which the manipulator is remotely controlled while having no access to the lower layers of software running on the industrial controller. This architecture contains a ABB® IRB140 manipulator endowed by an IRC5 industrial controller and a JR3® 6-DOF force-moment sensor. The communication between the IRC5 and the external computer is achieved by a remote control application between Simulink Real-Time™ and RAPID through the RS-232 protocol with a sampling rate of $33Hz$. To validate and prove the effectiveness of the postulated approach, several experiments of representative applications were performed utilizing an analytical trajectory planner.

Keywords: Force Control, Standard Industrial Controller, Position-based Force/ Moment Control, Trajectory Planning

Contents

List of Tables

List of Figures

Nomenclature

Physics

$\boldsymbol{\mu}, M, m$ Moment.

$\boldsymbol{\omega}, \dot{\boldsymbol{\omega}}$ Angular velocity and acceleration vectors.

$\boldsymbol{\tau}$ Joint torques vector.

$\boldsymbol{F}, \boldsymbol{f}$ Force.

$\boldsymbol{h}$ 6×1 Force and moment vector.

$\boldsymbol{n}, \boldsymbol{s}, \boldsymbol{a}$ Unit vectors.

$\boldsymbol{p}, \dot{\boldsymbol{p}}, \ddot{\boldsymbol{p}}$ Position, velocity and acceleration vectors.

$\boldsymbol{q}, \dot{\boldsymbol{q}}$ Joint coordinates and velocities.

$\boldsymbol{R}$ 3×3 Rotation matrix.

$\boldsymbol{T}, \boldsymbol{A}$ Homogeneous 4×4 transformation matrix.

$\eta, \boldsymbol{\epsilon}$ Scalar and vector parts of a unit quaternion.

$\mathcal{Q}, \boldsymbol{e}$ Unit quaternion and quaternion error vectors.

d, v, a, α DH parameters.

x, y, z Cartesian components.

Mathematics

$\boldsymbol{E}$ Quaternion operator.

$\boldsymbol{I}$ 3×3 Identity matrix.

$\boldsymbol{J}$ Jacobian matrix.

$\boldsymbol{S}$ Skew-symmetric operator.

Δ Delta operator.

s, c Sine and cosine.

Trajectory

$\boldsymbol{c}, \rho$ Center and radius of the circle.

$\boldsymbol{r}, \vartheta, \dot{\vartheta}, \ddot{\vartheta}$ Axis, angle, velocity and acceleration of rotation.

$\boldsymbol{t}, \boldsymbol{n}, \boldsymbol{b}$ Unit vectors.

$\Gamma, \boldsymbol{p}, \boldsymbol{p}'$ Path and parametric representations.

$s, \dot{s}$ Timing law.

Control

$\boldsymbol{B}, \boldsymbol{K}, \boldsymbol{M}$ 3×3 Damping, Stiffness and Mass matrices.

$\boldsymbol{S}, \bar{\boldsymbol{S}}$ Selection matrix.

$\boldsymbol{u}$ Controller output.

ω_b Bandwidth.

T_s Sampling time.

Subscripts

ω Wrist.

c Compliant frame or controlled.

cd Displacement between compliant and desired frames

e End-effector frame or environment.

f Final.

i Initial.

o, O Rotation.

p Position.

P, I Proportional and Integral.

r Reference frame.

s Sensor.

u Undeformed.

Superscripts

-1 Inverse.

0 With respect to base reference/ origin frame.

c	With respect to compliant frame.
d	With respect to desired frame.
i	Initial.
T	Transpose.

Glossary

ADC	Analog-to-digital converter.
Base reference frame	Reference frame placed at the base od the manipulator.
CAD/CAM	Computer-aided design and computer-aided manufacturing.
CLIK	Closed-loop inverse kinematics.
CNC	Computer Numerical Control.
CPU	Central processing unit.
Cutoff frequency	Limiting frequency a filter can process. Signals at higher frequencies are attenuated.
DH	Denavit-Hartenberg.
DOF	Degree-of-freedom.
DSP	Digital signal processor.
End-effector	The tip of the manipulator.
FIFO	*First in first out* is a way of reading from a buffer. Data is ready in the same order as it has arrived.
FSM	Finite state-machine.
FTS	Force/ torque sensor.
GL	Grau de liberdade.
IFR	International Federation of Robotics.
IRB140	The ABB® manipulator used.
IRC5	The industrial controller from ABB® used.
OS	Operating system.
RAM	Random access memory.
RSI	Robot Sensor Interface.
RTOS	Real-time operating system.
SLRT	Simulink real-time.
SME	Small and Medium Enterprises.

Strain gages	Sensor whose resistance varies with applied force. Converts an applied force into change in electrical resistance.
TCP	Tool center point.
Tread	Smallest sequence of programmed instructions that can be managed independently by the CPU.

Chapter 1

Introduction

1.1 Background and Motivation

Industry is always seeking to optimize productivity and improve the work process with industrial robots playing a big part in it. The continuous development of robotics technology has brought an improvement of product quality standards and smaller manufacturing costs combined with the possibility of eliminating harmful or off-putting tasks for the human operator in a manufacturing system. That allied with flexibility of utilization in different tasks and the capability of been reprogrammable lead to a spread in an increasingly wider range of applications in manufacturing industry. Moreover, they can surpass humans due to their capabilities of performing autonomous and repetitive operations requiring high precision and large payloads in industrial settings. The *IFR World Robotics 2018* report [1] points towards a growth, robot sales in 2017 reach a new peak for the fifth year in a row. The use of industrial manipulators have increased the productivity and competitiveness of many companies. The big investment in industrial manipulators was made by big companies of automotive parts and electrical/ electronics components, Small and Medium-sized Enterprises (SMEs) are starting to see the benefits long term.

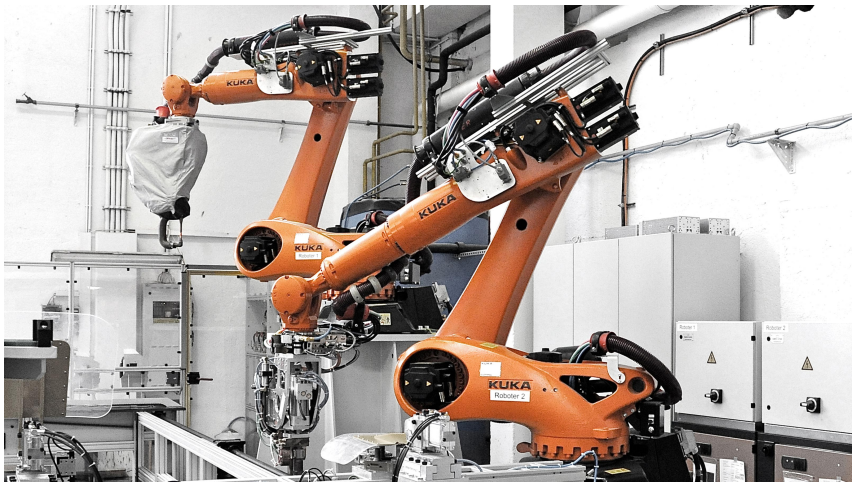


Figure 1.1: Industrial robotic work cell with two KUKA KR150 using different tools[1]

[1] Retrieved from: `https://www.kuka.com/en-de/industries/solutions-database/2016/07/solution-robotics-meiller`

Industrial robots are designed to meet the requirements for the widest set of potential applications, which is difficult to achieve. Classes regarding payload capacity, number of robot axes and workspace volume have emerged for application categories such as assembly, palletizing, painting, welding, machining and general handing tasks. Versatility enables robots to work in both rigid and flexible automation [2]. Robot-based working cells are more flexible and allow the production of different products at the same time, since they can be easily adapted. Figure 1.1 depicts an example of a robotic work cell with high flexibility used for manufacturing elevator doors custom sized. Two types of welding process (spot welding and projection welding) are performed in the robotic cell and one of the robots has an automatic tool system which allows it to handle the door panel.

Focusing in machining applications, robots could acquire all functionalities of CNC-machines and represent a reasonable alternative. Displaying good accuracy, they provide larger workspace and more flexibility. The same robot can realize diverse manufacturing processes, making it universal, while CNC-machines can only execute one or a group of similar operations. In modern Computer Numerical Control (CNC) systems the machining trajectory design is automated and performed in a CAD/CAM environment. It is used for creating spatial representation of a part and optimization of the tool paths and cutting parameters. Based on the control algorithm, CNC-machines can be sensitive to external disturbances by having a position/ velocity feedback.

By contrast, robots in machining applications often use force and torque sensors that allow online estimation of the deflections in tool locations with respect to the desired ones. The force and torque sensors are usually integrated into a robot's wrist, and the robot controller is able to do an appropriate modification of the prescribed robot motion upon the information provided by measuring the interaction force. They need to be more than position controlled, that will only be sufficient to perform tracking tasks where there is no interaction with the surrounding environment. Force control is up the essence to take in consideration the physical interaction between the robot tool and the working objects or surfaces, since forces and moments will arise from contact.

One of the most commonly used approaches in force control is Hybrid force/ motion control, which allows a robot manipulator to follow a position trajectory and simultaneously adjust the forces applied to the environment based on measurements from sensors, handling them as two separate sub problems. In [3], two different ways to implement a hybrid position/ force control in a KUKA KR6/2 were studied. One via a Robot Sensor Interface (RSI), which is an add-on technology [4], and the other using the robot system KUKA Force Torque Control Technology Package (FTCtrl), KUKA Roboter GmbH (2007). Both technologies can be purchased in the manufacturing company (KUKA) and provide the software interface between controller and sensor allowing the user to program an interaction task.

For each different application, commercial manipulators feature specific add-ins for their own controllers (drivers) and hardware tools or sensors. Apart from the high installation and maintenance costs, most of the economic efforts go to the remaining equipments of a robotic work cell that provide the environment for the robot operation. The idea of accomplishing similar interface and integrating solutions which allow a typical industrial manipulator to perform several different operations without any investment in a special robot controller with reliable accuracy is the motivation behind this work.

As technology allows us to transform and adapt industrial manipulators to better fulfil multiple purposes, avoiding excessive costs while perform precise robotic tasks, we take for granted the true meaning behind *industrial robot*, which according to the definition of the International Organization for Standardization is an "*automatically controlled, reprogrammable multipurpose manipulator programmable in three or more axes*" (ISO 8373:2012 [5]).

1.2 Objectives

The key goal of this thesis is to develop a force and moment control strategy that can be applied in robotic machining endowed by a standard industrial controller (motion controlled). Accordingly, such strategy must ensure the regulation of contact forces arising from interaction of the robot tool with surfaces and working objects to a certain range of work for the feasibility of the task assigned to the manipulator. It must be used as platform to increase robot system flexibility allowing the realisation of a handful of new applications.

Considering both strengths and weaknesses of standard industrial controllers addressed in previous works [6, 7], this thesis is based on the hardware-software architecture developed in [6]. Exploiting the potentialities and limitations of the high-level programming language of the industrial controller with the use of external sensors and software. Therefore, one of the main goals is to implement a force and moment control strategy suitable for robotic machining using a standard close-ended industrial controller without having access to the lower layers of the software, through a remote control application [6] between Simulink Real-Time™ and RAPID with a force/ moment sensor. Keeping in mind the communication between manipulator controller (IRC5) and the SLRT as bottleneck of the hardware-software architecture, since it is achieved via RS-232 with a sampling rate of $33Hz$.

1.3 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 is devoted to the state of art and literature review on the force/ moment control strategies. It includes a review on the classification of the interaction between manipulator and environment and addresses the problem of 6-DOF interaction tasks. Chapter 3 presents the experimental setup and describes both hardware and software used with the remote application. In Chapter 4 is proposed the force/ moment control strategy implemented. Moreover, is presented the position and orientation trajectory planner. In Chapter 5 the force/ moment proposed is experimentally validated using the hardware-software architecture designed and the main results are summarized and discussed. Finally, Chapter 6 presents the concluding remarks and suggestions for future work.

Chapter 2

State of the Art

The research field in force control of robot manipulators has been subject of investigation in the last three decades. Such a wide interest is motivated by the general desire of providing robotic systems with enhanced sensory capabilities. Robots using force, distance and visual feedback are expected to autonomously operate in unstructured environments. Force control plays a fundamental role in the achievement of robust and versatile behaviour of robotic systems in open-ended environments.

This chapter addresses the interaction between manipulator and the environment, regarding a non-minimal six-DOF interaction task and giving an overview on operational space control schemes to the constrained motion case. For practical reasons, a literature study is performed on the most common and relevant approaches of force control. In this study particular attention is paid not only to traditional indices of control performance, but to the reliability and applicability of algorithms and control schemes in industrial robotic systems.

2.1 Manipulator Interaction with the Environment

Purely motion control strategies have been successfully used on robotic tasks involving a null or weak interaction between the manipulator and its environment. An interaction task with the environment using position control could be only achieved if the task was accurately planned, like on CNC-machines. This would require an accurate model of both the robot manipulator (kinematic and dynamics) and the environment (geometry and mechanical features). Manipulator modelling can be known with enough precision, but a detailed description of the environment is difficult to obtain.

The design of the interaction control is carried out under two simplifying assumptions defined by Siciliano et al. in [8]. The robot and the environment are perfectly rigid and purely kinematic constraints are imposed by the environment; the compliance of the system is located in the environment and the contact force and moment are approximated by a linear elastic model. Frictionless contact is assumed, however, the robustness of the control should be able to cope with situations where some of the ideal assumptions are relaxed. Only interaction with compliant environments is of interest, since torque control is only possible if the robot is equipped with torque sensors and this is not the case for the industrial

robots used (see Section 3).

One of the fundamental requirements for the success of a manipulation task is the capacity to handle interaction between the manipulator and the environment, the control of this interaction is crucial for accomplishing a number of tasks where the robot's end-effector has to manipulate an object or perform some operation on a surface. As stated by Siciliano et al. [9], a complete classification of possible robot tasks is practically infeasible due to the large variety of applications, nor would such a classification be really useful to find a general strategy to control the interaction with the environment. Instead, the author categorized the interaction through the manipulator's ability to react to a contact force. The term compliance refers to a variety of different control methods in which the end-effector motion is modified by contact forces. A manipulator is able to have a compliant behaviour during the interaction and that can be achieved either passively or actively. Regarding the assumptions mentioned above, there is no compliance on the robot or in mechanical device with passive compliance. Therefore, active compliance involves constructing a force feedback in order to achieve a programmable robot reaction.

Vukobratovic et al. [10] have classified the various compliant motion control methods according to different criteria in a tree diagram. The authors schematized the force control methods by the application of the basic variables (position, velocity, acceleration, force and moment) and their relationship. The relevant branch to the present work is within the active compliance and concerns how the force feedback is made. The two categories are explicit and implicit control.

In explicit force control the contact force between the end-effector and a surface is measured by a force/ torque sensor (FTS) and compared with the desired value of force. The control action acts directly upon the force exerted by the environment and from the control error a suitable motion (torque inputs) of the robot is generated at the robot's joints. On the other hand, implicit force control seeks to control the dynamics of interaction. From the measured force a distance is calculated which modifies the current robot position, i. e., the force control error is converted to an appropriate robot motion adjustment and then added to the positional control loop.

During the interaction, the environment imposes constraints on the end-effector motion, denoted kinematic constraints. The contact with a stiff surface is generally referred to as *constrained motion* [2]. Motion can be constrained by the environment along both the translational and the rotational degrees-of-freedom which corresponds to a six-DOF interaction task [11]. Khatib [12] was the first to address the control of end-effector motion and contact forces with a general six-DOF controller, considering that both forces and moments may arise during the task execution when the end-effector tends to violate the constraints.

A suitable description of the end-effector orientation should be adopted to describe and perform a six-DOF interaction task. The usual minimal representation of orientation is given by a set of three Euler angles. According to [11], this formulation fails in the occurrence of representation singularities and can lead to an inconsistency between the moment applied during the task execution and the corresponding displacement in terms of Euler angles, they are not always aligned to the same axis. The latter is due to the fact that a set of three Euler angles does not represent a vector in the Cartesian space.

To overcome the drawbacks of the previous formulation, the author concluded that the unit quater-

nion, a non-minimal representation characterised by four parameters, is the most suitable way to represent the end-effector orientation. It has a physical meaning and mitigates the effects of representation singularities. Hereafter, it is assumed that unit quaternion representation will be use to describe the end-effector orientation in a six-DOF interaction task.

2.2 Force Control

The two main approaches to control the interaction of the manipulator and its environment are Hybrid force/ position control and Impedance Control. Hybrid control decomposes the task space into force and position controlled directions [13]. Many tasks are naturally described in the ideal case by such task decomposition. On the other hand, Impedance control does not regulate motion or force directly, instead regulates the ratio of force to motion [14].

Hybrid control enables the tracking of position and force references but requires an accurate model of the environment. The controller structure depends directly on the geometrical or analytical environment model. In practice, however, the environment is characterized by its impedance, which could be inertial (as in pushing), resistive (as in sliding, polishing or drilling) or capacitive (spring-like). In addition, the presence of modelling errors leads to unwanted movements along the force controlled directions and unwanted forces along position controlled directions. Hybrid control designs neglect the impedance behaviour of the robot in response to these imperfections and Impedance control only provides a partial answer, since contact forces cannot be directly imposed and may grow in an uncontrolled manner due to modelling errors of the environment impedance [15].

Concerning the complementary between Hybrid and Impedance control approaches, Anderson et al. [16] combined both and proposed the Hybrid Impedance control. The kernel part of the algorithm is Raibert and Craig's [13] hybrid position/force control scheme but with Impedance control in the position subspace instead of the usual position control. Both control parts use the force feedback to realize system impedance along each DOF, which allows a simultaneous regulation of impedance and either force or motion, to impose a desired manipulator behaviour. The main objective of this controllers is to achieve a robust behaviour under environment modelling errors.

Trying to cope with uncertainties in the environment geometry and regarding the task space decoupling of Hybrid control but in a full-dimensional space (without the selection matrices), Chiaverini et al. [17] presented the Parallel force/ position control. The control action is obtained as the sum of the two parallel control actions, force and position control. However, force control prevails over position control due to the integral control action, at the expense of a position error.

Several problems stated above are more acute during the transition between free space and interactive movements, requiring the use of some kind of controller switching strategy based on contact force information, which may induce an unstable manipulator behaviour. In the attempt of solving the transition between the two states without using any supervising strategy, Almeida et al. [18] proposed the Force-Impedance controller. It is a controller that behaves as an impedance (position) controller up to contact and afterwards, changes to an Impedance or Force control.

Some of the control strategies mentioned above assume that the force controller can act on the joint torque or motor current level of the robot actuated system and are not strictly related to the work developed in this thesis. Since, it is often not possible to modify the controller of an industrial robot to get access to the motor currents or joint torques and the force control will be implemented in a commercial manipulator that does not allow any modification on the motion controller (see subsection 3.1.2). Still they are worth being mentioned because they represent important and essential steps towards the implemented controller. Feasible solutions to implement in industrial robotic systems are the position based controllers [6, 19].

2.2.1 Impedance and Admittance Control

Hogan [14] was the first to propose Impedance control in his seminal work, both theoretical and practically. He used the notion of causality to explain that if a system has impedance causality the other must be an admittance. Most of the interaction task environments are either moveable objects or other fixed structures, so they should be considered as admittances. Due to the fact that systems that are connected through interaction ports must complement each other, the manipulator must be considered as an impedance.

Impedance control does not allow the user to specify a desired contact force, it is conceived to relate the position error to the contact force through a mechanical impedance of adjustable parameters. This mechanical impedance represents the dynamic relation between contact force and end-effector displacements and it is characterized by a generalized mass matrix, a damping matrix and a stiffness matrix. A robot manipulator under impedance control is described by an equivalent mass-spring-damper system with the contact force as input.

The selection of good impedance parameters to achieve a satisfactory behaviour during the interaction is not easy to obtain. The execution of a complex task, involving different types of interaction, may require different values of impedance parameters. The controller dynamics along the free motion directions is different from the dynamics along the constrained directions, in the free motion there is a need to ensure a good performance of the controller following a given reference by the manipulator. Whereas in the latter, the controller must confer a compliant behaviour to the robotic system. In addition, the selection of these parameters have to cope with the errors in modelled contact geometry and other imperfections, such as unknown robot dynamics (joint friction), measurement noise and other external disturbances. The more compliant the impedance control is in keeping forces limited (for low values of impedance), the more the closed-loop behaviour is affected by these disturbances.

A possible solution to this problem may be the Admittance control or Position-based impedance control, where the motion control is separated from the Impedance control and each problem is taken separately. Resorting to an external FTS, the author in [6] developed a controller of this kind, requiring joint position feedback as well as force sensing. The design is ideal for applications using commercial manipulators featuring a motion controller, which he used on an ABB® IRB 140 with an ABB® IRC5 controller. Thus, the position and orientation control is achieved by the industrial manipulator controller,

guaranteeing a rejection of the disturbances, and the gains of the impedance control laws (2.1) and (2.2) can be set to ensure a satisfactory behaviour during the interaction with the environment.

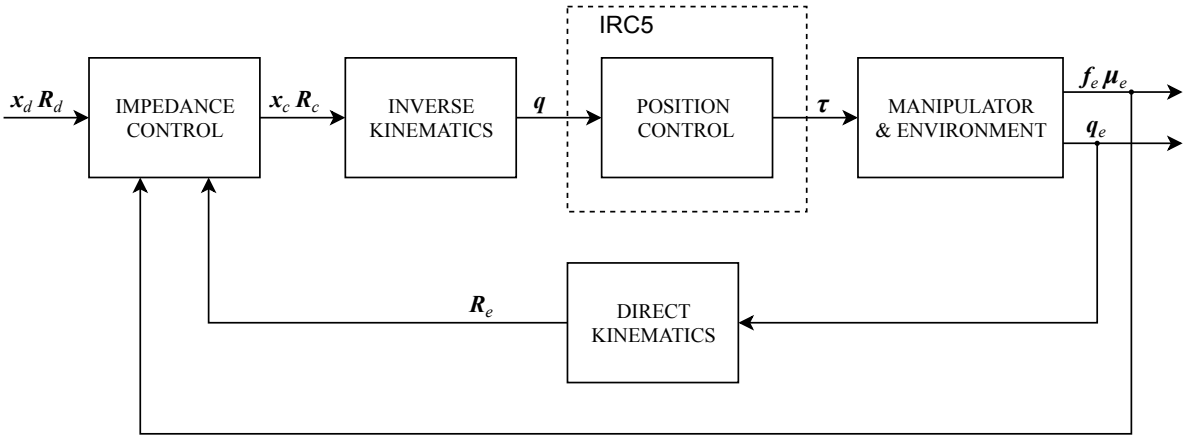


Figure 2.1: Block scheme of Position-based impedance control [9]

For simplification, the translational part and the rotational part of the impedance are not differentiated in the controller scheme represented in Figure 2.1, which represents the controller implemented. Even though, force and torque measurements were used to achieve a linear impedance control law for the translational and rotational directions (independently) and a non-minimal six-DOF interaction task (see Section 2.1) was considered.

The scheme is based on the concept of compliant frame [2], which is a suitable reference frame describing the ideal behaviour of the end-effector under impedance control. In other words, when the end-effector moves in free space during the task execution, it follows the desired trajectory. Whereas, when the end-effector is in contact with the environment, it shall follow the compliant trajectory. The translational impedance is given by the second-order dynamic equation:

$$\boldsymbol{F}_e = \boldsymbol{M}_d \Delta\ddot{\boldsymbol{x}} + \boldsymbol{B}_d \Delta\dot{\boldsymbol{x}} + \boldsymbol{K}_d \Delta\boldsymbol{x}, \tag{2.1}$$

where $\boldsymbol{F_e}$ is the 3×1 vector of generalized contact forces applied at the end-effector,$\boldsymbol{M}_d$, $\boldsymbol{B}_d$ and $\boldsymbol{K}_d$ are the 3×3 diagonal translational impedance parameters and the 3×1 vector $\Delta\boldsymbol{x}$ represents the positional error between the compliant and desired frames ($\Delta\boldsymbol{x} = \boldsymbol{x}_c - \boldsymbol{x}_d$).

The impedance equation for the rotational part can be defined analogously to the translational part and is given by [11],

$$\boldsymbol{\mu}_e^d = \mathbf{M}_o \Delta\dot{\boldsymbol{\omega}}_{cd}^d + \boldsymbol{B}_o \Delta\boldsymbol{\omega}_{cd}^d + \boldsymbol{K}'_o \boldsymbol{\epsilon}_{cd}^d, \tag{2.2}$$

where

$$\boldsymbol{K}'_o = 2\mathbf{E}^T(\eta_{cd}, \boldsymbol{\epsilon}_{cd})\boldsymbol{K}_o, \tag{2.3}$$

with

$$\boldsymbol{E}(\eta, \boldsymbol{\epsilon}) = \eta \boldsymbol{I} - \boldsymbol{S}(\boldsymbol{\epsilon}). \tag{2.4}$$

In the above equations (2.2–2.4), $\boldsymbol{\mu}_e^d$ represents the 3×1 vector of moments applied at the end-effector, $\boldsymbol{M}_o$, $\boldsymbol{B}_o$ and $\boldsymbol{K}_o$ are the 3×3 diagonal rotational impedance parameters, and $\boldsymbol{\epsilon}_{cd}^d$ is the vector part of the unit quaternion describing the orientation displacement (or the mutual orientation) between the compliant and desired frames with respect to the desired frame. Angular velocities $\boldsymbol{\omega}_{cd}^d$ are computed by integration of the quaternion propagation equations and the variation $\Delta\boldsymbol{\omega}_{cd}^d$ is the error between the angular velocities of the compliant and desired frames relative to the desired frame ($\Delta\boldsymbol{\omega}_{cd}^d = \boldsymbol{\omega}_c^d - \boldsymbol{\omega}_d^d$). The matrix $\boldsymbol{S}(\cdot)$ is the so-called skew-symmetric operator. For further understanding, see appendix A.1.

This method of control eliminates much of the computation required by the conventional methods. However it brings along a couple of liabilities in terms of bandwidth. As depicted in Figure 2.1 for the admittance control scheme, the outer loop is force-based while the inner loop is position-based. The stability of the overall system can only be ensured if the bandwidth of the motion control loop is higher than the bandwidth of the admittance controller [11].

2.2.2 Hybrid Position/ Force Control

If a detailed model of environment is available, a widely adopted strategy of force control is the Hybrid position/ force control which aims to control position along the unconstrained task directions and force along the constrained task directions. Two different formulations of Hybrid control can be found in the literature, the geometrical formulation and the analytical formulation. However, in this work we will not focus on the latter since it leads to loss of geometric and physical meaning of the variables involved.

The geometrical approach was proposed in [13] and it was motivated by the task analysis made in [20], where the author introduced the ideas of natural constraints and artificial constraints to model an interaction task from a geometrical point of view. Along each DOF of the task space, the environment imposes a force or a position constraint. Such constraints are termed natural constraints and are originated by the task geometry. The manipulator can only control the variables which are not subject to natural constraints, i.e., the unconstrained variables. The control structure uses the artificial constraints, imposed by the task execution strategy, to specify the objectives of the control system so that desired values can be imposed only onto those variables not subject to natural constraints. The approach is geometrical since the directions of the natural constraints are orthogonal to the directions of the artificial constraints, the two sets of constraints are complementary.

The hybrid position/ force controller proposed in [13] is not a control law, it is merely an architecture designed where different control laws might be applied. According to Mason's [20] definition, the term is used in a more general sense and defines any controller based on the division into force and position controlled directions. Figure 2.2 presents the control scheme that illustrates the main idea of the Hybrid position/ force control, two complementary control loops assigned to each task subspace with its own sensory system and control law. Where position and force are controlled in a non-conflicting way in two orthogonal subspaces defined in a task-specific frame, since either position or force is controlled along

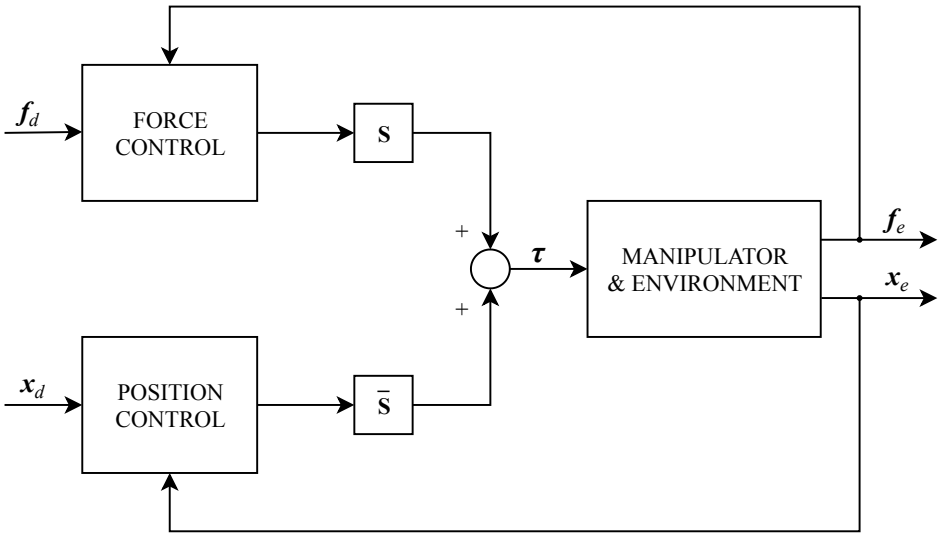


Figure 2.2: Block scheme of a hybrid force/ motion control for a compliant environment [2]

each task space direction.

Among the artificial constraints, a distinction between the motion controlled variables and the force controlled variables is made sorting to binary selection matrices. These matrices can only be applied to task geometries with limited complexity, for which separate control modes can be assigned. Bruyninckx et al. [21] defined and grouped interaction tasks in which position and force controlled directions are clearly identified and the task space decomposition can be included in hybrid control based on selection matrices.

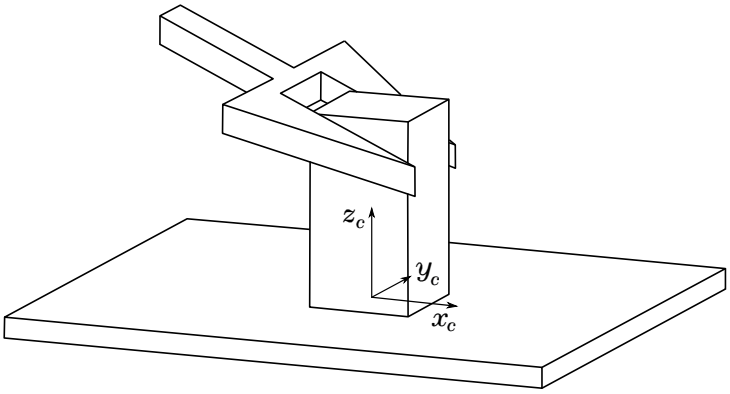


Figure 2.3: Sliding of a prismatic object on a planar surface [2]

Its basis resides on the controller simply ensuring the artificial constraints while neglecting the natural ones. Since there is no need to control variables already constrained by the environment. Let's take the practical case of sliding a prismatic object on a planar surface with constant speed. Considering the compliant frame [2] ($O_c - x_c y_c z_c$), the goal of this task is to slide a prismatic block or a tool over a planar surface along the x_c axis, while pushing with a given force against the surface. Choosing the constraint frame attached to the contact plane as shown in Figure 2.3, and assuming rigid and frictionless contact,

the planar surface imposes motion constraints on the prismatic object: linear velocity along axis z_c and angular velocities along axes x_c and y_c. Force constraints describe the impossibility to exert forces along axes x_c and y_c and moment along axis z_c. The artificial constraints regard the variables not subject to natural constraints. Therefore, it is possible to specify the artificial constraints for linear velocity along x_c, y_c, angular velocity along z_c, force along z_c and moments about x_c and y_c. Notwithstanding, in the presence of friction or if the manipulator is not able to reproduce the natural constraints, non-null force and moment may also arise along the velocity controlled DOFs.

On the other hand, Siciliano et al. [2] formulated a Hybrid force/ motion control using the acceleration-resolved approach, where the dynamic model of the robot is decoupled and linearized at the acceleration level. By computing the elementary displacement of the end-effector with respect to equilibrium pose, in terms of the end-effector acceleration, they can decompose the interaction in velocity controlled subspace and force controlled subspace. Through the inverse dynamics control law (A.12), the decoupling between force control and velocity control is achieved by the meaning of an acceleration referred to the base frame.

2.2.2.1 Position-based (implicit) Force Control

In a commercial robotic system it is suitable to implement implicit or position-based force control by closing a force-sensing loop around the position controller. The practical reason why the methods based on explicit force control can not be suitably applied in commercial robotic system lies on the fact that commercial robots are designed as ”positioning devices”.

De Schutter et al. [22] presented a method for compliant motion control based on the theory of Mason [20] in hybrid force/position. The author implemented it with a force-control loop around the position control and since the position controller provides a basis for realization of force control, this concept was referred as external force control or position-based (implicit) force control. As depicted in Figure 2.4, the input of the force controller is the difference between desired and actual contact force in the task frame. The output is an equivalent position in force-controlled directions that is used as the reference input to the position controller. Force control block in this scheme has a twofold role: to compensate for the effects of the environment (contact process), and to track the desired force. Commonly, a PI force controller is applied.

According to the hybrid position/ force control concept the equivalent position in force direction x_f is superimposed onto the orthogonal vector x_p in the compliance frame, which defines the nominal position in the orthogonal position-controlled directions. The robot behaviour in the force direction is practically affected only by the acting force. The position controller remains unchanged, except for the additional transformations between the Cartesian and task frames, which have to be introduced since in general case these two frames do not coincide.

As stated in [19, 23], the main features of implicit force control scheme are its reliability and robustness. Implemented in commercial robotic systems, this scheme allows the control design in a incremental manner, facilitating the tuning and validation. However, it exhibits some drawbacks regarding the accuracy of contact forces and is mainly limited by the precision of robot positioning (sensor resolution).

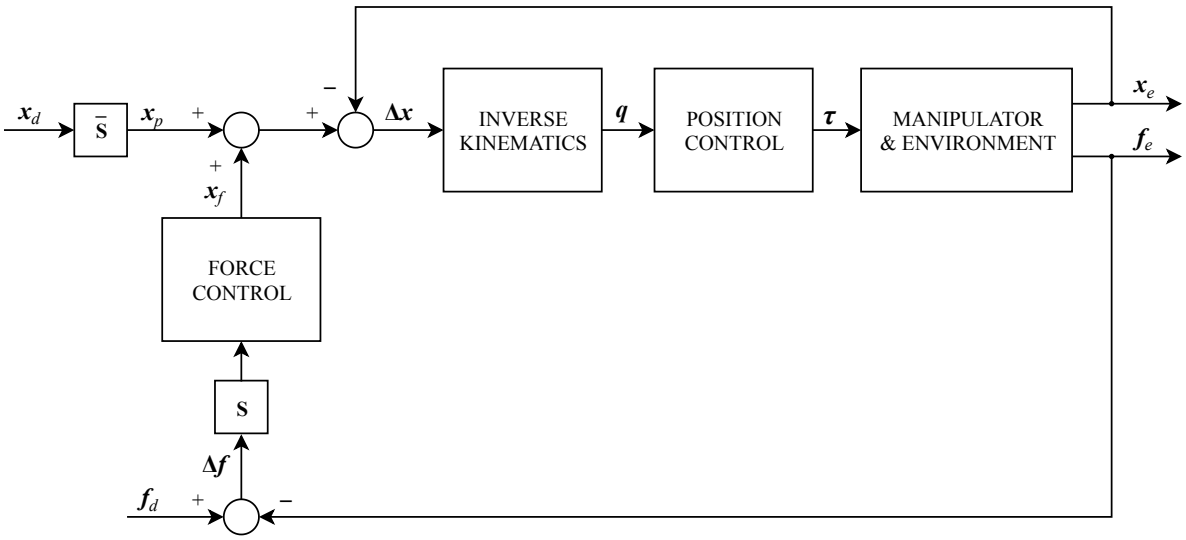


Figure 2.4: Block scheme of a position-based (implicit) force control [19]

It can become disturbed when at contact with a stiff environment. The performance of implicit force control is significantly limited by the bandwidth of the position controller.

2.2.3 Parallel Force/ Position Control

A model of unstructured environments is very difficult to obtain, hence Hybrid control can only be adopted in case of interaction between structured environments. In most practical situations, a detailed model of the environment is not available and modifying the behaviour of the controller according to the environment on real time is not possible. Moreover, the selection matrices nullify sensor information, considering it negligible, which could be helpful in situations when there is lack of knowledge about the environment.

To offer some robustness with respect to the uncertainties in the environment, Chiaverini et al. in [17] proposed the Parallel force/ position control. Firstly, the controller was only designed for force-position control then Natale et al. in [24] formulated it for moment and orientation. The key feature of the controller is having force and position control along each task space direction, i. e., the same task space direction is both force and position controlled without any selection mechanism. Position and force references must be specified for each task space direction. The controller combines a PD position control loop with a PI force control loop where the control action is obtained as the sum of the two parallel control loops. In the resulting Parallel control there is dominance in the force control action above the position one to ensure force control along each task directions, even in the constrained directions. That means, the force tracking is dominant to accommodate contact forces (planned and unplanned) in any situation, while the position control loop allows the compliance (deviation from the nominal position) to attain the desired forces at the expense of a position error. The prevalence from force control prevents the undesirable effects described for the case of hybrid control.

The force and motion control actions can be designed based on a simplified model of the environment while providing some sort of robustness to uncertainty. In fact, even if an unexpected force along a

planned unconstrained direction builds up, the force control reacts to regulate the force error to zero, which is the desired value along all the planned unconstrained directions.

To summarize, the main difference between the hybrid approach and the parallel approach is that, in the former, the contact geometry directly influences the structure of the controller, while in the latter, it influences only the references in terms of the desired motion and desired force [24]. Furthermore, in the control system the actual constrained and unconstrained directions are identified during task execution by properly using force measurements, without any filtering action.

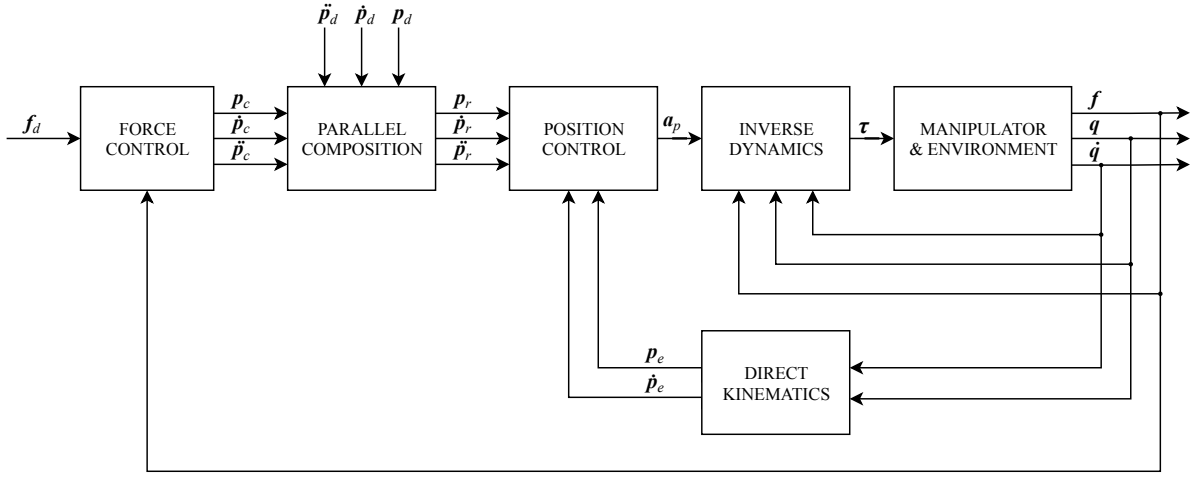


Figure 2.5: Block scheme of a Parallel Control [11]

In [8, 11] the authors formulated a Parallel Control variant for a six-DOF interaction task, where an inner motion loop should be designed and the references to be tracked should be suitably computed by an outer force loop. This way, force and moment are regulated along the constrained directions and the desired trajectory is tracked along the unconstrained directions. The resulting parallel controller for the translational part is outlined by the block scheme Figure 2.5. In the figure, the subscript r denotes the reference frame to be tracked, and its position $\boldsymbol{p}_r$ is computed through the technique of the parallel composition defined as,

$$\boldsymbol{p}_r = \boldsymbol{p}_c + \boldsymbol{p}_d \tag{2.5}$$

$$\dot{\boldsymbol{p}}_r = \dot{\boldsymbol{p}}_c + \dot{\boldsymbol{p}}_d \tag{2.6}$$

$$\ddot{\boldsymbol{p}}_r = \ddot{\boldsymbol{p}}_c + \ddot{\boldsymbol{p}}_d \tag{2.7}$$

being $\boldsymbol{p}_c$ in (2.5) the solution to the differential equation expressing the force control law

$$\boldsymbol{K}_{Pf}\ddot{\boldsymbol{p}}_c + \boldsymbol{K}_{If}\dot{\boldsymbol{p}}_c = \Delta\boldsymbol{f} \tag{2.8}$$

with $\Delta\boldsymbol{f} = \boldsymbol{f}_d - \boldsymbol{f}$. It is noteworthy that $\boldsymbol{p}_c$ resulting from integration of the (2.8) provides an integral control action on the force error.

In regard the rotational part, it is worth pointing out that the desired orientation trajectory is specified

as a relative orientation between the desired frame and the compliant frame, in the sense that the quaternion $\mathcal{Q}_{dc}$ is a rotation about an axis aligned to an unconstrained direction and defined in the compliant frame. Therefore, the parallel composition for the rotational part is defined as,

$$\mathcal{Q}_r = \mathcal{Q}_c * \mathcal{Q}_{dc} \tag{2.9}$$

$${}^c\boldsymbol{\omega}_r = {}^c\boldsymbol{\omega}_c + {}^c\boldsymbol{\omega}_{dc} \tag{2.10}$$

$${}^c\dot{\boldsymbol{\omega}}_r = {}^c\dot{\boldsymbol{\omega}}_c + {}^c\dot{\boldsymbol{\omega}}_{dc} \tag{2.11}$$

where $\mathcal{Q}_c$, $\boldsymbol{\omega}_c$ and $\dot{\boldsymbol{\omega}}_c$ characterize the rotational motion of the compliant frame. These quantities can be computed since the rotational motion of the compliant frame has been computed according to the differential equation expressing the moment controller

$$\boldsymbol{K}_{Po}{}^c\dot{\boldsymbol{\omega}}_c + \boldsymbol{K}_{Io}{}^c\boldsymbol{\omega}_c = \Delta^c\boldsymbol{\mu} \tag{2.12}$$

with $\Delta^c\boldsymbol{\mu} = {}^c\boldsymbol{\mu}_d - {}^c\boldsymbol{\mu}$. As for the previous force and position control scheme, $\mathcal{Q}_c$ results from integration of the (2.12) together with the quaternion propagation (see Appendix A.1) providing an integral control action on the moment error.

The block diagram of the resulting moment and orientation control scheme is the counterpart of the force and position control scheme in Figure 2.5, where the moment control generates the orientation, angular velocity and angular acceleration of the compliant frame for $\Delta^c\boldsymbol{\mu}$. The parallel composition is computed using the relative desired rotation with respect to the time varying compliant frame (2.9–2.11) in order to generate the corresponding reference quantities (input of the orientation control).

Chapter 3

Experimental Setup

This chapter provides a description of both hardware and software used. It is important to describe the components used before the force and moment control developed, since the control strategy must be built taking into account both strengths and limitations of the hardware and software available. Additionally, the performance and effectiveness of such strategy depends on both the hardware and software used as well as on the communication between components.

Based on the equipment available at the ACCAII Laboratory of Instituto Superior Técnico, the ABB® IRB140 industrial manipulator endowed by an ABB® IRC5 industrial controller was chosen as the robotic system for this thesis. The main goal was to develop a force-moment control strategy suitable to perform robotic tasks with a standard industrial controller using a FTS that can sense contact forces arising from interaction with the environment on the end-effector. Thus, the sensor chosen to give was a JR3® 6-DOF force-moment sensor. Finally, an external computer running a Simulink Real-Time™ model was used to run the RTOS where the controller was implemented.

3.1 Hardware Overview

The experimental setup used consists of four elements: the industrial manipulator IRB140, the IRC5 controller, the FTS and the external computer. Figure 3.1 shows the connections between these components. The FTS is attached to the end-effector and is directly connected to the external computer via a PCIe Ethernet adapter while the communication between the IRC5 controller and the external computer is established via RS-232. Motion control is guaranteed by the IRC5 controller. In the following sections, each of them is briefly described.

3.1.1 ABB® IRB 140

The IRB 140, shown in Figure 3.2, is a compact 6-DOF industrial manipulator that has 6 revolute joints, a payload capacity of 6kg and weighs 98kg [25]. The manipulator structure consists of an anthropomorphic arm on the first three joints (1-3) and spherical wrist on the last three (4-6), the working range of each joint is illustrated on Table 3.1.

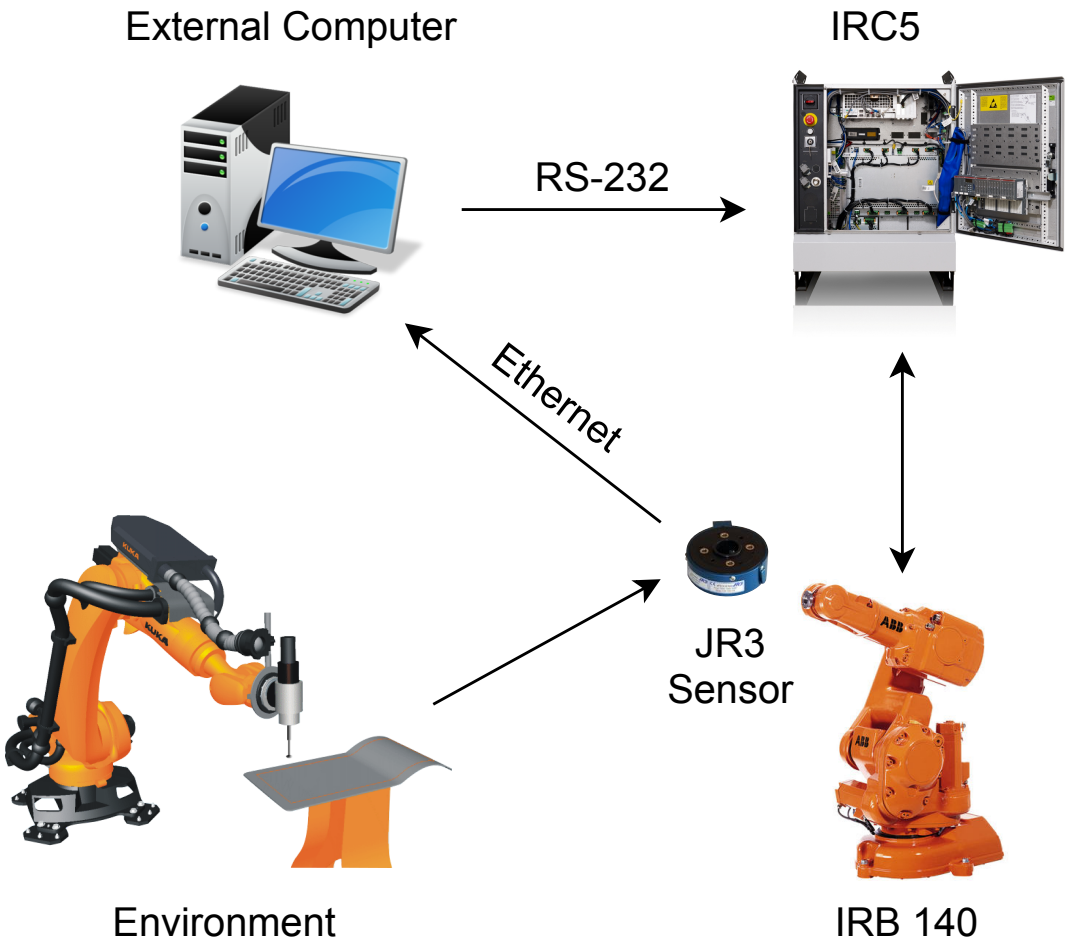


Figure 3.1: Experimental Setup

Figure 3.2: IRB 140 manipulator [25]

Table 3.1: IRB 140 range of motion

Axis	Range of motion	
	Max.	Min.
1	$+180°$	$-180°$
2	$+110°$	$-90°$
3	$+50°$	$-230°$
4	$+200°$	$-200°$
5	$+115°$	$-115°$
6	$+400°$	$-400°$

Mathematically speaking, this manipulator can be assessed as a kinematic chain of rigid bodies (links) connected by means of revolute joints. Then, with the right model, it can be expressed either in joint torque space τ, joint space $q(t)$ or operational space (Cartesian space $x_e(t)$). The transformation

between the three representative spaces can be established by a kinematic or a differential relation.

3.1.1.1 Direct Kinematics

Direct kinematics consists on describing the end-effector pose (position and orientation) as a function of the joint coordinates with respect to the base reference frame. Thus, for a n-DOF manipulator, this function is given by the following homogeneous transformation matrix,

$$\boldsymbol{T}_n^0(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{R}_n^0 & \boldsymbol{p}_n^0 \\ \boldsymbol{0} & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{n}_n^0 & \boldsymbol{s}_n^0 & \boldsymbol{a}_n^0 & \boldsymbol{p}_n^0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.1}$$

where $\boldsymbol{R}_n^0$ is the rotation matrix of the frame attached to the end-effector with respect to the base reference frame, $\boldsymbol{n}_n^0, \boldsymbol{s}_n^0, \boldsymbol{a}_n^0$ are the unit vectors of the frame attached to the end-effector, $\boldsymbol{p}_n^0$ is the position vector of the origin of such frame expressed in the base frame and $\boldsymbol{q}$ is the $n \times 1$ vector of joint variables. A systematic approach to define the coordinate frame associated to each link is given by the Denavit-Hartenberg Convention (DH), whose rules are exhaustively described in [2]. By attaching a coordinate frame to each link and defining a homogeneous transformation matrix $\boldsymbol{A}_i^{i-1}$ for each frame pair i and $i-1$ as a function of the joint variable $\boldsymbol{q}_i$, the direct kinematics function can be written as,

$$\boldsymbol{T}_n^b(\boldsymbol{q}) = \boldsymbol{A}_1^0(q_1)\boldsymbol{A}_2^1(q_2)...\boldsymbol{A}_n^{n-1}(q_n), \tag{3.2}$$

which, in general, is nonlinear.

The coordinate frames obtained following the steps of this approach for the IRB 140 links are shown in Figure 3.3 and the resulting DH parameters are shown in Table 3.2. The manipulator dimensions in the figure are depicted in millimetres.

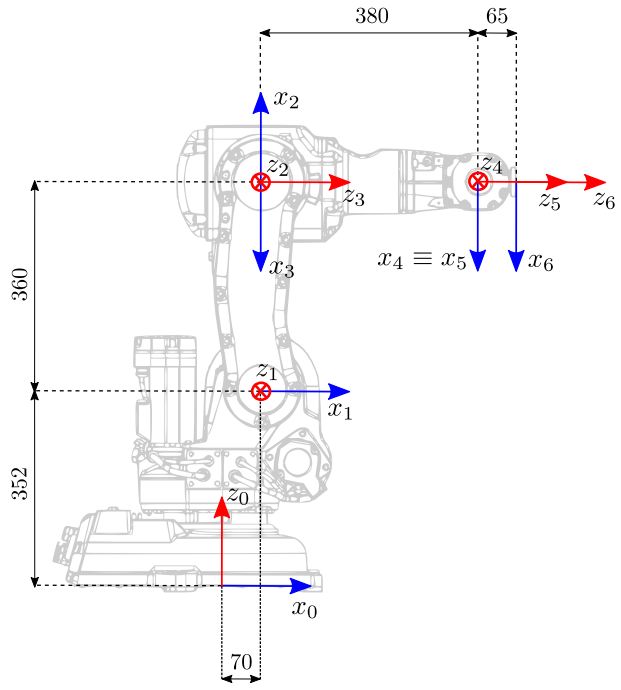


Figure 3.3: IRB 140 DH reference frames

Table 3.2: IRB 140 DH parameters.

Link	d_i(mm)	v_i	a_i (mm)	α_i (°)	Ref. (°)
1	352	q_1	70	-90	0
2	0	q_2	360	0	-90
3	0	q_3	0	90	180
4	380	q_4	0	-90	0
5	0	q_5	0	90	0
6	65	q_6	0	0	0

3.1.1.2 Inverse Kinematics

In opposition to direct kinematics, the inverse kinematics problematic consists of finding the joint coordinates that correspond to a desired end-effector pose. However, the problem of inverse kinematics is much more complex and in general nonlinear, which translates into the solution often being non unique or not obtainable in closed form. When obtainable, the computation of closed form solutions requires algebraic or geometric intuition to find significant points on the structure with respect to which is convenient to express position and/ or orientation as function of a reduced number of unknowns. An alternative method to solve the inverse kinematics of redundant manipulators is CLIK (Closed-Loop Inverse Kinematics), which is exhaustively described in [2].

Siciliano et al. in [2] have derived the solution of manipulators with a spherical wrist and the solution of an anthropomorphic arm, which are the constituents of IRB 140 structure. Nevertheless, there are misalignments regarding the DH reference frames between the general cases, for which the solutions were derived, and the IRB 140 that needs to be considered. There are 4 configurations of an anthropomorphic arm compatible with a given wrist position and 2 solutions for a spherical wrist, which means there are up to 8 admissible manipulator configurations (sets of joint coordinates). These in turn result in the same end-effector pose. Calculating the inverse kinematics for the IRB 140 also requires defining an additional function to choose the optimal configuration for each situation. Since it is not the scope of the presented work, a simple condition user defined was chosen.

The solution is divided in two parts, at first the aim is to find the joint variables q_1, q_2 and q_3 for a given wrist position $\boldsymbol{p}_w$ (anthropomorphic arm solution), then to find q_4, q_5 and q_6 for a given end-effector orientation $\boldsymbol{R}_6^3$ (spherical wrist solution).

Direct kinematics for the wrist position $\boldsymbol{p}_w$ is expressed by (3.2) for the DH parameters in Table 3.2, end-effector position and orientation is specified in terms of $\boldsymbol{p}_0^6 = \boldsymbol{p}_e$ and $\boldsymbol{R}_0^6 = [\boldsymbol{n}_e \quad \boldsymbol{s}_e \quad \boldsymbol{a}_e]$. In view of IRB 140 DH reference frames (Figure 3.3), the position of the wrist can be geometrically obtained by,

$$\boldsymbol{p}_w = \boldsymbol{p}_e - \boldsymbol{a}_e d_6. \tag{3.3}$$

From the anthropomorphic arm solution in [2], considering the misalignments for IRB 140 frames and

setting $a_3 = d_4$, joint variables q_1, q_2, q_3 are given by,

$$\begin{aligned} q_1 &= \text{atan2}(\pm p_{w_y}, \pm p_{w_z}) + \pi \\ q_2 &= \text{atan2}(s_2, c_2) - \frac{\pi}{2} \\ q_3 &= \text{atan2}(s_3, c_3) - \frac{\pi}{2} \end{aligned} \tag{3.4}$$

with,

$$c_3 = \frac{\tilde{p}_{w_x}^2 + \tilde{p}_{w_y}^2 + \tilde{p}_{w_z}^2 - a_2^2 - d_4^2}{2a_2 d_4} \tag{3.5}$$

$$s_3 = \pm\sqrt{1 - c_3^2} \tag{3.6}$$

$$c_2 = \frac{\pm\sqrt{\tilde{p}_{w_z}^2 + \tilde{p}_{w_y}^2}(a_2 + d_4 c_3) + \tilde{p}_{w_z} d_4 s_3}{a_2^2 + d_4^2 + 2a_2 d_4 c_3} \tag{3.7}$$

$$s_2 = \frac{\tilde{p}_{w_z}(a_2 + d_4 c_3) \mp \sqrt{\tilde{p}_{w_x}^2 + \tilde{p}_{w_y}^2} a_2 a_3}{a_2^2 + d_4^2 + 2a_2 d_4 c_3}. \tag{3.8}$$

Following the computation of q_1 a correction must be made to the wrist position, since frames 0 and 1 have different origins in Figure 3.3 while in the anthropomorphic arm solution they have the same. Thus, the corrected $\boldsymbol{p}_w$ vector is

$$\tilde{\boldsymbol{p}}_w = \boldsymbol{p}_w - \begin{bmatrix} a_1 c_1 \\ a_1 s_1 \\ d_1 \end{bmatrix}. \tag{3.9}$$

Afterwards, direct kinematics is calculated for the first three joint variables (3.4) obtaining the rotation matrix of the reference frame on joint 4 with respect to the base frame $\boldsymbol{R}_3^0$. Hence, is possible to obtain the end-effector orientation $\boldsymbol{R}_6^3$ for which the spherical wrist solution [2] will correspond computing

$$\boldsymbol{R}_6^3 = \boldsymbol{R}_3^{0^T} \boldsymbol{R}_6^0 = \begin{bmatrix} n_x^3 & s_x^3 & a_x^3 \\ n_y^3 & s_y^3 & a_y^3 \\ n_z^3 & s_z^3 & a_z^3 \end{bmatrix}. \tag{3.10}$$

From the rotation matrix expression it is possible to compute the joint variables q_4, q_5, q_6 directly, i. e.,

$$\begin{aligned} q4 &= \text{atan2}(a_y^3, a_x^3) \\ q5 &= \text{atan2}(\sqrt{a_x^{3\,2} + a_y^{3\,2}}, a_z^3) \\ q6 &= \text{atan2}(s_z^3, -n_z^3) \end{aligned} \tag{3.11}$$

for $q_5 \in [0, \pi]$, and

$$
\begin{aligned}
q4 &= \text{atan2}(-a_y^3, -a_x^3) \\
q5 &= \text{atan2}(-\sqrt{a_x^{3\,2} + a_y^{3\,2}}, a_z^3, a_z^3) \\
q6 &= \text{atan2}(-s_z^3, n_z^3)
\end{aligned}
\tag{3.12}
$$

for $q_5 \in [-\pi, 0]$.

From the aforementioned 8 configurations for the same pose, user must define in which the manipulator will operate. Therefore, binary values were assigned for shoulder ($0-$right and $1-$left), elbow ($0-$up and $1-$down) and wrist ($0-$up and $1-$down) positions to cope with all the possible configurations. To each combination of the 3 binary values correspond one of the inverse kinematics solutions.

3.1.1.3 Differential Kinematics and Statics

Direct and inverse kinematics establish the relation between the joint coordinates and the end-effector pose. On the other hand, the differential kinematics defines the relationship between the joint velocities $\dot{\boldsymbol{q}}$ and the corresponding end-effector linear $\dot{\boldsymbol{p}}_e$ and angular $\boldsymbol{\omega}_e$ velocities. For a n-DOF manipulator, the differential kinematics function is defined by,

$$
\boldsymbol{v}_e = \begin{bmatrix} \dot{\boldsymbol{p}}_e \\ \boldsymbol{\omega}_e \end{bmatrix} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{3.13}
$$

where $\boldsymbol{J}$ is the manipulator geometric Jacobian ($6 \times n$ matrix), which is a function of the manipulator configuration $\boldsymbol{q}$.

The inverse differential kinematics, as the name implies, determines the joint velocities that result in the given end-effector linear and angular velocities. Since the manipulator Jacobian matrix defines a local linear mapping for a given configuration, the inverse kinematics problem can be locally solved using the inverse of the Jacobian, $\boldsymbol{J}^{-1}$,

$$
\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\boldsymbol{q})\boldsymbol{v}_e \tag{3.14}
$$

Due to the linearity above highlighted, the manipulator Jacobian is a useful tool to establish the relation between the generalized forces applied to the robot end-effector $\boldsymbol{F}_e$ and the corresponding joint torques $\boldsymbol{\tau}$,

$$
\boldsymbol{\tau} = \mathbf{J}^T(\boldsymbol{q})\mathbf{F}_e \tag{3.15}
$$

3.1.2 ABB® IRC5 Industrial Controller

The IRB 140 manipulator described in the previous subsection is endowed with an IRC5 industrial controller, shown in Figure 3.4. This controller is computer-based and runs the VxWorks® real-time operating system (RTOS). The firmware responsible for loading and booting the OS (operating system) is the RobotWare, which is developed by ABB® itself.

Figure 3.4: IRC5 industrial controller

As stated by Lucas in [7], the IRC5 controller is comprised by two main modules: the control module and the drive module. The latter is responsible for supplying power to the robot motors while the former consists of the main computer and the axis controller. The main computer acts as a high-level robot controller responsible for trajectory planning and managing of the control parameters. On the other hand, the axis controller executes the low-level motion control receiving the references generated by the main computer and tracking them using a PID-based control law for each joint, as depicted in Figure 3.5. In this way, each joint is controlled independently using a position feedback provided by the joint resolvers which means that the IRB 140 manipulator is controlled as a decoupled linear system via a decentralized position controller.

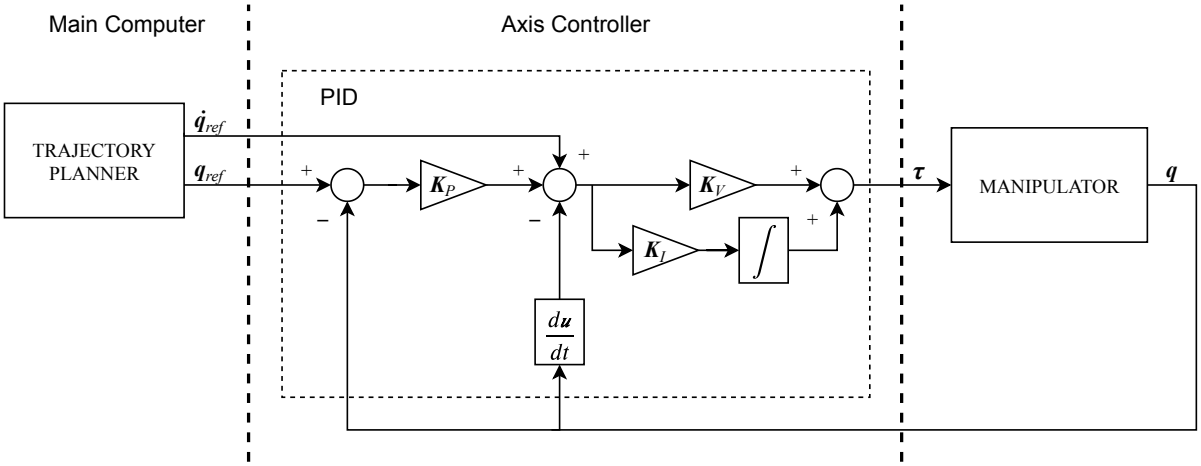


Figure 3.5: IRC5 control module [7]

Unfortunately, the manufacturer do not grant the user access to the lower layers of the software running on the IRC5, which means that the end-user is not aware about the trajectory planning algorithms employed by the main computer and has no access to the low-level motion controller. This also means that the only way to program the IRC5 is by using its own high-level programming language, RAPID, that was designed by ABB®. Moreover, the lack of access to the low-level motion controller poses a serious challenge for this thesis work, since most of the force-moment control strategies are formulated

on joint torques τ or accelerations, which, in general, are directly send to the low-level motion controller, bypassing the high-level layers. Leaving latency and bandwidth strictly dependent on the manipulator's controller.

3.1.3 Force/ Moment Sensor

JR3® 6-DOF force-moment sensor converts forces and moments along three orthogonal axes into electronic signals. Its built-in strain gages convert mechanical loads into analogue signals which are then digitalized by an analogue-to-digital converter (ADC). Raw data is then processed by the digital signal processor (DSP), which is integrated in a board connected to the external PC through a PCIe bus and monitors the incoming data. The sensor is powered directly from the computer, having another integrated circuit on-board responsible to manage the power supplied to it. The computer can access data from the DSP through a RAM.

The JR3 system can provide decoupled and digitally filtered data at 8kHz. However, the bandwidth of the DSP is lower than that since the cutoff frequency of a filter is 1/16 of the sample rate for that filter. Incoming raw data passes through 6 cascaded low-pass filters, each having a cutoff frequency 4 times lower than the preceding filter and causing a delay in the signal, approximately equal to

$$delay \approx \frac{1}{Cutoff \quad Frequency} \tag{3.16}$$

Previously to the work developed in [6], there was other works at ACCAII for which was already developed a driver for the SLRT. It deals force with decoupling in the 6-DOF and removes offsets [26]. Thus, no identification of the sensor was needed.

The JR3 sensor has a cylindrical shape making it suitable to be mounted between the end-effector and the tool. When coupled to the manipulator's end-effector the sensor Cartesian reference frame is rotated from the end-effector reference frame ($\theta \approx 26^{\circ}$).

For a matter of modelling and control, was assumed that the sensor casing is infinitely stiff and the strain gages deform linearly with the input force, thus having no significant underlying dynamics. Which holds true as long as the sensor signal is not too filtered. In that way, it can be assumed that the applied force at the sensor is equal to one applied at the end-effector. Since moment resulting from forces applied in a direction orthogonal to sensor $z-$axis is neglected, the vector of generalized forces exerted at the end-effector is given by (3.17).

$$\boldsymbol{F}_e = \boldsymbol{R}_s^e \boldsymbol{F} \quad \text{where,} \quad \boldsymbol{R}_s^e = \begin{bmatrix} -\cos(\theta) & -\sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

3.1.4 External Computer

The external computer is responsible for acquiring data from the force/ moment sensor and to communicate with the IRC5 controller. As it will be explained in subsection 3.2.1 the external computer works

as a target computer. Therefore, information from the host computer is not important to mention since it will only be used to create, compile and upload an application on the target. Target computer is the one that must endure with the application uploaded.

This computer runs a RTOS, Simulink Real Time™, and its technical specifications are depicted in Table 3.3. Besides the specifications, the motherboard features 1 PCI bus, 1 PCIe bus and a serial port to allow connections with other devices.

Table 3.3: External computer technical specifications.

OS	CPU	RAM
Simulink Real-Time™	Intel Pentium 4	2 Gb

3.2 Software Overview

There are two computers running two different operating systems that make part of the experimental setup: the IRC5 controller and the external (target) computer. In the following, the software running in each of them is described along with the communication interface between them.

3.2.1 Simulink Real-Time™

Simulink® is a MATLAB-based block diagram environment for model-based design, simulation and analysis of multidomain dynamic systems. It is developed by MathWorks and is designed to run on common computer operating systems. When used together with MATLAB® combines text and graphical programming to design a system in simulation environment enabling test it before moving to hardware. Simulink Real-Time™ is the real-time operating system from MathWorks that allows users to create real-time Simulink® applications and run them on dedicated target computers connected to physical systems. Real-time operating systems are able to process data as it comes in without buffering delays and have minimal latency in interrupting and thread switching.

Although SLRT is designed to work together with Speedgoat target computer hardware, it is possible to use other machines without loss of performance as long as certain parameters are met [27]. As in the case of the external computer mentioned in the previous subsection.

Target computer only runs the real-time application but features no means for interaction with the user. Actual programming is done in the host computer, that can be any computer installed with an ordinary OS and running MATLAB®. The host computer creates an application, compiles it to C++ code and uploads it to the target computer. Apart from that, allows the user to start/stop simulations, change the fundamental sampling time and to collect data. The connection between target and host computers is made through TCP/IP interface.

SLRT has block sets representing drivers for commonly used communication and I/O protocols that are connected through the PCI and PCIe computer ports. Concerning industrial communication proto-

cols, drivers for CAN, EtherCAT, raw Ethernet, RS-232 and real-time UDP are provided in the Simulink® Library. Since RS-232 was the only communication protocol available in the IRC5 controller on the ACCAII laboratory is the one used for the communication between operating systems (see subsection).

3.2.2 Robotstudio and RAPID

The interface with the IRC5 controller and thus with the IRB 140 manipulator is made through ABB's simulation and offline programming software RobotStudio. This software is an application for modeling, offline programming and simulation of robot cells. It creates a virtual IRC5 controller, that emulates an exact copy of the software (RobotWare) running on the real controller, and enables realistic simulation and programs debugging on an external PC before uploading them to the real controller. When connected with the real IRC5 via IPv4 connection, RobotStudio gives the user access to the IRC5 parameters, as well as allowing the programs creation and upload.

As stated before, the only way to program the IRC5 and consequently in RobotStudio is by using the RAPID high-level programming language. A RAPID task (program) is divided into modules, which in turn consist of a number of routines with a combination of instructions. There are three types of routines: *procedures*, *functions* and *trap* routines, and they can contain three kinds of data: *constants*, *variables* and *persistents*, which can be assigned to a certain data type. There are many data types defined in RAPID, but all of those are based on only three: *num*, *string* and *bool* [28].

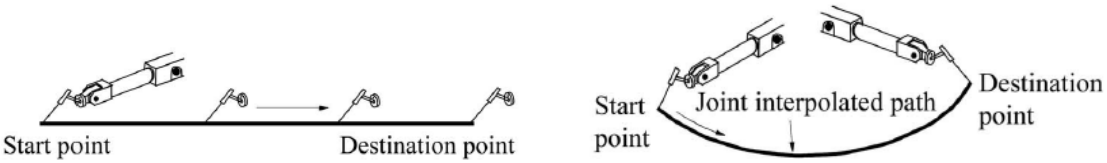
A task can be divided in many modules but only one contains the entry procedure, called *main*. Therefore, executing a RAPID task means executing the *main procedure* that may call other routines contained in different modules. The RAPID task is executed sequentially instruction-by-instruction, however, it is possible to control the program flow by using *IF* conditions, *WHILE* and *FOR* loops and interrupts. An interrupt occurs when a certain condition is met, causing the suspension of the normal program execution by transferring the program flow control to the corresponding *trap* routine. When the instructions of the trap routines are executed, the program execution continues from where the interrupt occurred.

By resorting to a feature called *Multitasking*, several tasks can be executed at the same level by the IRC5 controller in a round-robbin way [28]. This feature allows running tasks in (pseudo) parallel, however it is not advisable to use when real-time conditions are required since it brings along a concern about asynchronism. If a task is longer than the fundamental quantum, than there is no way to assure that every task is running at a fixed sampling time [6].

Moving the manipulator is done by calling a `Move` function, the robot motion is programmed as pose-to-pose movements which means that the robot moves from the current position to the one specified by the user in the positioning instruction employed. Among the various motion instructions available in RAPID, the main ones are `MoveAbsJ` and `MoveL`, functions that differ in the position data input. The former takes joint coordinates of the desired manipulator pose, while the latter, operates on the operational space, the position data argument is defined by the TCP Cartesian position and orientation. That is, when `MoveL` is used, the IRC5 needs to solve the inverse kinematics problem, since the motion

control is performed in the joint space.

Moreover, the path and trajectory planning between two positions is carried out by the IRC5 controller according to the function used. When using `MoveAbsJ`, it is applied an independent linear interpolation to each joint such that all joint reaches their desired coordinate at the same time. This motion instruction is the fastest and robustest but mostly unpredictable way to move the manipulator and is usually used when the TCP path accuracy is not too important [28]. When the `MoveL` instruction is employed, the TCP moves along a straight line between the start and destination points, which means that the joints follow a non-linear path in the joint space. Furthermore, when moving near a singularity, the linear interpolation used may cause problems, since the velocity of some joints may be too high and, on those cases, the linear path velocity is reduced.



(a) (b)

Figure 3.6: End-effector path with `MoveL` (a) and `MoveAbsJ` (b) [28]

Figure 3.6 depicts the way a path is interpolated by the two above-mentioned `Move` functions, which have the following syntax:

```
Move [Robtarget/jointtarget] [speeddata] [zonedata] [tooldata];
```

Note that besides the difference on the position data, the argument data of `MoveL` and `MoveAbsJ` is the same. The argument data of the positioning instructions defines the robot motion characteristics and is described in the following:

- `jointtarget` - defines the target joint coordinates in degrees. Used with the `MoveAbsJ` instruction.
- `Robtarget` - defines the target TCP Cartesian position and orientation with reference to the base frame, expressed in millimeters and quaternions, respectively. Used with the `MoveL` instruction.
- `speeddata` - specifies both linear and orientation desired velocity of the TCP in mm/s and in $^\circ/s$, respectively. As an alternative to the velocity, the duration of the motion in seconds can be specified.
- `zonedata` - specifies the position accuracy by defining the destination point as a fly-by or as a stop point. If it is defined as a stop point, the robot will stop when reaches that point and only after can start the movement towards the next point. On the other hand, fly-by points are used to get continuous movement through the programmed points. In order to achieve a continuous movement, a corner path is generated in the vicinity of the destination point (see Figure 3.7), which means that the programmed point is not reached. This corner path allows the robot to smoothly adjust its motion towards the next programmed position. This data argument specifies the radius

in millimeters of the zone defined around the destination point that determines the corner path. In this way, if the radius is defined as 0, the destination point is treated as a stop point.

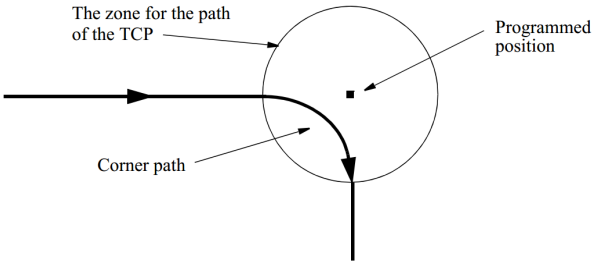


Figure 3.7: Corner path [28]

- `tooldata` - specifies the objects coupled to the end-effector (force/ moment sensor and contact tool or gripper) used during the movement. The tool is already defined in RAPID and it is specified in this data argument. Tool data describes the characteristics of a tool and can affect robot movements. The TCP position is used to calculate the velocity and the corner path of the movement, that it, only the TCP will follow the specified path at the programmed velocity. The characteristics needed to describe a tool are detailed in [28] and the tool system used is depicted in Figure 3.8 (dimensions are in mm).

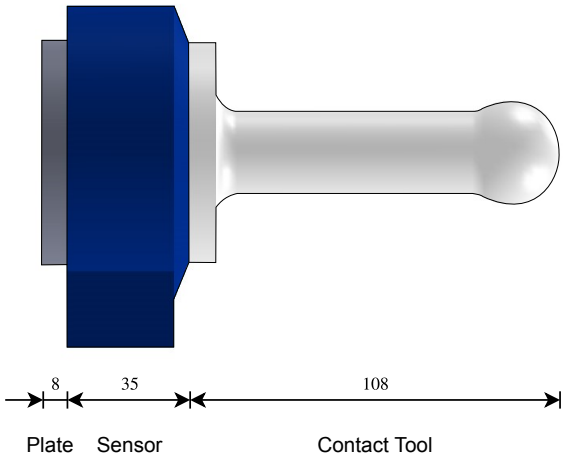


Figure 3.8: Tool System

There are also RAPID functions to read the current position of the manipulator, both in Cartesian and joint space. The `CRobT` function returns the current TCP position while the `CJointT` function returns the current joint coordinates. This functions are worth to mentioning due to their role on building the working space, in the form of safety conditions.

3.2.3 Remote Control

Following the work developed by Sampaio in [6], the only viable communication protocol for the interface between the external computer and IRC5 controller is through RS-232, a protocol for serial based com-

munications. It works on lower baudrate when comparing with TCP/IP or UDP protocols, but it works well for small amount of data exchange and with a 2 meter cable has no communication latency (installation on the ACCAII laboratory).

The remote control algorithm created by the author relies on the synchronism between the two computers on reading, writing and transmitting information in buffers. Each buffer includes 6 joint values, an header and a terminator, all represented as floating points. The header and the terminator are responsible to ensure that no message is corrupted, both RAPID and SLRT will abort simulation if the values differ from each other.

Synchronism is obtained on the SLRT side at read/ write functions level, hence, due to user access restrictions, there is limited knowledge about the architecture of Robotware. SLRT guarantees a FIFO behaviour to the reading function, data is analysed in order of arrival. Working asynchronously can lead to situations where the receiving machine reads its buffer while the transmitter does not send the complete message. On the RAPID side, there is a concern to avoid arbitrary packing and unpacking, by specifying a index in the array where the first byte is stored. This way, even working at different sampling frequencies, it is assured that IRC5 executing the RAPID program is synchronised with SLRT.

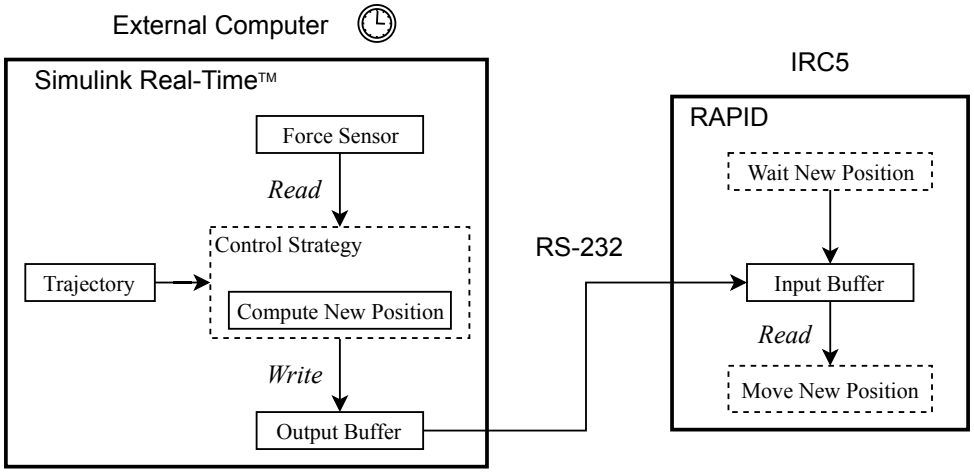


Figure 3.9: Remote Control Algorithm

The communication interface imposes that at each time sample, SLRT first reads the input buffer and then writes to the output buffer. Whereby a Simulink cycle consists in reading the input buffer and the force sensor, compute the new position based on the control strategy and write it to the output buffer. However, a force control action needs more computational effort and the input buffer would overload. Therefore, has been withdraw the communication from IRC5 to the external computer, it is no longer a bi-directional communication. Simulink no longer receives or reads the input buffer and the RAPID program no longer writes the actual position on the output buffer. This modification is not significant, since the actual position value sent to SLRT was not for control purposes but rather plotting data and that can be achieved directly on the SLRT. The RAPID cycle now reads new positions from the input buffer and moves to the new position. Both cycles are depicted in Figure 3.9.

Given the RobotWare version installed, the lack of drivers for other communication protocols and the low-access to IRC5 controller, this remote control algorithm is the only possible way to make the

communication interface between the two computers.

Chapter 4

Methods and Implementation

In this chapter is proposed a force-moment control strategy for close-ended industrial manipulators. The idea is to exploit the precision of the manipulator as a ”positioning device” with the use of external software and sensors. Nevertheless, the typical industrial controllers, like the standard IRC5, lack the tools for implementing force control strategies, which represents a serious challenge. This way, the main goal is to make the robot sensitive to contact forces so it can perform robotic machining and assembly tasks by endowing the existing robotic system with external sensors and software. Briefly within the approach here taken, the IRC5 industrial controller is responsible for the position control and the force-moment control strategy will accommodate the user defined trajectory with the data collected from the sensor. Position, orientation and force/ moment must be specified along each direction of the task frame and the control strategy rests on the combination of the following two modules:

- **Trajectory Planning** - responsible for the analytical trajectory between initial and target points, it generates a time sequence of values for the end-effector motion and orientation so the manipulator follows a geometrically specified path in space. User specifies initial and target parameters for both position/ orientation and time to describe the desired trajectory or defines the distance to be travelled with a average velocity;
- **Position-based Force/ Moment Control** - responsible for regulation to a desired constant force and moment, and tracking of a time-varying desired pose trajectory considering a non-minimal representation of the operational space (position and quaternion orientation) when there is contact between the robotic system and the environment

4.1 Trajectory Planning

In material handling tasks, it is sufficient to assign only the pick-up and release locations of an object, whereas, in machining or assembly tasks, the end-effector has to follow a desired trajectory (path motion). To this end, an operational space trajectory planning algorithm based on parametric curves [2] is proposed. The goal of the algorithm described in the following section is to generate the timing

laws for the end-effector operational space starting from a precise description of the desired motion and orientation.

Moving the manipulator is done by calling a `Move` function (see subsection 3.2.2) whose arguments are target destination, travelling speed, accuracy of motion and the type of interpolation. Meaning, the IRC5 controller features algorithms that interpolate paths between two points, and in what concerns path interpolation and latency, as already mentioned, `MoveAbsJ` is the fastest way to move the manipulator [6]. By leaving the trajectory planning for the IRC5 there would be no control or access to the interpolated points between the starting and final position.

If the end-effector motion has to follow a prescribed trajectory of position and/or orientation, this must be expressed analytically. It is then necessary to refer to motion primitives defining the geometric features of the path and time primitives defining the timing law on the path itself.

For the definition of path primitives it is convenient to refer to the parametric description of paths in space,

$$\boldsymbol{p} = \boldsymbol{f}(s). \tag{4.1}$$

Consider a path Γ represented by the parametric representation (4.1) and let $\boldsymbol{p}$ be a point corresponding to the arc length s. To each value of s a welldetermined path point corresponds, and then the arc length can be used as a parameter. Except for special cases, $\boldsymbol{p}$ allows the definition of three unit vectors characterizing the path (tangent, normal and binormal unit vectors). The orientation of such vectors depends exclusively on the path geometry, while their direction depends also on the direction induced by (4.1) on the path.

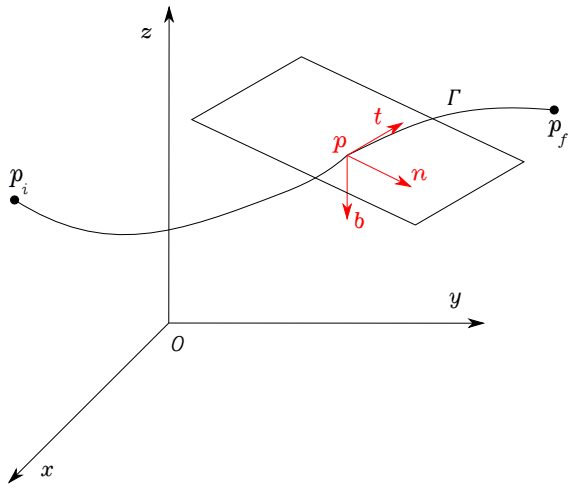


Figure 4.1: Parametric representation of a path in space [2]

It can be shown that the three unit vectors (Figure 4.1) are related by simple relations to the path representation Γ as a function of the arc length (4.2).

$$\begin{aligned} \boldsymbol{t} &= \frac{d\boldsymbol{p}}{ds} \\ \boldsymbol{n} &= \frac{1}{\left\| \frac{d^2\boldsymbol{p}}{ds^2} \right\|} \frac{d^2\boldsymbol{p}}{ds^2} \\ \boldsymbol{b} &= \boldsymbol{t} \times \boldsymbol{n}. \end{aligned} \tag{4.2}$$

Typical path parametric representations are reported below which are useful for trajectory generation in the operational space.

4.1.1 Rectilinear Path

For a rectilinear path, consider the linear segment connecting point $\boldsymbol{p}_i$ to point $\boldsymbol{p}_f$, the parametric representation of this path is

$$\boldsymbol{p}(s) = \boldsymbol{p}_i + \frac{s}{\|\boldsymbol{p}_f - \boldsymbol{p}_i\|}(\boldsymbol{p}_f - \boldsymbol{p}_i). \tag{4.3}$$

where $\boldsymbol{p}(0) = \boldsymbol{p}_i$ and $\boldsymbol{p}(\|\boldsymbol{p}_f - \boldsymbol{p}_i\|) = \boldsymbol{p}_f$. Hence, the direction induced on Γ by the parametric representation (4.3) is that going from $\boldsymbol{p}_i$ to $\boldsymbol{p}_f$. Differentiating (4.3) with respect to s gives

$$\frac{d\boldsymbol{p}}{ds} = \frac{1}{\|\boldsymbol{p}_f - \boldsymbol{p}_i\|}(\boldsymbol{p}_f - \boldsymbol{p}_i) \tag{4.4}$$

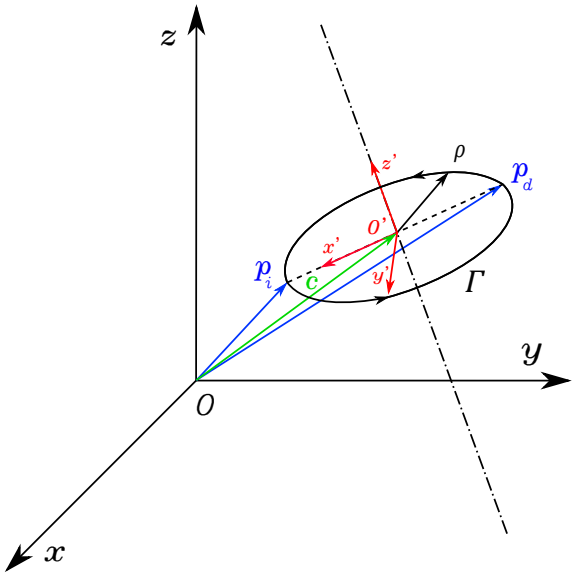


Figure 4.2: Parametric representation of a circle in space

4.1.2 Circular Path

For a circular path, consider a circle Γ in space. Before deriving its parametric representation, it is necessary to introduce its significant parameters. Suppose that the circle is specified by assigning

(Figure 4.2):

- the unit vector of the circle axis $\boldsymbol{z}'$,
- the initial position $\boldsymbol{p}_i$ and a point on the circle $\boldsymbol{p}_d$ that defines the circle diameter.

With these parameters, the position vector $\boldsymbol{c}$ of the centre of the circle can be found,

$$\boldsymbol{c} = \boldsymbol{p}_i + \frac{\boldsymbol{p}_d - \boldsymbol{p}_i}{2}. \tag{4.5}$$

The aim is to find a parametric representation of the circle as a function of the arc length. Consider the frame $O' - x'y'z'$, where O' coincides with the centre of the circle, axis x' is oriented along the direction of the vector $\boldsymbol{p}_i - \boldsymbol{c}$, axis z' is the characteristic circle plane vector (user defined) and axis y' is chosen so as to complete a right-handed frame (4.9). When expressed in this reference frame, the parametric representation of the circle as function of the arc length is

$$\boldsymbol{p}'(s) = \begin{bmatrix} \rho\cos(s/\rho) \\ \rho\sin(s/\rho) \\ 0 \end{bmatrix}, \tag{4.6}$$

where $\rho = \left\| \frac{1}{2}(\boldsymbol{p}_i - \boldsymbol{p}_d) \right\|$ is the radius of the circle and the point $\boldsymbol{p}_i$ has been assumed as the origin of the arc length. For a different reference frame, the path representation becomes

$$\boldsymbol{p}(s) = \boldsymbol{c} + \boldsymbol{R}\boldsymbol{p}'(s), \tag{4.7}$$

where $\boldsymbol{c}$ is expressed in the frame $O - xyz$ and $\boldsymbol{R}$ is the rotation matrix of frame $O' - x'y'z'$ with respect to frame $O - xyz$ which can be written as

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{x}' & \boldsymbol{y}' & \boldsymbol{z}' \end{bmatrix}, \tag{4.8}$$

where $\boldsymbol{x}', \boldsymbol{y}', \boldsymbol{z}'$ indicate the unit vectors of the frame expressed in the frame $O - xyz$ which are given by

$$\begin{aligned} \boldsymbol{x}' &= \frac{\boldsymbol{p}_i - \boldsymbol{c}}{\|\boldsymbol{p}_i - \boldsymbol{c}\|} \\ \boldsymbol{y}' &= \boldsymbol{z}' \times \boldsymbol{x}' \end{aligned} \tag{4.9}$$

Differentiating (4.7) with respect to s gives

$$\frac{d\boldsymbol{p}}{ds} = \boldsymbol{R} \begin{bmatrix} -\sin(s/\rho) \\ \cos(s/\rho) \\ 0 \end{bmatrix} \tag{4.10}$$

4.1.3 Orientation Path

End-effector orientation typically is specified in terms of the rotation matrix of the (time-varying) end-effector frame with respect to the base frame. The three columns of the rotation matrix represent the

three unit vectors of the end-effector frame with respect to the base frame.

An alternative way to generate a trajectory for orientation of clearer interpretation in the Cartesian space can be derived by resorting to angle and axis description. Let $\boldsymbol{R}_i$ and $\boldsymbol{R}_f$ denote respectively the rotation matrices of the initial frame $O_i - x_i y_i z_i$ and the final frame $O_f - x_f y_f z_f$, both with respect to the base frame. The rotation matrix between the two frames can be computed by (4.11) recalling that $\boldsymbol{R}_f = \boldsymbol{R}_i \boldsymbol{R}_f^i$ and $\boldsymbol{R}^T = \boldsymbol{R}^{-1}$.

$$\boldsymbol{R}_f^i = \boldsymbol{R}_i^T \boldsymbol{R}_f = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{4.11}$$

If the matrix $\boldsymbol{R}^i(t)$ is defined to describe the transition from $\boldsymbol{R}_i$ to $\boldsymbol{R}_f$, it must be $\boldsymbol{R}^i(0) = \boldsymbol{I}$ and $\boldsymbol{R}^i(t_f) = \boldsymbol{R}_f^i$. Hence, the matrix $\boldsymbol{R}_f^i$ can be expressed as the rotation matrix about a fixed axis in space; the unit vector $\boldsymbol{r}^i$ of the axis and the angle of rotation ϑ_f can be computed by

$$\vartheta_f = \cos^{-1}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right) \tag{4.12}$$

$$\boldsymbol{r}^i = \frac{1}{2\sin\vartheta_f} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \tag{4.13}$$

for $\sin\vartheta_f \neq 0$.

The matrix $\boldsymbol{R}^i(t)$ can be interpreted as a matrix $\boldsymbol{R}^i(\vartheta(t), \boldsymbol{r}^i)$ and computed via (A.3); it is then sufficient to assign a timing law to ϑ with $\vartheta(0) = 0$ and $\vartheta(t_f) = \vartheta_f$, and compute the components of $\boldsymbol{r}^i$ from (4.12). Since $\boldsymbol{r}^i$ is constant, the resulting velocity and acceleration are respectively

$$\boldsymbol{\omega}^i = \dot{\vartheta}\boldsymbol{r}^i \tag{4.14}$$

$$\dot{\boldsymbol{\omega}}^i = \ddot{\vartheta}\boldsymbol{r}^i. \tag{4.15}$$

Finally, in order to characterize the end-effector orientation trajectory with respect to the base frame, the following transformations are needed:

$$\boldsymbol{R}_e(t) = \boldsymbol{R}_i \boldsymbol{R}^i(t) \tag{4.16}$$

$$\boldsymbol{\omega}_e(t) = \boldsymbol{R}_i \boldsymbol{\omega}^i(t) \tag{4.17}$$

$$\dot{\boldsymbol{\omega}}_e(t) = \boldsymbol{R}_i \dot{\boldsymbol{\omega}}^i(t) \tag{4.18}$$

4.1.4 Timing Law

The timing law for the s coordinate is defined using a third-order polynomial to ensure no discontinuity both in position and velocity.

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{4.19}$$

$$\dot{s}(t) = a_1 + 2a_2 t + 3a_3 t^2 \tag{4.20}$$

Equations (4.19) and (4.20) describe the cubic polynomial timing law. Since four coefficients are available, it is possible to impose initial and target positions as well as velocity constraints for the same points, usually set to zero.

$$\begin{cases} s(t_i) = 0 \\ s(t_f) = s_f \\ \dot{s}(t_i) = \dot{s}(t_f) = 0 \end{cases} \Leftrightarrow \begin{cases} a_0 = \frac{3t_f t_i^2 - t_i^3}{(t_f - t_i)^3} s_f \\ a_1 = -\frac{6t_f t_i}{(t_f - t_i)^3} s_f \\ a_2 = \frac{3(t_f - t_i)}{(t_f - t_i)^3} s_f \\ a_3 = -\frac{2}{(t_f - t_i)^3} s_f \end{cases} \tag{4.21}$$

Coefficients a_0, a_1, a_2, a_3 are obtained solving the boundary equations defined in (4.21). The same logic is applied for each motion primitive stated previously, since the boundary constraints only differ in the value of $s(t_f)$. For the rectilinear and circular path $s(t_f)$ is given, respectively, by (4.22) and (4.23), for the orientation the timing law is made for the angle ϑ and the final boundary condition is (4.24).

$$s(t_f) = \left\| \boldsymbol{p}_f - \boldsymbol{p}_i \right\| \tag{4.22}$$

$$s(t_f) = 2\rho\pi \tag{4.23}$$

$$s(t_f) = \vartheta_f \tag{4.24}$$

The formulation described above can be set to a distance with average velocity instead of target position and time resorting to the velocity equation.

Since a path and a trajectory have been specified in the operational space in terms of $\boldsymbol{p}_e(t)$ and $\boldsymbol{R}_e(t)$, inverse kinematics techniques can be used to find the corresponding trajectories in the joint space $\boldsymbol{q}(t)$.

4.2 Position-based Force/ Moment Control

In this section the main concern is to develop a force-moment control strategy that can easily be realized in commercial industrial robots endowed with standard industrial controllers that are exclusively position controlled, as the IRB140 manipulator and IRC5 controller used for the present work. The integration of a force-moment control strategy with the closed-ended IRC5 industrial controller represents the major effort of this thesis, since this industrial controller was not designed to have a force control mode.

Therefore, the conditions imposed on this approach, keeping in mind the work goals, are:

- Manipulator's movement in free motion situations shall not be affected by the introduction of the control action,
- The control law must be simple in a way that can be used in real time,
- No changes should be made to the original controller implying hardware modifications and there is no access to low-level layers of the software.

Accordingly with the literature review, it is required a controller that takes advantage of the manipulator position controller, so position-based force control is best suited. A controller of this kind, rests on the robot dynamic parameters update and optimized kinematic and dynamics models present in standard industrial controllers to achieve high precision and repeatability on position control.

As the name suggests and as already mentioned, position-based control only concerns position and force control with the implementation of an external force control loop around the position controller. The manipulator's position is accommodated with force data from a force/ moment sensor. However, the main goal of this thesis is to implement a force-moment control on a standard industrial manipulator. Therefore, following the same line of thought in force control, an outer orientation-moment control loop will be implemented around the manipulator position controller - reason to be called position-based force/ moment control.

A generic block diagram of this controller is represented in Figure 4.3, where the force/ moment control generates position and orientation corrections to the planned trajectory then sends them as joint coordinates to the IRC5 position controller resorting to inverse kinematics.

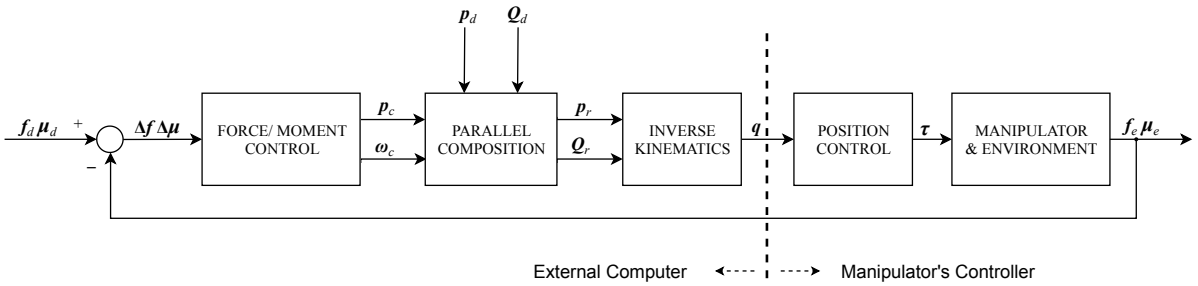


Figure 4.3: Generic block scheme of position-based force/ moment control

The subscript d denotes the desired values, the subscript c denotes the deviation resulting from force-moment controller action and r denotes the reference frame to be tracked and its position $\boldsymbol{p}_r$ and orientation $\boldsymbol{Q}_r = \{\eta_r, \boldsymbol{\epsilon}_r\}$ are computed through the parallel composition as it will be detailed on the following controller subsections.

The resemblance from parallel control and external force control are notorious. In a broad sense, it is like a parallel control implemented on the basis of a position-based force control, that is, an external force control loop without adopting a selection mechanism to discriminate between constrained and unconstrained motion, with a parallel composition that considers the force-moment control actions and

the desired values for position and orientation but laying the responsibility of position control on to the manipulator's controller (see Chapter 2).

The typical inner motion control on a position-based force control is compacted inside the position control block, since the proposed controller performs inverse kinematics outside that loop, as already mentioned, motion control is performed on the joint space. The force control part of the controller is responsible for accommodating manipulator's motion concerning the first three joint variables (q_1, q_2, q_3), while the moment-orientation control part is responsible for the last three joint variables (q_4, q_5, q_6). Thereby, force and moment-orientation control actions will act, respectively, on the translational (position) and rotational part of the desired trajectory.

Starting with the bullet points designated on this approach, when the manipulator is moving in free motion (when there is no contact with the environment), and the desired force and/ or torque are equal to zero, the manipulator's position control will move the robot to reference trajectory points received, as seen on the following controller subsections. Sensor noisy readings are too low for the controller to act on, force and moment errors assume null values. However, when the desired force and/ or torque are not zero but the manipulator is in free motion, the force-moment controller will act on the difference between the desired and measured values and the trajectory will have a deviation from the desired one due to that difference. To ensure that the manipulator has a similar behaviour in free motion when the desired force and/ or torque are not zero a finite state-machine is implemented.

The FSM, named Force-moment state-machine, is responsible for designating how the manipulator is controlled (position or force controlled) depending on force/ moment error values. In a broad sense, it will define when the force-moment controller takes action in addition to plan the trajectory to execute a given task.

Both force and moment control laws will be defined in a classical and simple perspective, as a PI controller in sense of achieving null stationary error (force and torque) and avoiding the differential component (D) due to noisy readings from the force/ moment sensor. The stability of the controller is guaranteed if every pole lies on the left half-plane of $s-$plane and if the controller bandwidth is smaller than the robot motion controller bandwidth, which is a necessary condition for cascade control, the inner loop must be faster than the outer loop. Notwithstanding, open-loop stability of the coupled system is guaranteed given the robotic system is stable. Force and moment control actions can be designed on the basis of a simplified model of the environment (see Figure 4.4). Assuming $\boldsymbol{k}_s >> \boldsymbol{k}_e$ and $\boldsymbol{c}_s = \boldsymbol{c}_e \simeq 0$, since they can be neglected for cases where small velocities are used. Therefore, contact is modelled by a linear string.

Position-based force/ moment control action will be explained through 2 separated controllers (each control action individually) in the following subsections without considering the force-moment state-machine. Afterwards, in section 4.3 is presented the hardware-software architecture to integrate both control actions and the force-moment state-machine, that includes the trajectory planner, with the IRC5 industrial controller. This architecture is based on the experimental setup described in Chapter 3.

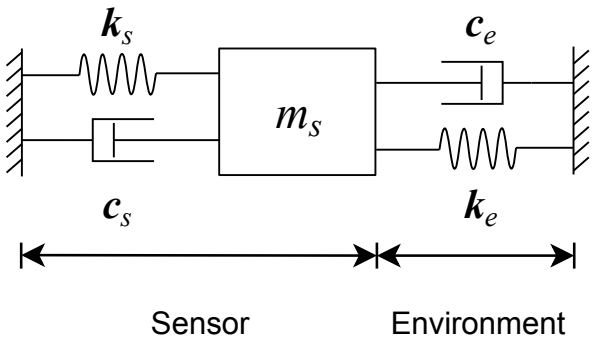


Figure 4.4: Simplified sensor and environment model [29]

4.2.1 Force Control

The idea behind the force control part of the position-based force/ moment control is to make the robot sensitive to contact forces on end-effector level providing position references (in the form of position accommodation) to the manipulator's position control system. The manipulator will be given the ability to 'feel' its surroundings, which is imperative to perform accurate machining tasks.

In detail, the position-based force control developed is depicted in Figure 4.5. The outer loop containing the force controller is the kernel of the control strategy, it has a twofold role: adjustment and force tracking. Force control block will adapt the manipulator's motion given the force sensed by the FTS and will track the desired force so it can be imposed to the environment surface.

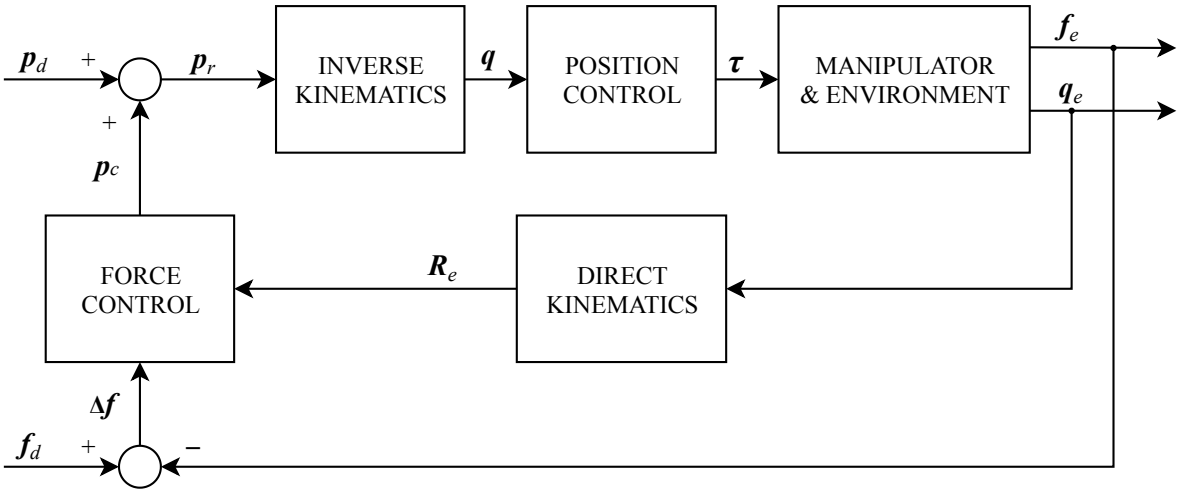


Figure 4.5: Block scheme of the position-based force control implemented

As one can see, the input of the force controller is the difference between the desired and actual contact force ($\Delta \boldsymbol{f} = \boldsymbol{f}_d - \boldsymbol{f}_e$) and a PI controller is considered so there would be no force error. Therefore, the force control law is given by

$$\boldsymbol{u}_f(t) = \boldsymbol{K}_{P_f} \Delta \boldsymbol{f}(t) + \boldsymbol{K}_{I_f} \int_0^t \Delta \boldsymbol{f}(\tau) d\tau, \tag{4.25}$$

where $\boldsymbol{u}_f$ is the displacement due to the error between the desired and sensed force, K_{P_f} and K_{I_f} are, respectively, the proportional and integral gains from the controller.

For the tuning process it is worth mentioning, the bigger K_{I_f}, the closer the zero is from the origin and lesser will be the integral action on the error. On the other hand, the bigger K_{P_f} the fastest and more oscillatory the system will be.

Position accommodation is defined as the parallel composition

$$\boldsymbol{p}_r = \boldsymbol{p}_d + \boldsymbol{p}_c \tag{4.26}$$

where $\boldsymbol{p}_r$ is the position of the reference frame to be tracked, $\boldsymbol{p}_d$ is the desired position and $\boldsymbol{p}_c$ is the displacement due to force contact computed by the force controller with respect to reference base frame. Since the controller acts on the force imposed by the environment on the manipulator, a premultiplication with the rotation matrix from direct kinematics for end-effector actual orientation ($\boldsymbol{R}_e$) transforms the resultant displacement from the sensor frame to the manipulator base reference frame. When there is contact with the environment, $\boldsymbol{p}_r$ represents the position of the compliant frame (see section 2.2.2) with respect to the base reference frame.

For better understanding, one considers the following hypothetical situation: is desired for the robot to apply a null force/ pressure on a surface ($\boldsymbol{f}_d = \boldsymbol{0}^T$) and the noisy sensor readings are considered equal to zero ($\boldsymbol{f}_e \simeq \boldsymbol{0}^T$).

When the manipulator is moving in free motion the force controller input is equal to zero ($\Delta \boldsymbol{f} \simeq \boldsymbol{0}^T$). Accordingly, the output of the force control law will be also equal to zero ($\boldsymbol{p}_c \simeq \boldsymbol{0}^T$). Force control action will be null, there would be no motion correction and the manipulator will perform the planned trajectory. On the other hand, when the manipulator's tool enters in contact with the environment, the force controller input will be no longer equal to zero and the controller will generate the necessary position deviation for the force imposed to the surface be the desired, even at the expense of a position error and highlighting the dominance of force control over position control.

4.2.2 Moment and Orientation Control

The idea behind the moment-orientation control part is similar to the counter force part but concerning moment and orientation, that is, regulation to a constant desired moment and tracking of a time-varying desired orientation trajectory considering a geometrical approach where orientation displacements are described in terms of unit quaternions. So, instead of external force control loop contains an outer loop with a moment/ orientation controller, which provides orientation references (in the form of orientation accommodation) to the manipulator's position control system. Apart from improving the ability to 'feel' given by the force part by considering the constraints imposed by the environment on the end-effector orientation, it will allow the manipulator to perform assembly tasks.

The strategy presented above for force control can be conceptually pursued also for moment and orientation control, as shown in Figure 4.6. The outer loop containing the moment controller and the parallel composition is the core of the control strategy, since the output of moment control block is a angular velocity and the trajectory planned will be a quaternion orientation.

As one can see, the input of the moment controller is the difference between the desired and contact

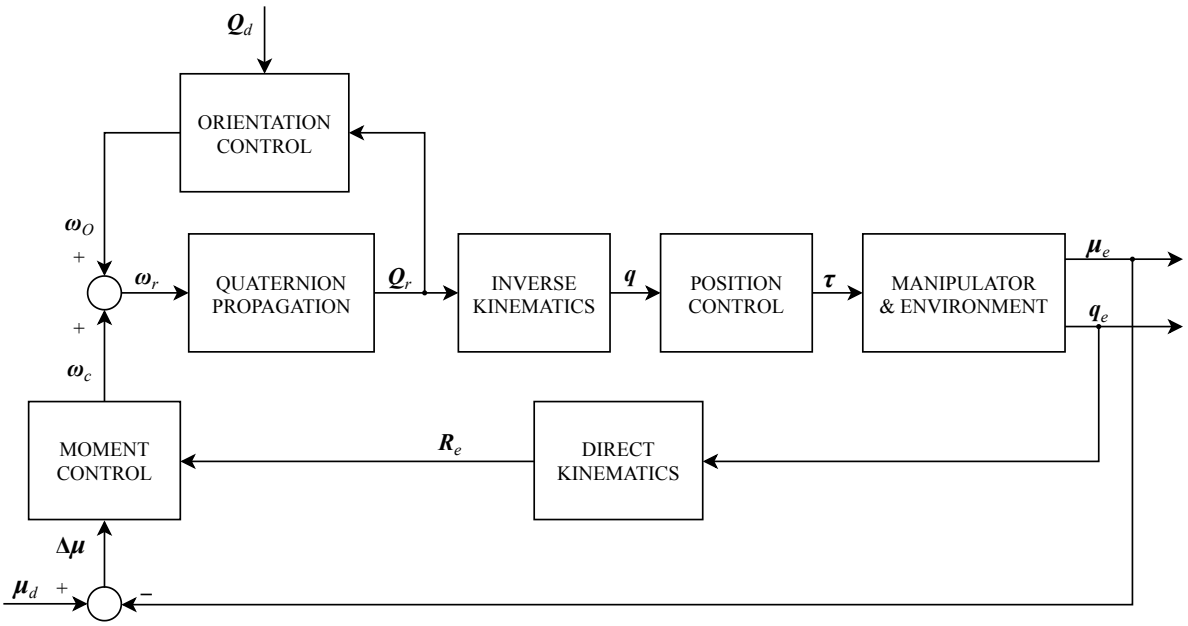


Figure 4.6: Block scheme of the position-based moment control implemented

moment ($\Delta\boldsymbol{\mu} = \boldsymbol{\mu}_d - \boldsymbol{\mu}_e$) and a PI controller is considered since it is undesirable to have stationary error. Therefore, the moment control law is given by

$$\boldsymbol{u}_m(t) = \boldsymbol{K}_{P_m}\Delta\boldsymbol{\mu}(t) + \boldsymbol{K}_{I_m}\int_0^t \Delta\boldsymbol{\mu}(\tau)d\tau, \tag{4.27}$$

where $\boldsymbol{u}_m$ is the angular velocity due to the error between the desired and sensed moment, K_{P_m} and K_{I_m} are, respectively, the proportional and integral gains from the controller.

Orientation accommodation is defined as the parallel composition

$$\boldsymbol{\omega}_r = \boldsymbol{\omega}_c + \boldsymbol{\omega}_O \tag{4.28}$$

where $\boldsymbol{\omega}_r$ is the angular velocity of the reference frame to be tracked, $\boldsymbol{\omega}_O$ is the angular velocity necessary to rotate the reference frame r to the desired frame and $\boldsymbol{\omega}_c$ is the angular velocity resulting from the contact moment computed by the moment controller with respect to reference base frame. Since the controller acts on the moment imposed by the environment on the manipulator, a premultiplication with the rotation matrix from direct kinematics for end-effector actual orientation ($\boldsymbol{R}_e$) transforms the resultant angular velocity from the end-effector's frame to the manipulator base reference frame.

Since the inverse kinematics has an orientation as input, a quaternion is computed from the resultant angular velocity due to contact moment ($\mathcal{Q}_c$). However, for the planned orientation trajectory to be considered the orientation difference between reference and desired must be computed. This difference is achieved through a quaternion error ($\boldsymbol{e}_O$) and then expressed as an angular velocity with respect to the base frame (ω_O) for the parallel composition (4.28). The same line of thought to compute a angular velocity from a quaternion error can be seen on CLIK (closed-loop inverse kinematics) [30].

The equations that rule the orientation control block on the above diagram are given by

$$\boldsymbol{\omega}_O = \boldsymbol{K}_O \boldsymbol{e}_O \tag{4.29}$$

$$\boldsymbol{e}_O = \eta_r \boldsymbol{\epsilon}_d - \eta_d \boldsymbol{\epsilon}_r - S(\boldsymbol{\epsilon}_d)\boldsymbol{\epsilon}_r, \tag{4.30}$$

where $\boldsymbol{\omega}_O$ is the angular velocity error between reference and desired frames with respect to the base reference frame, K_O is a suitable positive definite matrix gain and $\boldsymbol{e}_O$ is the quaternion error between reference and desired frames.

It is worth pointing out that the computation of the quaternion from the angular velocity due to the contact moment gives rise to an integral action. The quaternion $\mathcal{Q}_r$ is computed from $\boldsymbol{\omega}_r$ by integrating the propagation equations (A.1a-A.1b) with initial conditions $\mathcal{Q}_r(t=0) = \mathcal{Q}_u$, being $\mathcal{Q}_u$ the quaternion describing the orientation of the undeformed frame with respect to the base frame.

For better understanding, one considers the following situation: for the robot to maintain a desired orientation the moment to be imposed on the environment must be null ($\boldsymbol{\mu}_d = \mathbf{0}^T$) and the noisy sensor readings are considered equal to zero ($\boldsymbol{\mu}_e \simeq \mathbf{0}^T$).

When the manipulator is moving in free motion the moment controller input is equal to zero ($\Delta\boldsymbol{\mu} \simeq \mathbf{0}^T$). Accordingly, the output of the moment control law will be also equal to zero ($\boldsymbol{\omega}_c \simeq \mathbf{0}^T$). Moment control action will be null and from the orientation control will come a null angular velocity ($\boldsymbol{\omega}_O = \mathbf{0}^T$), since reference and desired frames are aligned and the error between them it will be $\mathcal{Q}_O = \{1, \mathbf{0}\}$. There would be no motion correction and the manipulator will perform the planned orientation trajectory. On the other hand, when the manipulator's tool comes in contact with the environment, the moment controller input will be no longer equal to zero. The controller will generate the necessary orientation deviation for the moment imposed on the surface may be the desired, even at the expense of an orientation error, which as in the force part, highlights the dominance of moment control over orientation control.

4.2.3 Force-Moment State-Machine

The Force-moment state-machine plays an important role on the control strategy developed, due to its responsibility for planing the task execution through a Finite State-Machine (FSM), which defines how the manipulator is controlled hinging on the contact with the environment.

The idea behind it is to provide the inputs to the controller depending on the force/ moment error. Assuming that they are exerted by manipulator on the environment surface with reference to the end-effector frame. Moreover, the goals are to avoid the unnecessary activation of the force-moment control action when the manipulator is in free motion and the force/ moment peak value, which rises when the manipulator contacts with environment. Therefore, it was defined three thresholds for the force/ moment error ($\Delta\boldsymbol{h} > 0$, $\Delta\boldsymbol{h} = 0$ and $\Delta\boldsymbol{h} < 0$) that define in which task phase the manipulator is: free motion, non-contact to contact transition with the environment and the machining/ assembly task itself. Given this, the state machine developed runs as depicted in Figure 4.7, where $\Delta\boldsymbol{h}$ represents the error between desired and desired force-torque vector ($\Delta\boldsymbol{h} = \boldsymbol{h}_d - \boldsymbol{h}_e$) with $\boldsymbol{h}_i = [\boldsymbol{f}_i^T \quad \boldsymbol{\mu}_i^T]$.

As one can see, the state machine is composed by four states: Approach, Force Stabilization, Task

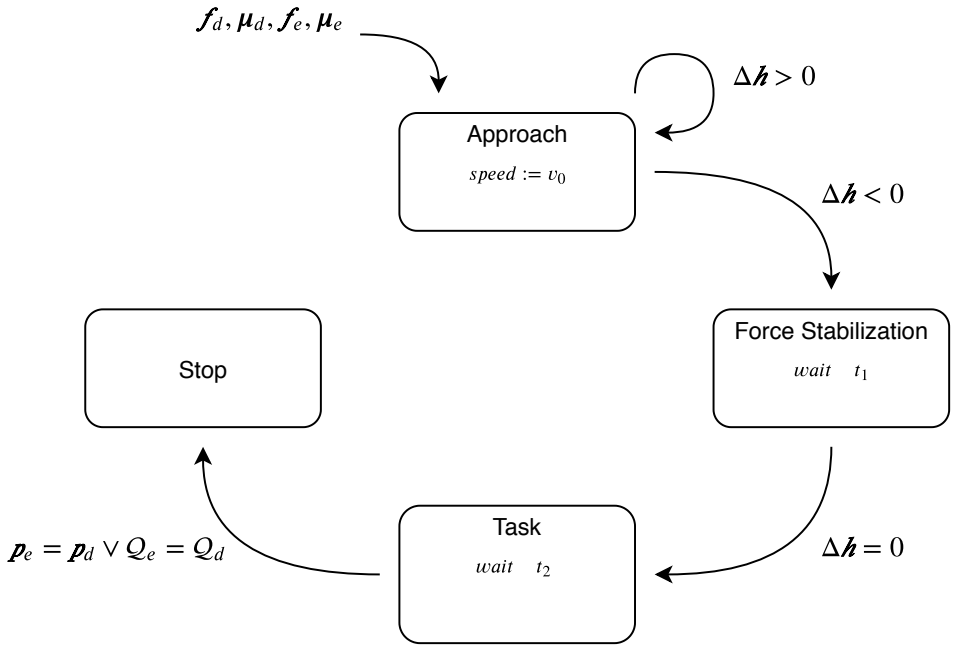


Figure 4.7: Force-moment state-machine

and Stop, and receives as input the desired and sensed force/ moment values. Thus, most of the transitions between states are defined by the error between those values. While $\Delta \boldsymbol{h} > 0$, meaning there is no contact with the environment, Force-moment state-machine is accountable to make the manipulator perform an approach to the material, whose location does not to be known at a user defined speed (constant or profile shaped). At this point, the robot is only position controlled and the "Approach" state designs the trajectory to be followed (as in Section 4.1). Though, the robotic system is still sensitive to forces and moments, it is in fact when the error decreases below zero ($\Delta \boldsymbol{h} < 0$) that occurs state transition. The manipulator goes from a non-contact scenario to contact with a given surface and it becomes force controlled with the objective to track the desired force/ moment. This way, a stabilization on that value is made by holding end-effector position and orientation right before the contact occurred (characterized by a peak on the force or moment), that is, the last manipulator pose in the previous state. Controller action takes place after t_1 seconds to let the FTS readings normalize and consequently avoid misleading position and/ or orientation accommodations. Hence, as long as the error is not null the manipulator will accommodate its pose until the desired force/ moment is not obtained.

When $\Delta \boldsymbol{h} = 0$ and after t_2 seconds conditions are met for the task itself to be performed. The manipulator continues force controlled and will start running the task from the pose where the error was null. Within Task state there is also a user defined trajectory planner that specifies the desired pose for a given machining/ assembly task. Until the manipulator reaches the desired position or orientation the task is not complete.

The behaviour of the state-machine above described results in a simple way to validate the force-moment controller, overcoming the necessity to use this type of controllers when the contact is already established. Its methodology can be adapted to more complex environments and tasks, since there is no

transitions to previous states. As a matter of fact, if the error remains above zero the manipulator moves in free motion and stays position controlled performing a planned trajectory. However, when contact is made with the environment a task will be performed until a desired pose is obtained.

Note that this structure, although suitable for most machining tasks (involving the manipulator to apply pressure on a surface), it is not suitable for every assembly task. However, it can be implemented differently given the case as long as sensor signals and transition conditions are consistent.

4.3 Hardware-Software Architecture

In this section is detailed the hardware-software architecture used to integrate the force-moment control strategy described throughout this chapter with the industrial IRC5. Since both software and hardware used were already described in Chapter 3, the following section focus on how the control strategy is split between the two computers: external computer and the IRC5 controller.

As mentioned before, the starting basis of the present thesis is to exploit the remote control algorithm developed in previous works in ACCAAI laboratory with external software and sensors, in a way that force-moment control strategies for robotic machining or assembly tasks can be integrated with a traditional close-ended IRC5 controller.

While defining the hardware-software architecture, the objective was to implement the maximum number of components necessary to the task realization directly in the external computer. In that way, the motion controller precision present in IRC5 could be exploited and the basis of the remote control application developed by [6] could be kept unaltered. Given the difficulties experienced in synchronizing both computers in reading, writing and sending data through the serial port, there was a risk of obtaining lower bandwidth with the increasement of exchanged data. Keeping this in mind, trying to make the most with its strengths and limitations, the strategy is split between the two computers as depicted in Figure 4.8. Note that the communication between the IRC5 controller and external computer shown is performed via RS-232 at a rate of $33Hz$, the best that can be achieved for the IRC5 used.

As one can see, Force-moment state-machine provides the controller inputs in the first three states, even when the manipulator is approaching the material, which at programming level, corresponds to null input values on the force and moment error variables. This way, as detailed in section 4.2, the controller will act as a block with unitary transfer function and the manipulator will only be position controlled. Both ”Approach” and ”Task” states have their own trajectory planner with the same purpose of providing the desired pose to the controller.

By using a trajectory planner on the external computer, the trajectory can be discretized in intermediate points. Smaller sub-paths are created and the analytical trajectory between initial and target points is specified. Therefore, the ability of the manipulator’s controller to define the trajectory between the given points is taken. This way, the manipulator’s interpolation by the `Move` function will be done on intermediate points in the path. This implementation was not only designed to avoid controller’s interpolation in assigning the intermediate points but also to enable the action from the force-moment controller implemented on a smaller distance. If the trajectory was only defined by two points and interpolated by

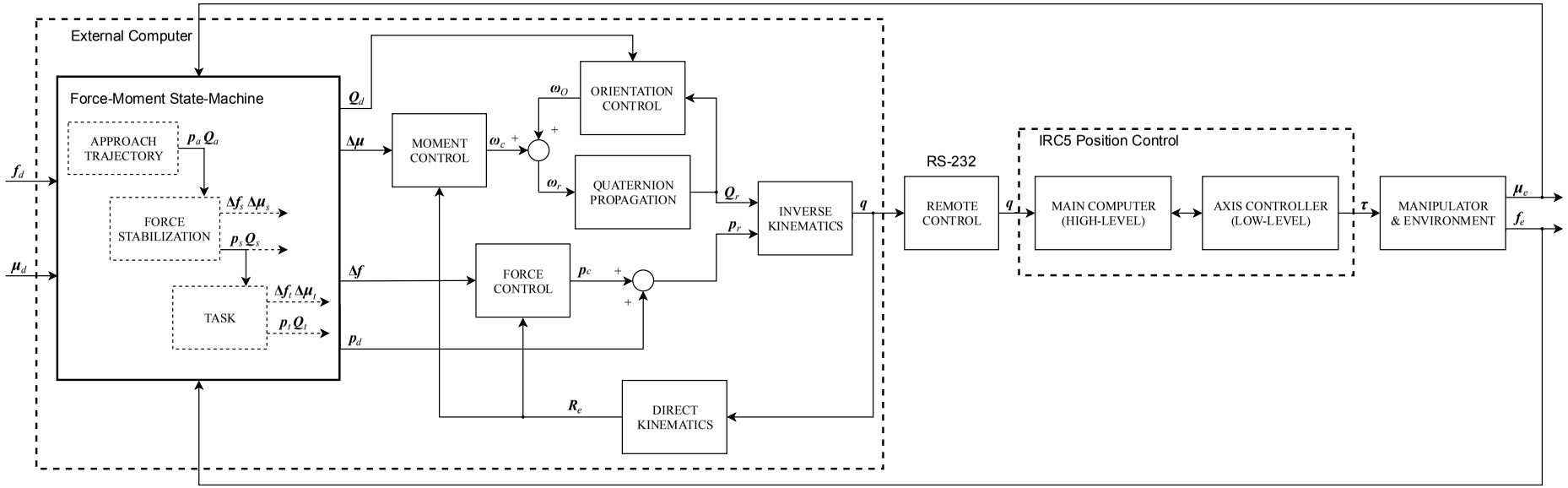


Figure 4.8: Hardware-software architecture

the `Move` function between them, the data received by the FTS could only be used in those two points. Since there is no access to IRC5 main computer trajectory planner, the information of the sensor could not be used to accommodate the manipulator's motion between points. Trajectory discretization will allow the force-moment controller to act in each trajectory point given the data received by the FTS.

Sampling rate has a direct influence on the trajectory discretization, the sub-paths are smaller the higher the sampling rate. As the sampling frequency of the communication between the external computer and the IRC5 is fixed at $33Hz$, the algorithm will generate an intermediate point at each $0.03s$. Therefore, the system is limited to run in small velocities since the time between initial and final points of the trajectory is chosen so that the manipulator has a smooth behaviour considering the travelled distance, the environment and the force-moment control action. When in free motion, between intermediate points the manipulator will run on the maximum speed possibly in $0.03s$ for the given sub-path distance, to ensure that the manipulator reaches every point in time.

The proposed controller performs inverse kinematics in the external computer, different from typical position-based force control Figure 2.4, since the RAPID function `MoveAbsJ` has the joint coordinates as input, which was chosen for instilling a better performance to the controller compared to the others functions available. The unpredictability presented in this function is overcome by reducing considerably the movement distance to a point where it will be linear, which is achieved by the analytical trajectory planner.

Since the manipulator is being remotely controlled it will only move when a new joint position is sent by the external computer. Which means that there is no need for the IRC5 to send its pose to the external computer, communication can be unidirectional (see subsection 3.2.3). When the joint coordinates are sent to IRC5 they are also used to perform the direct kinematics.

Chapter 5

Results

In this chapter are presented results concerning the practical implementation of the control strategy proposed in chapter 4 where the position-based force/ moment controller will be validated through several experiments. The tests were thought considering the functions in the Force Control module for IRC5 [31] and trying to resemble real industrial tasks. In the following the controller is implemented separately, since they can be applied independently and generally moment control is used in assembly tasks while force control in machining tasks.

5.1 Experiment 1: Force Control - Pressure

The first experiment intends to be representative of a machining task where a specific pressure is desired to be applied against a surface, like polishing and grinding. The manipulator will be given the ability to 'feel' its surroundings and follow the environment surface and impose a certain pressure against it. Which means that the robot will change its position in order to apply a constant force on a surface, even if the surface is not known. Moments and forces on x and y directions are neglected since they are not relevant to the experiment.

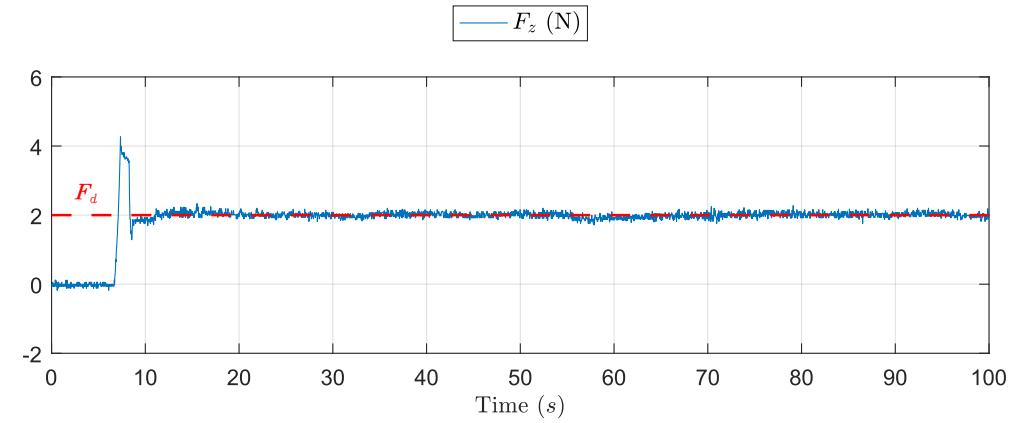
Besides testing the force controller this experiment also tests the transitions of Force-moment state-machine regarding the force error along z_e axis. Moreover, in order to illustrate the ability of the controller to act given any desired position or trajectory, the manipulator performs a circular path.

Controller gains were selected through simulation tunning and adapted experimentally due to remote control bandwidth and task execution speed. The force controller gains used are,

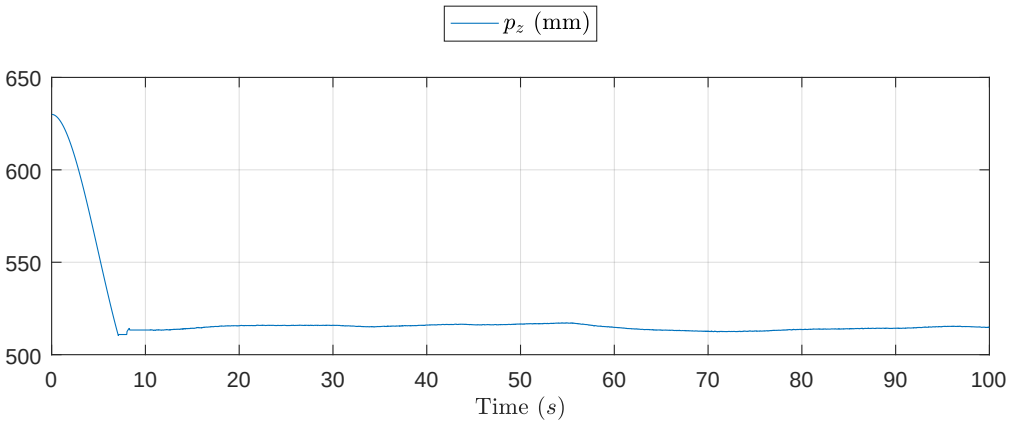
$$\boldsymbol{K}_{P_f} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \boldsymbol{K}_{I_f} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 5 \end{bmatrix} \tag{5.1}$$

The main results of this experience are depicted in Figure 5.1 and some snapshots taken at meaningfully moments are shown in Figure 5.2.

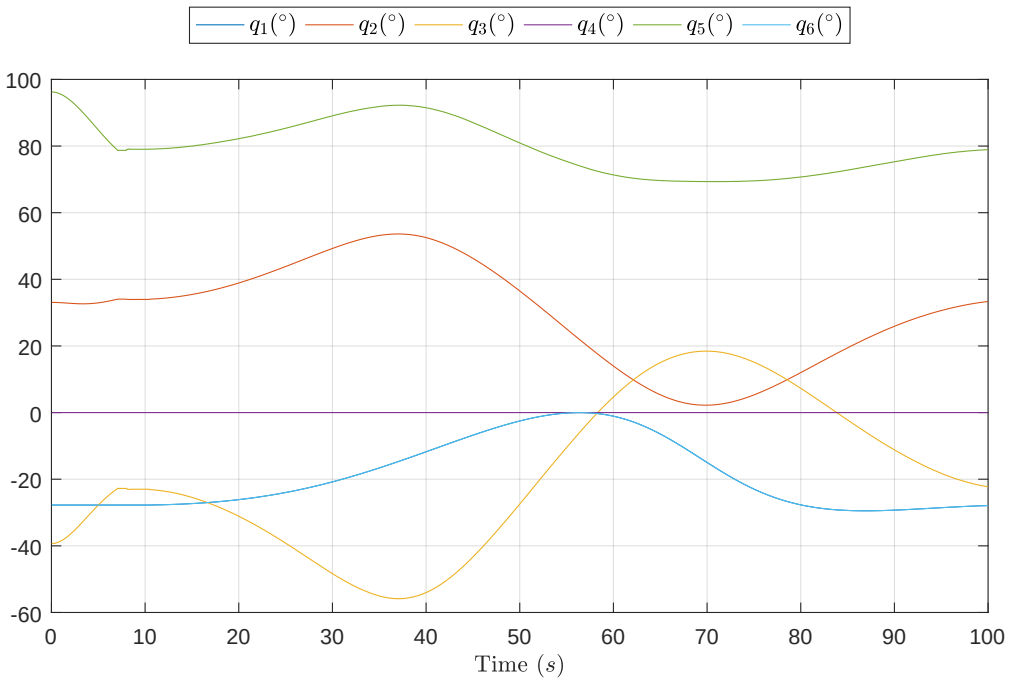
From Figure 5.1, it is clear that the manipulator accommodates its position along z axis to impose a



(a) Force along z_e axis



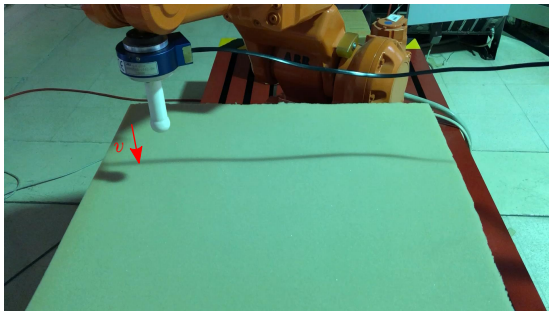
(b) TCP position with respect to base frame



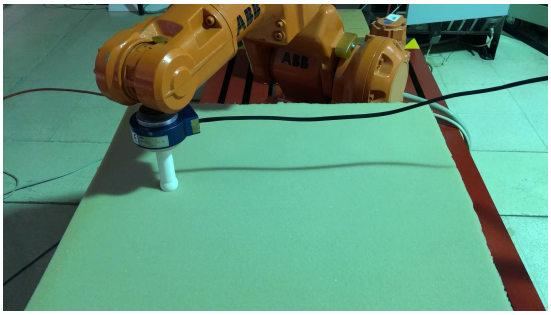
(c) Manipulator joint coordinates

Figure 5.1: Main results of the first experiment

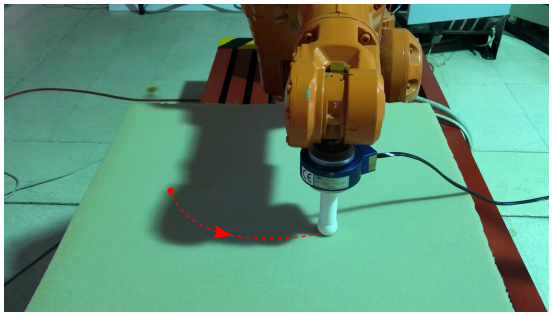
desire force of $2N$ on the surface (in the negative direction of z axis) while it is position controlled along x and y axis. When there is no contact with the environment the manipulator executes an approaching trajectory and is not force controlled. This situation can be observed during the time interval in which the sensed force is approximately zero. However, when the surface is felt at time instant $t = 7.11s$ (see snapshot of Figure 5.2(b)), the force hits a peak and exceeds the desired force causing the force error value to be below zero. The manipulator becomes force controlled until the task completion. Priori to material detection, whose location it is unknown, the robot starts to perform a downward movement in Figure 5.2(a) with a parabolic shaped velocity profile (due to the third-order polynomial on the timing law) with an average value of $10mm/s$.



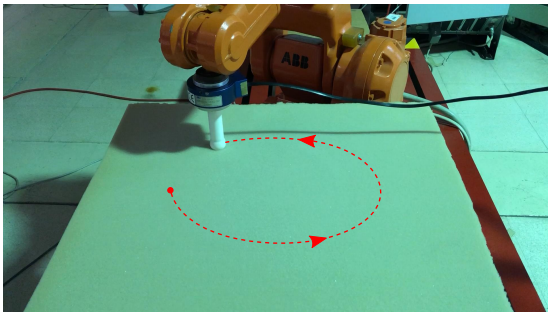
(a) $t \approx 0s$



(b) $t \approx 8s$



(c) $t \approx 49s$



(d) $t \approx 81s$

Figure 5.2: Snapshots of the first experiment

Immediately after the surface is felt, the manipulator stops its movement and holds the pose where the error value decreased below zero ($\Delta \boldsymbol{f} < 0$) to avoid damaging the tool and let the sensor readings normalize. The robotic system enters on the ”Force Stabilization” state, where force control action takes place $1s$ after the contact so that the system can avoid leading the manipulator to any misleading position during the sensor reading normalization. In this period forces sensed are ignored (real time system). The manipulator stays in that state until $t = 10.5s$, when the transition condition is true ($\Delta \boldsymbol{f} = 0$). However, due to the system $0.03s$ sampling rate, the robotic system can not achieve a position that nullifies the force error. The condition value was adapted to a interval where the error is considered zero. But only after the error values are consecutively within the interval (with the help of a counter) the condition is considered true.

Around $t = 11.4s$ the manipulator starts the task itself and performs a circular trajectory on the xy plane with a diameter of $300mm$ and an average velocity of $5mm/s$ from the position where force error was considered null. Figures 5.2(c) and 5.2(c) depicts two different time instances on the circular

path. In Figure 5.1(c) the three stages of the pressuring task, hence the three states of FSM, can be observed. Within $t \in [0, 7.11]$ the robotic system approaches the surface material, afterwards until $t = 10.5$ manipulator's pose stays almost constant since the force is stabilizing to the desired force value and the accommodations are small. From $t = 11.4s$ (there is a wait time of $0.9s$ between states) to $t = 100s$ is performed a circular path at a slower speed than the approach state. Note that despite of being a translational movement the desired TCP orientation is kept constant and perpendicular to the surface, therefore not all joints responsible for the orientation have constant values as the case of joints q_5 and q_6, which will move together with joint q_1.

5.2 Experiment 2: Force Control - Speed Change

The goal of the second experiment is to evaluate the effectiveness of the controller to change the task execution velocity due to excessive process forces in the opposite direction of the movement, when the direction of the movement is both position and forced controlled.

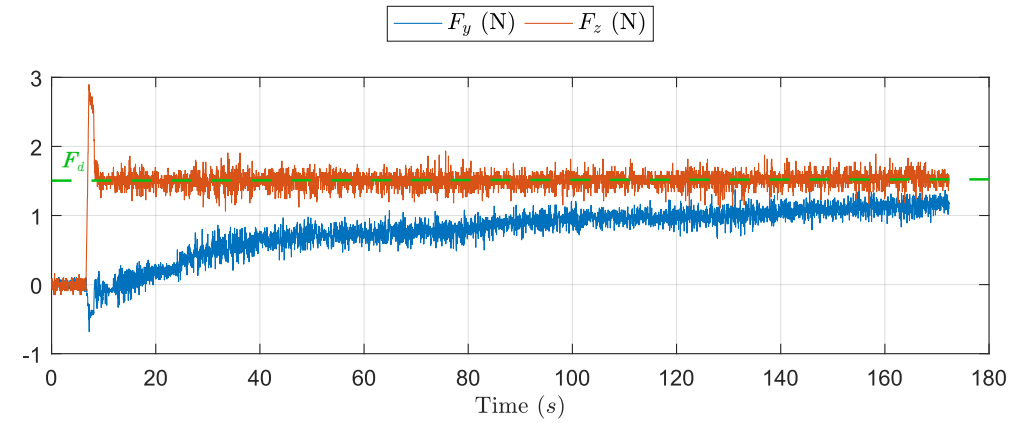
This experiment runs with transition conditions on Force-moment state-machine regarding force error along z_e-axis since y direction is both position and force controlled. The manipulator is performing a rectilinear trajectory on the $y-$axis while imposing a pressure of $1.5N$ on the surface (positive direction of z_e-axis). This means that the robot will change its position in order to apply a constant force on a surface perpendicularly to the movement and adapting the task execution speed given the F_y values, even if the surface is not known. Machining tasks like deburring and polishing with non regular surfaces or cutting processes are encompassed in this test, although a cutting process could be interpreted differently and alternatively the manipulator would be position controlled on the axis x and z and both position and force controlled along $y-$axis. Forces along the x-axis and moments in all directions are not relevant for this application and will not be considered.

Controller gains were selected through simulation tunning and adapted experimentally due to remote control bandwidth and task execution speed. The force controller gains used are,

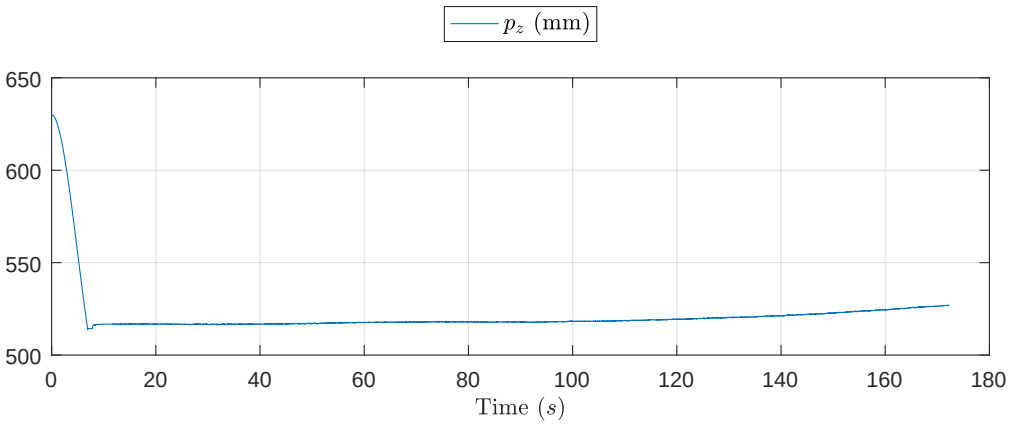
$$\boldsymbol{K}_{P_f} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \boldsymbol{K}_{I_f} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{bmatrix} \tag{5.2}$$

The main results of this experience are depicted in Figure 5.3 and some snapshots taken at purposeful moments are shown in Figure 5.4.

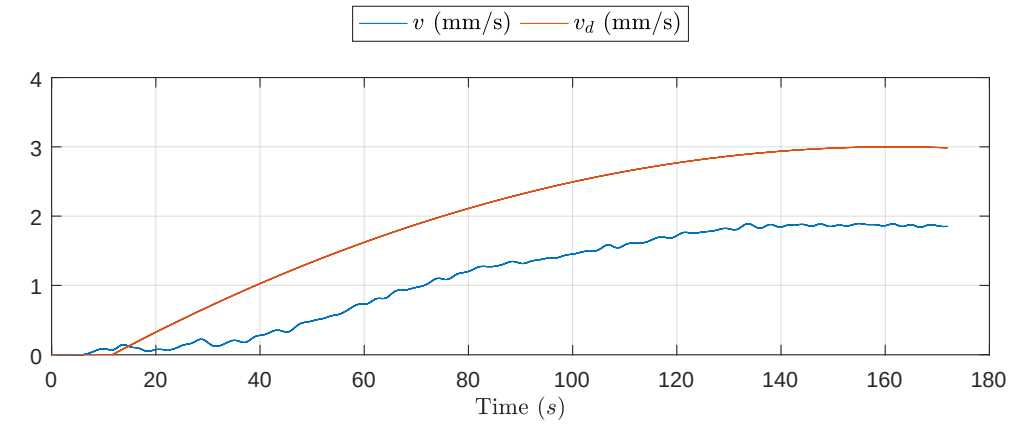
From Figure 5.3, it is clear that the manipulator accommodates its position along z axis to impose a desire force of $1.5N$ on the surface (in the negative direction of z axis) while task execution velocity is changed due to the values of F_y. Similar to previous experiment, the manipulator is not force controlled while there is no contact with the environment, within the interval $t \in [0, 6.93]$. During this period of time it is executing a downwards approaching trajectory with an average velocity of $15mm/s$ (see snapshot Figure 5.4(a)). When the surfaced is felt (see snapshot of Figure 5.4(b)), the force peak does not affect the TCP position due to the waiting time between states. As one can see, force peak occurs at $t = 7.17s$



(a) Force along y_e and z_e axis



(b) TCP position with respect to base frame

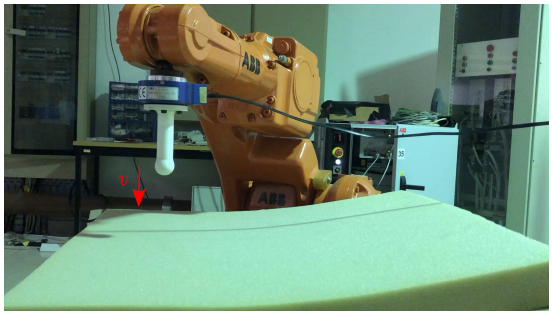


(c) Velocity profile

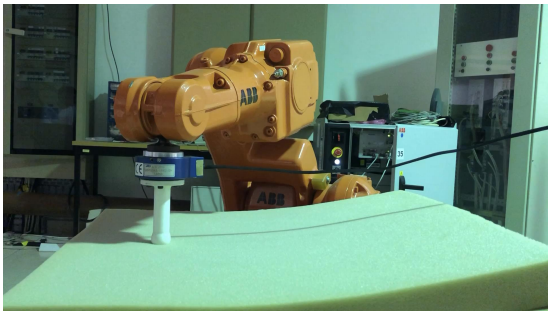
Figure 5.3: Main results of the second experiment

during the period which force values are ignored by the control strategy therefore its action is mitigated.

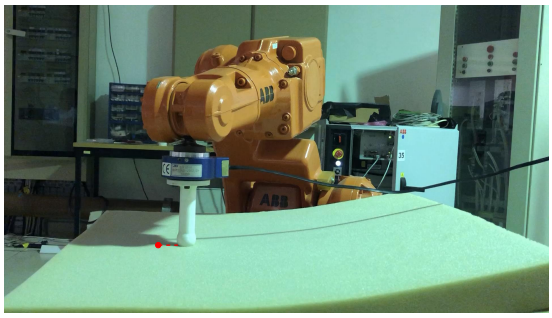
Afterwards, between $t = 7.83s$ and $t = 10.5s$ the manipulator stabilizes the position with small accommodations so the desired force value can be imposed. At $t = 11.4s$ the manipulator starts the task itself and performs a rectilinear trajectory on the xy plane with a velocity profile defined by v_d in Figure 5.3(c) from the position where force error was considered null. The profile in blue represents the task execution velocity, which was estimated using a $\alpha - \beta$ filter as in the work [7] and then passed through a low pass filter. The velocity of the robot during the "Force Stabilization" state is observable right before the parabolic profile starts. From the same plot it can be observed the difference between desired and task execution velocity profiles. The speed reduction demonstrates that the controller works for directions both position and force controlled. However, while the force increases the velocity decreases(see Figure 5.3(a)). That is due to the way the trajectory planner is computing the next trajectory point which does not match with the purpose of speed changing in a robotic task. It is intend to change the task velocity so it can decrease the force to a desired value.



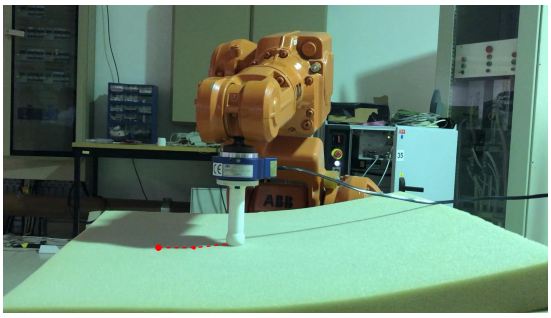
(a) $t \approx 0s$



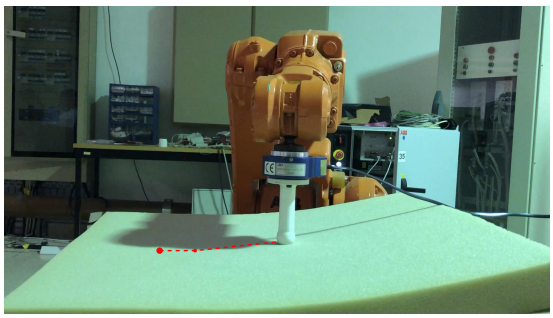
(b) $t \approx 8s$



(c) $t \approx 79s$



(d) $t \approx 130s$



(e) $t \approx 172s$

Figure 5.4: Snapshots of the second experiment

For this experiment, there is a difference in the environment surface from the previous experiment, it

is not completely flat as depicted in Figure 5.4. Situation that can also be seen on the TCP position in Figure 5.3(b). Surface slope begins to be sensed by the robotic system at $t = 79s$ (see Figure 5.4(c)), when the difference between velocity profiles begins to increase. Previously, the difference is almost constant and is due to friction between the tool and the material. Throughout all experiment the desired TCP orientation is kept constant and perpendicular to the surface. Figure 5.4(e) marks task finale where the desired position was achieved, FSM entered in the "Stop" state.

Just like in the Pressure experiment, the material location and surface contour are not known. Despite the fact that manipulator starts task execution on the position held from the "Force Stabilization" state (where force error was considered zero) it follows the surface contour. That is, force regulation to a desired value prevails over position control.

5.3 Experiment 3: Moment Control - Assembly

Similarly to the first experiment but regarding moment control, here is desired to be applied a torque against a surface. This particular test has similarities with a situation that can occur in a peg-in-hole assembly task. The manipulator will 'feel' the environment as a torque and will try to impose a certain moment against it. As in the previous experiments, the environment is not known. Force controller is not required and will not be used.

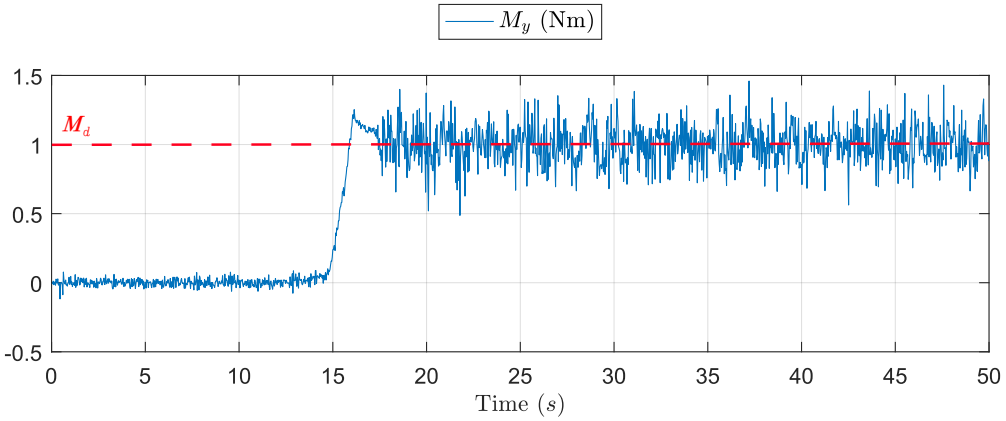
This experiment will only portray the first two states of the Force-moment state-machine, since assembly task is more complex than machining tasks. They generally have algorithms running above the control action to decide how to react upon a given torque and that is not in the scope of the present work. As already mentioned, this thesis intends to serve as platform to design machining and assembly tasks. Therefore, it tests how a moment influence the end-effector orientation and runs with transition conditions on FSM regarding moment error along y_e-axis.

Controller gains were selected through simulation tunning and adapted experimentally due to remote control bandwidth and task execution speed. The moment and orientation controller gains used are,

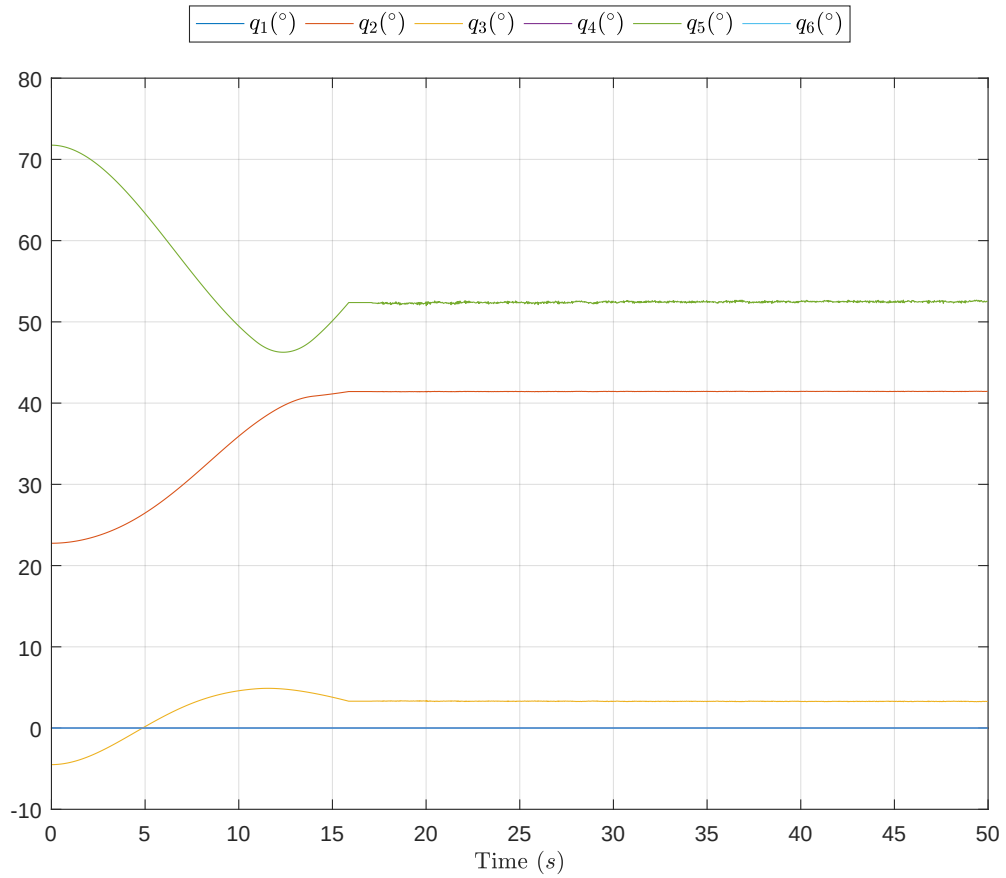
$$\boldsymbol{K}_{P_m} = \begin{bmatrix} 0.3 & 0 & 0 \\ 0 & 0.3 & 0 \\ 0 & 0 & 0.3 \end{bmatrix}, \qquad \boldsymbol{K}_{I_m} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \qquad \boldsymbol{K}_O = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix} \tag{5.3}$$

The main results of this experience are depicted in Figure 5.5 and some snapshots taken at meaningfully moments are shown in Figure 5.6.

From Figure 5.5, it is depicted that the manipulator accommodates its orientation along $y-$axis (on the q_5 joint) to impose a desire moment of $1Nm$ on the surface. Similar to previous experiment, it makes an approach to the work object within the interval $t \in [0, 15.87]$. The manipulator executes a downwards trajectory of $210mm$ (see snapshot Figures 5.6(a)) and then a rotation of 30° along y_e-axis (see snapshot Figure 5.6(b)). However, the surface is felt (see snapshot of Figure 5.6(c)) before it can reach the final orientation for the desired rotation. As for the force control experiments, Force-moment state-machine avoids the moment peak sensed when contacting the object.

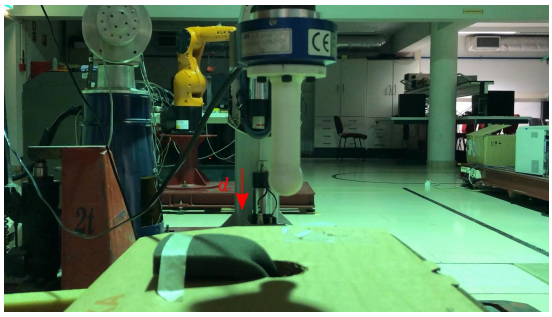


(a) Moment along y_e axis

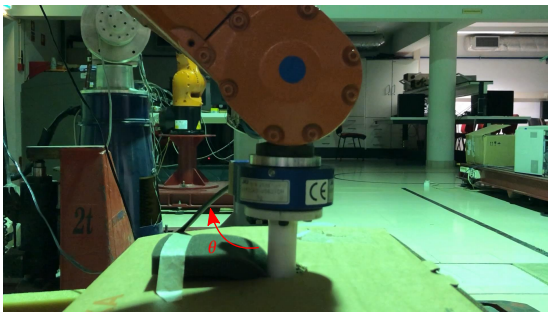


(b) Manipulator joint coordinates

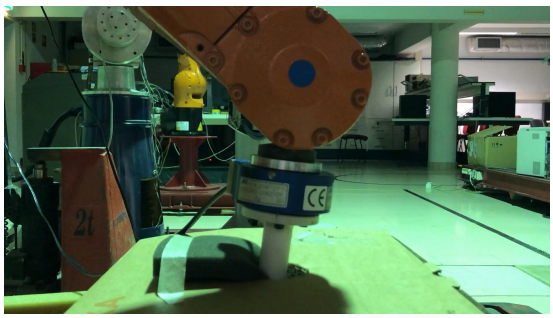
Figure 5.5: Main results of the third experiment



(a) $t \approx 8s$



(b) $t \approx 15s$



(c) $t \approx 18s$

Figure 5.6: Snapshots of the third experiment

Although, Figure 5.5(a) represents the manipulator maintaining the desired imposed moment it also shows bigger fluctuations around that value comparing to the experiments on Force control. That is due to the low sampling rate used, the robotic system can not react in time. In Force control that effect can be overcome by lowering the execution task speed. However, for the controller gains chosen the effects of those fluctuations are almost unnoticed on joint coordinate q_5 (see Figure 5.5(b)) which is where the effect would be shown.

The Force-moment state-machine transitioning from the states ”Approach” to ”Force Stabilization” holds a good pose approximation for which the moment imposed is the desire. The noncontact to contact transition smoothness can be attributed to the low approaching velocity.

Chapter 6

Conclusions

This thesis focused on developing a force/ moment control strategy with the aim of giving a industrial manipulator endowed by standard industrial controller the ability to 'feel' its surroundings. However, it was aware from the very beginning of the limitations in the hardware-software architecture regarding the communication bandwidth of $33Hz$ and that it would affect the conditions to perform a task with such a controller. Velocities, controller gains and consequently response time needed to consider that drawback. Otherwise, the system would not have time to react to the forces sensed. This also affected the characteristics of the environment for which the controller was tested. It was used a compliant material.

In order to validate the theoretical findings, the force/ moment controller developed was tested in a number of experiments for representative interaction tasks. Commercially available industrial robots are purposefully utilized to demonstrate the credibility of force control in the perspective of the next generation robot control units with enhanced sensory feedback capabilities. The control strategy was experimentally validated using the hardware-software architecture proposed, which differs from those found in other academic works essentially because it has a moment control action implemented without any access to the low-level axis controller of the IRC5. Most of moment control strategies proposed in the literature are implemented using formulations that can not be used in close-ended industrial manipulators. The approach, as it is characteristic of position-based controllers, only uses the manipulator as a 'positioning device' exploiting both position and orientation. Therefore, the hardware-software architecture used allows the integration of the force/ moment controller proposed with industrial manipulators like the IRB140 without any investment in a special industrial controller, which was the motivation for carrying out this work.

This controller can be designed considering a simplified model of the environment for both position and orientation while providing some sort of robustness to uncertainty. Other strategies demand *a priori* knowledge of the contact surface characteristics and the dynamics inherent in order to tune properly the controllers gains. In industrial environments and many others, that information it is not available nor is easy to obtain in a precise way. That makes infeasible or very hard to utilize such control strategies, given their dependency for the environment model.

The results of the experiments show how the control strategy presented can increase the robotic system flexibility by being capable of perform machining and assembly tasks where the force-moment action should prevail motion control. They also enhanced force-moment state-machine importance on the task design. Using the force-moment controller with different goals in its states, overcomes the drawbacks of the controller itself on performing a task as the only component on the control strategy. This type of controllers require that contact must already be established, specially in stiff environments. The noncontact to contact transition without the force-moment state-machine would probably damage the tool used due to the force peaks that rise when contacting the environment.

Besides bypassing the concern with impact control it also enables the force/ moment controller's action only when there is contact with the environment. Allowing the robotic system to perform free motion movements when there is no contact and avoiding force peaks that appear on contact.

Although the controller is designed to operate in a full-dimensional space without using selection matrices it is the user that defines which directions are going to be force and/ or position controlled. It is here presented that for hybrid situations, where force and position controlled directions are orthogonal the architecture developed can be directly applied on a robotic application. However, for a situation where a direction is both force and position controlled the trajectory planner running on the force-moment state-machine, therefore apart from the controller, must be adapted. The next desired position on the trajectory should be defined accordingly to the sensed force on the direction of the movement, so it can be maintained at desirable values.

Considering that the goal was achieved and a force/ moment controller on a close-ended industrial manipulator was successfully implemented, it is our belief that a more capable robotic system would present better results and could perform more complex tasks. With higher sampling rate the system response time would be faster and the robotic system would be capable of performing machining tasks at real velocities. Force-moment state-machine for instance would be capable of achieving the pose where the force error is null or would narrower the interval.

6.1 Future Work

Considering the complexity of a force control strategy so it that can be applied to real industrial environments, the present work has room for improvement. In the following section are presented ideas that can be exploited in the future.

- **Speed Change**

  The ability of the position-based force control to change the task execution velocity due to sensed forces on the opposite direction of the movement is a consequence of how the trajectory planner is implemented. Velocity is not directly controlled, the position is. For this situation to be accounted and for keeping the hardware-software architecture intact, it is necessary to design a reactive trajectory planner that would run on the force-moment state-machine. That is, the trajectory's next desired pose would be computed so that a desired force can be achieved. Alternatively, the

controller could control velocity instead of position. The controller would act on the trajectory timing law, so it could be transformed into a position and then sent to the robot motion controller. That way, the control strategy would be different but it could still be implemented with the remote control used.

- **Communication Protocol**

  In this work the communication between the external computer and the IRC5 is achieved via RS-232 which limits the bandwidth of the system to $33Hz$. This frequency is seen as the bottleneck of the system since it influences directly the controller gains, velocity at which the manipulator can operate and the stiffness of the material used. Using an UDP protocol would allow $0.001s$ communication instead of $0.03s$.

- **Low-level access**

  As mentioned before, the force-moment control strategy is implemented using a standard close-ended IRC controller, meaning the control strategy was implemented around the IRC position control considering the robot as a "positioning device". The data received from the FTS could not be sent or read by the controller, the sensor needed to be connected to an external computer so the computer could communicate with the robot.

  - **External Guided Motion**

    There is a new ABB® option called External Guide Motion (EGM) in RobotWare 6 that allows *Path Guidance*, the robot follows a path generated by an external device; and *Path Correction*, a programmed robot path can be modified/ corrected using measurements provided by an external device. Since EGM Path Correction does not allow corrections in the path direct or in the orientation, EGM Path Guidance would be the only hypothesis for the control strategy developed in this work. EGM Path Guidance can be used to read positions from and write positions to the motion system at every $4ms$ with a control lag of $10 - 20ms$ depending on the robot type. Thus, the controller interpellator could be used to generate the path instead of the external trajectory planner.

  - **ABB Force Control Machining**

    Other hypothesis to act on lack of access to lower layers of the IRC controller would be with the ABB Integrated Force Control. This IRC package has force control RAPID instructions for machining and assembly, along with recovery path functions. With the capability of support any 6-DOF force torque sensor. Therefore, the control strategy location would be shifted to IRC controller and needed to be written in RAPID. System bottleneck would be no longer the communication between the external computer and the robot controller.

References

[1] International Federation of Robotics. Executive Summary World Robotics 2018 Industrial Robots. https://ifr.org/downloads/press2018/Executive_Summary_WR_2018_Industrial_Robots.pdf. [Accessed: 01/04/2019].

[2] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer, 1st edition, 2008.

[3] A. Winkler and J. Suchỳ. Robot ForceTorque Control in Assembly Tasks. *IFAC Proceedings Volumes*, 46(9):796–801, 2013.

[4] H. Born and J. Bunsendal. Programmable Multi Sensor Interface for Industrial Applications. In *Conference Documentation International Conference on Multisensor Fusion and Integration for Intelligent Systems. MFI 2001 (Cat. No. 01TH8590)*, pages 189–194. IEEE, 2001.

[5] ISO 8373:2012 robots and robotic devices – Vocabulary, March 2012.

[6] V. Sampaio. Towards Interaction Control of an Industrial Robotic Manipulator. Master's thesis, Instituto Superior Técnico, October 2017.

[7] R. Lucas. Towards Safe Human-Robot Interaction in Industrial Environments. Master's thesis, Instituto Superior Técnico, May 2018.

[8] B. Siciliano and L. Villani. *Robot Force Control*, volume 540. 1st edition, 1999.

[9] B. Siciliano and O. Khatib. *Springer Handbook of Robotics*. Springer, 2016.

[10] V. Miomir, S. Dragoljub, and E. Yury. *Dynamics and Robust Control of Robot-environment Interaction*, volume 2. World Scientific, 2009.

[11] C. Natale. *Interaction Control of Robot Manipulators: Six degrees-of-freedom tasks*, volume 3. Springer-Verlag Berlin Heidelberg, 2003.

[12] O. Khatib. A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[13] M. Raibert and J. Craig. Hybrid Position/ Force Control of Manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 102:126–133, 1981.

[14] N. Hogan. Impedance Control: An Approach to Manipulation: Part I-III. *Journal of Dynamic Systems, Measurement, and Control*, 107, 1985.

[15] J. D. Schutter, H. Bruyninckx, W. H. Zhu, and M. W. Spong. Force control: A bird's eye view. In K. P. Valavanis and B. Siciliano, editors, *Control Problems in Robotics and Automation*, volume 230, pages 1–17. Springer, Berlin, Heidelberg, 1998.

[16] R. J. Anderson and M. W. Spong. Hybrid Impedance Control of Robotic Manipulators. *IEEE Journal on Robotics and Automation*, 4(5):549–556, 1988.

[17] S. Chiaverini and L. Sciavicco. The Parallel Approach to Force/ Position Control of Robotic Manipulators. *IEEE Transactions on Robotics and Automation*, 9(4):361–373, 1993.

[18] F. Almeida, A. Lopes, and P. Abreu. Force-Impedance Control of Robotic Manipulators. In *Proceedings of the 4th Portuguese Conference on Automatic Control*, 2000.

[19] E. Dégoulange and P. Dauchez. External Force Control of an Industrial PUMA 560 Robot. *Journal of robotic systems*, 11(6):523–540, 1994.

[20] M. T. Mason. Compliance and Force Control for Computer Controlled Manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(6):418–432, 1981.

[21] H. Bruyninckx and J. De Schutter. Specification of Force-Controlled Actions in the "Task Frame Formalism" – A Synthesis. *IEEE Transactions on Robotics and Automation*, 12(4):581–589, 1996.

[22] J. De Schutter and H. Van Brussel. Compliant Robot Motion ii. a Control Approach based on External Control Loops. *The International Journal of Robotics Research*, 7(4):18–33, 1988.

[23] F. Pierrot, E. Dombre, E. Dégoulange, L. Urbain, P. Caron, S. Boudet, J. Gariépy, and J.-L. Mégnien. Hippocrate: A safe robot arm for medical applications with force feedback. *Medical Image Analysis*, 3(3):285–300, 1999.

[24] C. Natale, B. Siciliano, and L. Villani. Control of Moment and Orientation for a Robot Manipulator in Contact with a Compliant Environment. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation*, volume 2, pages 1755–1760. IEEE, 1998.

[25] *Product specification - Articulated robot*. ABB, 2004. 3HAC9041-1.

[26] *JR3 PCI-BUS Receiver Device Driver - Programmer an User Manual*. Instituto Superior Técnico.

[27] MathWorks®. Getting Started with Simulink Real-Time. `https://www.mathworks.com/help/xpc/getting-started-with-xpc-target-1.html`. [Accessed: 25/03/2019].

[28] *Techinical reference manual - RAPID overview*. ABB, 2004. 3HAC16580-1.

[29] J. N. P. da Silva. *Realização de Controlo de Força em Robôs Manipuladores Industriais*. PhD thesis, Faculdade de Ciências e Tecnologia da Universidade de Coimbra, 1999.

[30] S. Chiaverini and B. Siciliano. The Unit Quaternion: A Useful Tool for Inverse Kinematics of Robot Manipulators. *Systems Analysis Modelling Simulation*, 35(1):45–60, 1999.

[31] *Aplication manual - Force Control*. ABB, 2018. 3HAC050377-001.

Appendix A

Appendix A

A.1 Unit Quaternion

The relationship between the time derivative of the quaternion and the body angular velocity is established by the quaternion propagation:

$$\dot{\eta} = -\frac{1}{2}\boldsymbol{\epsilon}^T\boldsymbol{\omega}, \tag{A.1a}$$

$$\dot{\boldsymbol{\epsilon}} = \frac{1}{2}\mathbf{E}(\eta,\boldsymbol{\epsilon})\boldsymbol{\omega}. \tag{A.1b}$$

where

The skew-symmetric operator applied on a vector $\boldsymbol{\epsilon} = (\epsilon_x, \epsilon_y, \epsilon_z)$ will result in the skew-symmetric matrix:

$$\mathbf{S}(\boldsymbol{\epsilon}) = \begin{bmatrix} 0 & -\epsilon_z & \epsilon_y \\ \epsilon_z & 0 & -\epsilon_x \\ \epsilon_y & \epsilon_x & 0 \end{bmatrix}. \tag{A.2}$$

A.2 Angle and Axis

The rotation matrix corresponding to a given angle and axis is

$$\boldsymbol{R}(\vartheta,\boldsymbol{r}) = \begin{bmatrix} r_x^2(1-c_\vartheta)+c_\vartheta & r_x r_y(1-c_\vartheta)-r_z s_\vartheta & r_x r_z(1-c_\vartheta)+r_y s_\vartheta \\ r_x r_y(1-c_\vartheta)+r_z s_\vartheta & r_y^2(1-c_\vartheta)+c_\vartheta & r_y r_z(1-c_\vartheta)-r_x s_\vartheta \\ r_x r_z(1-c_\vartheta)-r_y s_\vartheta & r_y r_z(1-c_\vartheta)+r_x s_\vartheta & r_z^2(1-c_\vartheta)+c_\vartheta \end{bmatrix}. \tag{A.3}$$

A.3 Modelling

In every control problem an accurate model of the system to be controlled is very helpful to the purpose of controller design. A robot manipulator consists of kinematic chain of $n+1$ links connected by means

of n joints. One end of the chain is connected to the base link, whereas an end-effector is connected to the other end.

Let $\boldsymbol{q}$ denote the 6×1 vector of joint variables, $O_b - x_b y_b z_b$ and $O_e - x_e y_e z_e$ be a frame attached to the base of the robot (base frame) and a frame attached to the end-effector (end-effector frame), respectively.

The position of the end-effector frame with respect to the base frame is the position of the origin O_e represented by the 3×1 vector $\boldsymbol{p}_e$, while the orientation of the end-effector frame with respect to the base frame can be represented in many ways. The representation used will be the unit quaternion $\mathcal{Q} = (\eta_e, \boldsymbol{\epsilon}_e)$. Regarding the unit quaternions, a first subscript will denote the frame whose orientation is represented by the quaternion itself and a second subscript will denote the reference frame, hence it will be dropped when referring to the base frame. Let $\dot{\boldsymbol{q}}$ denote the vector of joint velocities, $\dot{\boldsymbol{p}}_e$ the 3×1 vector of end-effector linear velocity, and $\boldsymbol{\omega}_e$ the 3×1 vector of end-effector angular velocity. The differential kinematics model gives the relationship between $\dot{\boldsymbol{q}}$ and

$$\boldsymbol{v}_e = \begin{bmatrix} \dot{\boldsymbol{p}}_e \\ \boldsymbol{\omega}_e \end{bmatrix} \tag{A.4}$$

in the form

$$\boldsymbol{v}_e = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{A.5}$$

where $\boldsymbol{J}$ is the 6×1 end-effector geometric Jacobian matrix. The Jacobian can be partitioned as

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_p \\ \boldsymbol{J}_o \end{bmatrix} \tag{A.6}$$

to separate the contributions of the joint velocities to the linear and the angular velocity in (A.4). The joint configurations at which the matrix $\boldsymbol{J}$ is not full-rank are termed kinematic singularities.

The dynamic model of the manipulator can be written in the compact matrix form,

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{F}_v\dot{\boldsymbol{q}} + \boldsymbol{F}_s\mathbf{sgn}(\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} - \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{h}_e, \tag{A.7}$$

where $\boldsymbol{B}(q)$ is the $n \times n$ symmetric and positive definite inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ is the $n \times 1$ vector of the Coriolis and centrifugal torques, $\boldsymbol{g}(\boldsymbol{q})$ is the $n \times 1$ vector of the gravitational torques, $\boldsymbol{F}_v\dot{\boldsymbol{q}}$ and $\boldsymbol{F}_s\mathbf{sgn}(\dot{\boldsymbol{q}})$ the viscous and static friction torques, respectively, $\boldsymbol{\tau}$ is the $n \times 1$ vector of joint driving torques and $\boldsymbol{h}_e = [\boldsymbol{f}^T \quad \boldsymbol{\mu}^T]$ is the 6×1 vector of force and torque exerted on the manipulator's end-effector by the environment.

In fact, solving (A.7) for the joint accelerations, and neglecting the joint friction torques for simplicity, yields

$$\ddot{\boldsymbol{q}} = -\boldsymbol{B}^{-1}(\boldsymbol{q})\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{B}^{-1}(\boldsymbol{q})\boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{B}^{-1}(\boldsymbol{q})\boldsymbol{J}^T(\boldsymbol{q})(\boldsymbol{\gamma}_e - \boldsymbol{h}_e), \tag{A.8}$$

As the present work is focused on the problem of interaction control, the objective is to achieve a desired dynamic behaviour for the end-effector.

A good starting point to pursue this goal is the acceleration-resolved approach used for motion control, which is aimed at decoupling and linearizing the non-linear robot dynamics at the acceleration level, via an inverse dynamics control law. Therefore, in the presence of interaction with the environment, it is useful rewriting the dynamic model of the manipulator with respect to the end-effector acceleration

$$\dot{\boldsymbol{v}}_e = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q})\dot{\boldsymbol{q}}. \tag{A.9}$$

In particular, replacing (A.8) in the above expression yields

$$\boldsymbol{B}_e(\boldsymbol{q})\dot{\boldsymbol{v}}_e + \boldsymbol{n}_e(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\gamma}_e - \boldsymbol{h}_e, \tag{A.10}$$

where

$$\boldsymbol{B}_e = \boldsymbol{J}^T \boldsymbol{B} \boldsymbol{J}^{-1}, \tag{A.11a}$$

$$\boldsymbol{n}_e = \boldsymbol{J}^T(\boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{g}) - \boldsymbol{B}_e\dot{\boldsymbol{J}}\dot{\boldsymbol{q}}. \tag{A.11b}$$

By adopting the inverse dynamics control law

$$\boldsymbol{\gamma}_e = \boldsymbol{B}_e(\boldsymbol{q})\boldsymbol{\alpha} + \boldsymbol{n}_e(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{h}_e, \tag{A.12}$$

and cast (A.12) into the dynamic model (A.7) results in

$$\dot{\boldsymbol{v}}_e = \boldsymbol{\alpha} \tag{A.13}$$

where $\boldsymbol{\alpha}$ is a properly designed control input with the meaning of an acceleration referred to the base frame.