DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

ÉCOLE DOCTORALE 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Automatique, Productique et Robotique*

Par

Alexander OLIVA

Integration of vision and force control for physical interaction with robotic manipulators

Thèse présentée et soutenue à Rennes, le 22/02/2022
Unité de recherche : Inria-Rennes/IRISA

Rapporteurs avant soutenance :

Andrea CHERUBINI Professeur, Université de Montpellier, LIRMM
Youcef MEZOUAR Professeur, Université Clermont-Auvergne, Sigma

Composition du Jury :

Président :	Véronique PERDEREAU	Professeure, Sorbonne Université, Paris, ISIR, France
Examinateurs :	Joris DE SCHUTTER	Professeur, KU Leuven, Leuven (Arenberg), RAM, Belgique
	Andrea CHERUBINI	Professeur, Université de Montpellier, LIRMM, France
	Youcef MEZOUAR	Professeur, Université Clermont-Auvergne, Sigma, France
	Véronique PERDEREAU	Professeure, Sorbonne Université, Paris, ISIR, France
Dir. de thèse :	François CHAUMETTE	Directeur de recherche Inria, Inria/IRISA Rennes, France
Co-dir. de thèse :	Paolo ROBUFFO GIORDANO	Directeur de recherche CNRS,IRISA/Inria Rennes, France

To Tato...

« In fisica e in matematica è impressionante la sproporzione tra lo sforzo per capire una cosa nuova per la prima volta e la semplicità e naturalezza del risultato una volta che i vari passaggi sono stati compiuti. Nel prodotto finito, nelle scienze come nella poesia, non c'è traccia della fatica del processo creativo e dei dubbi e delle esitazioni che lo accompagnano. »

« In physics and mathematics, the disproportion between the effort to understand something new for the first time and the simplicity and naturalness of the result once the various steps have been completed is striking. In the finished product, in the sciences as in poetry, there is no trace of the effort of the creative process and the doubts and hesitations that accompany it. »

Giorgio Parisi, Physics Nobel Prize 2021.

Acknowledgments

This doctoral journey now draws to an end after a long trip that begun even before I obtained my doctoral position in this fantastic laboratory and set foot on Breton land. This manuscript summarizes and encapsulates the scientific and experimental work that I have done over the last three and a half years in a formal and concise manner, but the narrative style that is adopted in this type of literature leaves no room for recounting the vicissitudes and experiences that one faces both inside and outside of the laboratory. So I would like in this anecdote to thank everyone who, in one way or another, contributed to the achievement of this thesis work.

Robots have fascinated me since I was a child and I knew that the way forward was toward understanding the laws that govern them and make them seem endowed with life.... but passions sometimes succumb to more imperative reasons, especially if pockets have a say; so, after university I began to work in industry and began, reluctantly and in my late twenties, to get used to the idea that maybe, in my spare time and as a hobby, I could devote myself to some "Maker" projects in robotics. Then, one day, I was in L'aquila and went to visit my very dear robotics professor, Costanzo Manes, whom I owe enormous gratitude to for the extremely inspiring and motivating discussion we had that day, encouraging me to seek what is truly important to me, and I must say that my personal and professional life has never been the same since.

The transition to research occurred gradually. I first arrived beyond the Alps as a research engineer at Inria (Grenoble), an institution to which I am eternally grateful for everything it has given me over the years and which has always made me feel like part of a big family. There, under the supervision of my friend Agostino Martinelli (a.k.a. "Massiccio"), I took my first steps into the fascinating world of research. The conversations we used to have about mountains and science until I left for Rennes, also with Alessandro Renzaglia, were pure inspiration.

At the heart of these acknowledgements must be two extraordinary people and researchers, my supervisors: Paolo Robuffo Giordano and François Chaumette, whom, in addition to always being present and available (even late at night), gave me complete freedom in conducting research and created the optimal conditions

for allowing me to complete my doctorate, taking also into account the not trivial historical context we faced. It was an honour and a privilege to be under their supervision.

Much of the work I conducted in the lab was experimental, and this would have been really arduous without the help and support of Fabien Spindler, with whom it was always pleasant to work with and see in action working flawlessly.

During my third year, I had the opportunity to visit KU Leuven for a brief but rewarding period owing to collaboration with Prof. Joris de Schutter and Erwin Aertbeliën, whom I thank for welcome me to their laboratory and for the stimulating discussions.

At this point, the list of people who must be thanked for any scientific or non-scientific conversation is extensive. I'd like to thank Marco Cognetti in particular because he always kept his office door open for any questions I had during my first year of PhD. Thank you also to Claudio Pacchierotti and Marco Aggravi for countless tips and assistance, to Claudio Gaz for the fruitful collaboration, to Alberto Jovane, master of blender and Adam Khayam, wizard of any programming paradigm. Thanks to Nicola De Carli and John Thomas, great office neighbours for insightful talks... just thank you to all the RAINBOWERS, I cannot name you all.

I would also like to thank my family for their unconditional support, regardless of any decisions I may make in my life.

Last but not least, and as is always the case in these acknowledgements, one tends to leave last to the person who had a substantial weight in the success of the thesis, because without that person one should have had cloned oneself to be able to lead a "normal" life. And you thank that person because has always been by your side, every day to talk to you, because has always listened to you carefully and knows every point of your research even if you never showed an equation. You also thank that person for being able to give you your space when you were under dead-line during holidays, a special event or even a PACS. For all these reasons and thousands more, I thank you Naty.

I Hereby acknowledge the support of the Brittany region during my research visit at the KU Leuven.

Contents

Chapter 1

Introduction

Contents

1.1 On Vision-Force coupling: Motivation

Robotics, the term that *Isaac Asimov* assumed in his short story "Liar!" in 1941, refers to the science and technology studying *robots*, the mechanical machines he imagined devoid of feelings and with a "*positronic*" brain programmed by man. The term *robot*, on the other hand, appeared for the first time twenty years earlier (1920) in the science-fiction *play* "R.U.R. (Rossum's Universal Robots)" by the Czech writer *Karel Čapek* in which artificial biological organisms, that may be mistaken for humans, were assigned the task of replacing human beings in subordinate jobs. This word, coined by Karel Čapek's brother *Josef*, comes from the Czech "robota" which means indeed "hard work" or "forced labour" and is derived from *rab*, meaning "slave". Even though this "modern" term dates back to a century ago, our desire to build machines capable of replacing humans in any disparate situation or in tedious, repetitive, or hazardous tasks has much more ancient roots, today more fervent than ever.

(a)

(b)

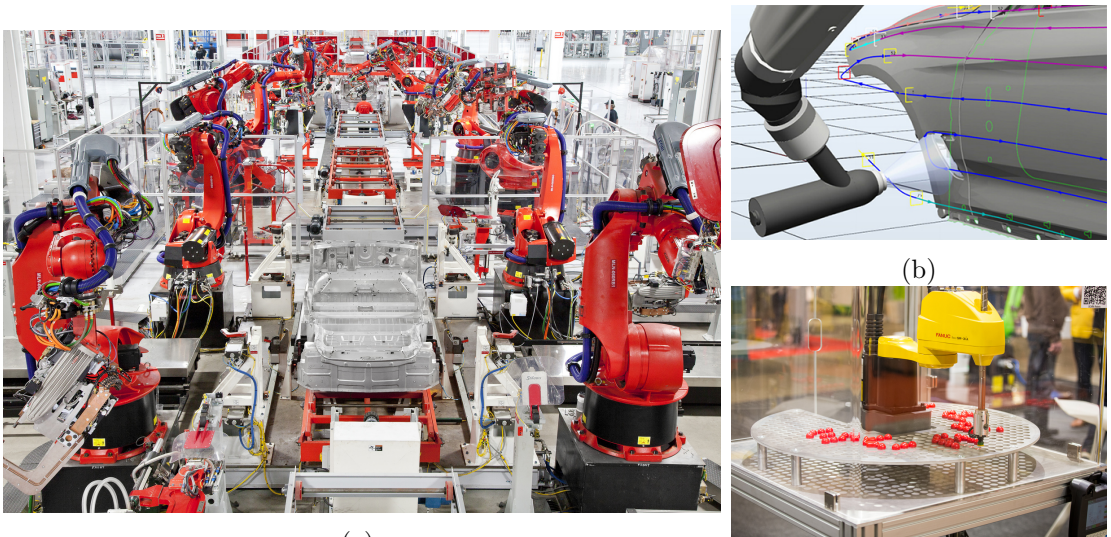
(c)

(d)

Figure 1.1 – From Greek mythology to date: (a) Obverse of silver didrachma from Phaistos depicting Talos ("$TA\Lambda\Omega N$"),$\sim$ 400 B.C.. (b) The Leonardo Da Vinci's "Automaton knight" $\sim$1495. (c) Unimate: First industrial robot, 1961. (d) Boston Dynamics' Atlas: the next generation, 2016.

In the Greek mythology from the time of Homer, Talos was a bronze giant, forged by Hephaestus, the blacksmith god of technology, to protect Europa in Crete. The word automaton, "self-moving", was first used by Homer (750-650 B.C.) to describe the many animated devices fabricated by Hephaestus. Figure 1.1a shows the obverse of a silver didrachma depicting Talos ("$TA\Lambda\Omega N$"). Figure 1.1b shows the Leonardo da Vinci's "Automaton knight", a humanoid automaton. In 1961, George Devol invented the world's first industrial robot "Unimate", shown in Figure 1.1c. Finally, Figure 1.1d shows Atlas:the next generation, developed by Boston Dynamics in 2016; it is likely to be the most advanced robot ever built to date, being able to parkour on an obstacle track as well as perform back-flips.

Modern factories are making large use of robotic manipulators to perform very specific, restricted and deterministic tasks, *e.g., welding, pick-and-place or spraying* (as shown in Figure 1.2), for which the exact sequence of the operations to be performed is perfectly known *a priori*. With these restrictions and considering the highly structured and well-controlled environments in which these robots are routinely used, the design of the task control becomes relatively simple resorting to path/trajectory planning techniques, which will benefit from the excellent kinematic capabilities of industrial robots and the available prior knowledge of the environment.



(a) (b) (c)

Figure 1.2 – (a) Tesla giga-factory assembly line: at each station several manipulators equipped with arc welding machines apply the preprogrammed welding points. (b) Path planned for an automated car-body spraying operation. (c) A SCARA robot performing pick-and-place operations in a well-structured environment.

These robots have been used as mere multipurpose automatons, to automate industrial production processes under the justification of greater productivity and quality control, without really exploiting their tremendous flexibility compared to fixed automated machines; this is also due to the fact that industrial manipulators are unsophisticated and simple-minded with very poor decision making skills right out of the box (commercial industrial robots are typically provided with *proprioceptive* joint position sensors only). Indeed, flexibility is the real strength that distinguishes *man*, the most versatile manipulator of nature, from machines. This great flexibility derives in part from the ability to process multiple sensory information; hence, our interest in projecting these sensing abilities into industrial robots, being essential for reaching higher levels of flexibility and autonomy. This will enable even small-sized companies with limited production volumes and different products to benefit from this technology. The possibility of this scenario was already envisioned by Bajcsy [1] four decades ago. The rise of the so-called *Cobots*, or Collaborative Robots, are indeed one of the key technologies that are driving the modern industrial revolution, also known as industry 4.0. *Cobots* are a driving technology that is pushing hard on the *smart production* front, That is, the forefront of new manufacturing technologies that foster collaboration among all the elements involved in the manufacturing process, i.e., collaboration among operators, machines, and tools.

Although path/trajectory planning approaches have proved their effectiveness in well-structured applications, which among other things require a considerable modeling and calibration engineering effort, those methods are hardly applicable to the highly dynamic scenarios illustrated above, which could even involve voluntary or accidental contacts with a human operator or the environment. *Sensor-based* control, on the other hand, offers more flexibility in dealing with the uncertainties that characterize unstructured/dynamic or even non-rigid environments.

Of all the available sensing modalities, vision and force sensing are among the most complementary and relatively inexpensive ones. Cameras are capable of providing a rich description of the scene in which the manipulator moves, allowing, for example, to accurately align the manipulator's end-effector with an object within an imprecisely calibrated or dynamically changing environment. In contrast, force sensors provide much more accurate, albeit localised, contact information, enabling the control task to be performed with greater precision than vision alone. With no surprise, vision and force sensing are the most widely adopted sensory configuration in the context of physical human-robot interaction (pHRI) among the sensing modalities (*i.e.*, vision, force(tactile), audition and distance) recently surveyed by Cherubini et al. [2].

Before delving into the difficulties that multi-sensory integration entails, the scientific community has sought to address the need to describe and characterize the control qualities of each specific sensory modality. Early vision and force sensing research focused on identifying the strengths and shortcomings of the control techniques associated with each of these sensors independently. Indeed, two strong communities, that of force control and the visual-servoing one, have achieved a number of significant breakthroughs in the past decades, bringing their respective fields to a state of relatively advanced maturity. This thesis is halfway between these two fields and aims at providing solutions to the challenging problem of the coupling of vision and force sensing by proposing both innovative combined control strategies as well as the tools needed to support the study, *e.g.,* in simulation, and the implementation of those strategies.

Two decades ago, force and/or vision sensors were still an exception in industrial settings and were mostly confined to the field of scientific research [3].

The need to develop novel integrated vision-force control techniques that are safer, robust and highly performing is also driven by the ever increasing adoption of *Cobots* in small and medium-sized businesses, where robots find themselves sharing the workspace with human operators and other machinery.

1.2 Challenges

There is of course significant interest in integrated vision-force approaches, as the development of combined vision-force applications is just a natural step in the evolution of sensor-based robotics [3], allowing the advantages of both sensors to be combined and the individual weaknesses to be overcome. However, extra needs arise as a result of the integration process, and attaining an effective combined use of vision and force is not straightforward. Vision and force sensors measure fundamentally different quantities, which leads to inherent difficulties when trying to combine data from the two sensors. Sensor integration approaches need a common representation of the sensory input to be integrated, but force and vision sensors do not share such a common representation of the data they provide. Furthermore, due to their distinct nature, the two sensors are often only useful at distinct stages of the task at hand. On the other hand, the most common force control schemes make use of wrist-mounted force/torque sensors, which are capable of measuring any kind of force: gravitational, inertial or contact force. The inertial coupling of the end-effector mass (and eventual payload) positioned downstream of the sensor introduces a fundamental problem when the controlled directions are shared between vision and force sensing. Further aspects character-

izing both sensors are the rates at which they can provide samples. Force/torque sensors usually provide very high rate measurements (1 kHz) that require little processing or interpretation for being used within a feedback control loop. Vision sensors, on the other side, run at much slower rates (typically 30 to 60 fps) and their measurements require high levels of processing for being usable in a control algorithm, which introduces non negligible latency in the availability of the control signal. Considering that the servo rate of a manipulator must be at least ten times greater than its mechanical resonance frequency [4], the restricted feedback rate in vision systems is found to be the main issue restricting the applicability of visual-servoing to real-world robot operations and its integration with conventional sensor feedback systems that generate commands directly for the low-level controller [5]. This is the case when one wants to take the manipulator's dynamics into account in the controller design. Finally, simplifying the models used to describe the system, such as neglecting certain phenomena that are difficult to model, e.g. friction, can in practice have significant effects on the system behaviour with consequent tracking performance degradation.

In this thesis we address several of these challenges such as:

- sensory integration, seen from the control perspective,
- increasing the sampling rate of visual information,
- mitigating the negative effects related to data latency, and
- system modelling and parameter identification.

1.3 Contributions

In this thesis we have contributed with the synthesis of controllers that realise the sensory integration of vision and force directly in the sensor space, i.e., in the visual feature space. These controllers achieve the highest degree of sensory integration, as they allow to control every possible direction with both sensory modalities.

In particular, one of the first contributions of this work was the mapping of the physical forces acting on the manipulator into the visual feature space. This allowed to derive, on one hand, an impedance control in the visual feature space, and on the other hand to extend the concept of compliant frame into the feature space. The former is an inverse dynamics controller, which allows the dynamics of the manipulator to be taken into account in the controller synthesis which do not require, in its simplest version, any force/torque sensor in order to operate.

Thanks to the mapping of forces into the visual feature space, it was possible to define an admittance law in such space, which we used to extend the capabilities of one of the vision/force control schemes in the literature.

Another aspect we tackled was the narrowing of the rate discrepancy between sensors. The use of a Bayesian estimation framework allowed us to virtually increase the availability of visual data at a higher rate and also to mitigate the negative effects due to the latency of computer vision algorithms. The proposed estimation models also allowed to reconstruct the velocity of a moving target observed by the camera. These estimators can be easily used in different control schemes.

The different control schemes developed in this work were validated through extensive simulations and experiments on a real platform in our laboratory. For what concerns the simulations, in this work we have also developed a physics co-simulation environment that allows not only to rapidly prototype visual servoing applications, but also to perform dynamic simulations with physical interaction with the environment. Thanks to this simulator, it was possible to study step by step all the problems and limitations of second-order visual servoing (a.k.a. Dynamic Visual Servoing) controllers. We could also design and test the estimators that allowed us to increase the camera's data rate, analyse its performance, and assess the impact of delayed measurements caused by visual feature extraction using computer vision techniques.

The realisation of this simulator was based on the model and estimated dynamic parameters of our manipulator. Another contribution of this thesis is indeed the accurate identification of a set of feasible dynamic parameters and a model of the dynamic friction of the robot joints thanks to the proposed framework for parameters retrieval.

1.4 Thesis structure

This thesis work is organized in four main parts. The first one (Part I) contains introductory background to the force control of robotic manipulators and visual control. Then, the state-of-the-art on the coupling of vision and force sensing is reviewed. The second part (Part II) highlights our contributions to the coupling of vision and force sensing in the visual feature space, allowing a robotic manipulator to actively achieve compliant motion, thanks to the model-based controllers and techniques developed and issued in the following author's publications [6, 7]. The third part (Part III) illustrates further contributions made on the dynamic model identification and parameter retrieval for robotic manipulators, which is

a key ingredient both for designing superior controllers which take into account the dynamics of the manipulator, as we did in Part II, and for performing more accurate simulations. This part of the thesis is based on the author's contributions [8, 9]. In the fourth part (Part IV) of the manuscript, we state the thesis conclusions and give some future research directions. We finally report complementary technical information in the appendices relative to the supplementary material of the author's paper [8].

Outline of Part I

This part contains the preliminaries and techniques involved in this thesis and a state-of-the-art review on the subject.

In Chapter 2 we provide all the necessary modelling background to deal with the control of manipulators. In particular, some elements of the kinematics and dynamics of a serial manipulator are given, as well as elements for the identification of dynamic parameters. Finally, classical force control schemes are discussed.

In Chapter 3 we give an introduction to visual servoing control techniques for both the cases of 2D and 3D features, showing some generalities and deriving the kinematic control law. We then present some more advanced schemes for both trajectory and target tracking and discuss the stability of the classic cases. We conclude with the derivation of the second-order visual servoing.

In Chapter 4 we review the different integrated vision-force control schemes present in the literature, highlighting their own strengths and weaknesses. This is followed by a discussion on how this thesis work places itself with respect to the state-of-the-art.

Outline of Part II

This part contains the author's contributions on the coupling of vision and force sensing at control level.

In Chapter 5, we present our results on second order visual servoing for both the cases of static and moving targets while we explore the possibility of using such controllers for physical interaction with the environment. We firstly derive the feature space impedance controllers for tracking both motionless or moving objects and then, we introduce two Extended Kalman Filters (EKF), based on the visual servoing geometric model, for increasing the rate of the visual information and estimating the target's velocity. This filters are derived for both Pose-Based

Visual Servoing (PBVS) and Image-Based Visual Servoing (IBVS) with image points as visual features. Simulations are carried out to validate the estimator performance during a dynamic Peg-in-Hole insertion task with a moving part. Experiments are also conducted on a real redundant manipulator with a low-cost wrist-mounted camera. Details on several issues encountered during the implementation are also discussed.

In Chapter 6 we illustrate the concept of compliant frame in the visual feature space and we propose a general framework, namely the *Extended External Hybrid* vision-force control scheme, for combining visual and force information in the visual feature space in a general fashion, since the treatment does not depend on the chosen visual features. Vision and force sensing are coupled in the feature space, avoiding both the convergence to a local minimum and the arising of inconsistencies at the actuation level. Any task space direction is simultaneously controlled by both vision and force. Compliance against interaction forces is achieved in feature space, along the features defining the visual task. Experiments on a real platform are carried out to evaluate the effectiveness of the proposed framework.

Outline of Part III

This part contains the author's contributions on the identification of robot models and the developed simulation tools.

In Chapter 7 we address the problem of extracting a feasible set of dynamic parameters characterizing the dynamics of a robot manipulator. We start by identifying through an ordinary least squares approach the dynamic coefficients that linearly parametrize the model. From these, we retrieve a set of feasible link parameters (mass, position of center of mass, inertia) that is fundamental for more realistic dynamic simulations or when implementing in real time robot control laws, using recursive Newton-Euler algorithms. The resulting problem is solved by means of an optimization method that incorporates constraints on the physical consistency of the dynamic parameters, including the triangle inequality of the link inertia tensors as well as other user-defined, possibly nonlinear, constraints. As use case, the approach is developed for our manipulator, *i.e.,* the popular Panda robot by Franka Emika, identifying for the first time its dynamic coefficients, an accurate joint friction model, and a set of feasible dynamic parameters. Validation of the identified dynamic model and of the retrieved feasible parameters is presented for the inverse dynamics problem using, respectively, a Lagrangian approach and Newton-Euler computations.

In Chapter 8 we present a new open-source simulator based on ROS and

CoppeliaSim for the Franka Emika Panda robot fully integrated in the ViSP [10] ecosystem, a powerful library for Visual-Servoing. The simulator features the dynamic model that has been accurately identified in the previous Chapter 7 from a real Panda robot, leading to more realistic simulations. The C++ API is an extended replica of the ViSP class of the real robot allowing to narrow the gap between simulation code and real control software deployment. Conceived as a multipurpose research simulation platform, it is well suited for fast prototyping and test of visual servoing applications as well as, in general, for any pedagogical purpose in robotic manipulators. All the software, models and CoppeliaSim scenes presented in this work are publicly available under free GPL-2.0 license.

Outline of Part IV

This part states the thesis conclusions and draws some possible future research lines.

Chapter 9 synthesizes the conclusions of this thesis work and summarises the main contributions made to the state-of-the-art. In addition, possible future research directions that would be worth exploring are discussed and some open questions are listed.

Outline of the Appendices

In Appendix A we compare the results of the dynamic parameter identification obtained with our method and with the classical approach that uses exciting trajectories. In Appendix B we show an example about the use of our optimization framework for the retrieval of a feasible set of dynamic parameters using nonlinear and conditional constraints on a 2R simulated robot.

1.5 Related publications

The following contributions were obtained during the course of this thesis work:

- C. Gaz, M. Cognetti, **A. Oliva**, P. Robuffo Giordano, and A. De Luca,"Dynamic identification of the Franka Emika Panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Robotics and Automation Lett.*, 2019
  Supplementary material and accompanying video:
  https://ieeexplore.ieee.org/document/8772145/media#media.

- **A. A. Oliva**, P. Robuffo Giordano, and F. Chaumette, "A general visual-impedance framework for effectively combining vision and force sensing in feature space", *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp.4441–4448, 2021.
  Video: https://www.youtube.com/watch?v=Qw0Dnhq2uRQ

- **A. A. Oliva**, E. Aertbeliën, J. de Schutter, P. Robuffo Giordano, and F. Chaumette, "Towards dynamic visual servoing for interaction control and moving targets," ICRA 2022 - *IEEE International Conference on Robotics and Automation*, May 2022, Philadelphia, United States. (to appear)
  Video: https://www.youtube.com/watch?v=fOSRnxd1dlE

- **A. A. Oliva**, F. Spindler, P. Robuffo Giordano, and F. Chaumette, "FrankaSim: A Franka Emika's Panda robot simulator with visual-servoing enabled capabilities," 2022 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. (submitted)
  Video: https://www.youtube.com/watch?v=Kxn3pXsK9h4&t=2s

Part I Preliminaries and State of the Art

Chapter 2

Force control of manipulators

Contents

2.1 Introduction

For many real-world robotic applications, interaction with the environment is a fundamental requirement and the ability of robots in managing this interaction often determines the successful execution of a task. For example, assembly or polishing require to control the exchanged forces at contact by regulating them to a specific value. The contact *wrench* between the end-effector and the environment is the most complete and effective quantity describing the state of the interaction, which is naturally described in the operational space [11].

For implementing an interaction control, either the exact knowledge of the location and geometry of the environment is required for accurately planning the task trajectory, or the robot needs to be equipped with force sensing capabilities to adapt to uncertainties and avoid high contact forces along the constrained directions. An effective way to deal with constrained motions is via active compliance, which can be achieved through impedance control [12] by imposing a *mass-spring-damper* behavior of the robot in contact with the environment. This scheme, as well as *compliance* or *stiffness* control, belongs to the category of *indirect* force control methods [13], since they achieve *open-loop* force control via *closed-loop* position control. Many other approaches have been explored in the past decades for including force sensing capabilities inside a motion control scheme, such as the widely studied Hybrid Position/Force [14] that is based on the task description [15], the parallel scheme [16] or, an approach based on external force loops [17]. The ability of these methods to perform *closed-loop* force control via explicit closure of a force feedback loop places them among the *direct* force control methods [13].

2.2 Manipulator kinematics

A manipulator can be mechanically described as a kinematic chain consisting of rigid bodies (links) connected by means of revolute or prismatic joints, which constitute its degrees of freedom (DoF). One end of the chain is bounded to a base (or the floor) while at the other extreme a tool is usually attached (see Figure 2.1). The overall motion of the structure is achieved through the composition of elementary motions of each link with respect to the previous one due to the motion of the q_i-th joint ($\forall i = 1, ..., n$). In order to manipulate the tool end point (end-effector), the description of the *pose* ($\boldsymbol{x}_e = [{}^f\boldsymbol{p}_e^\top \, {}^f\boldsymbol{\phi}_e^\top]^\top$), position (${}^f\boldsymbol{p}_e$) and orientation (${}^f\boldsymbol{\phi}_e$), of such point is required.

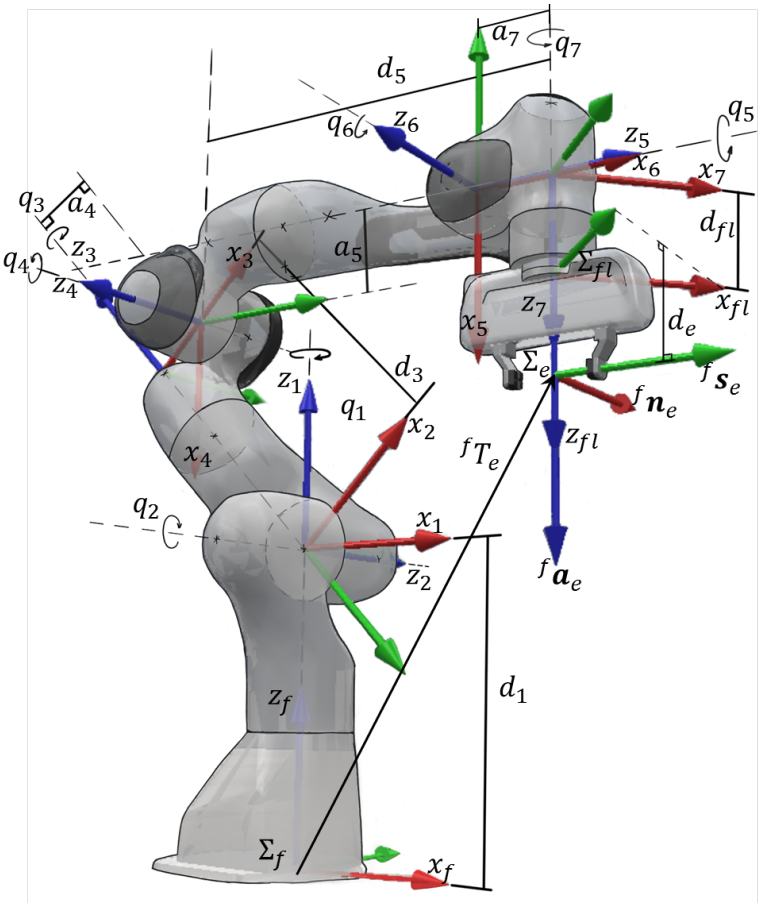
This is referred to as the *direct geometric model* [13]

$$\boldsymbol{x}_e = \mathcal{K}(\boldsymbol{q}) \tag{2.1}$$

being $\boldsymbol{q} \in \mathbb{R}^n$ the vector of joint variables. To the pose vector ($\boldsymbol{x}_e$) is associated an homogeneous transformation matrix ${}^f\boldsymbol{T}_e({}^f\boldsymbol{p}_e, {}^f\boldsymbol{\phi}_e) \in SE(3) = \mathbb{R}^3 \times SO(3)$ representing the pose of the end-effector frame (Σ_e) expressed in floor frame (Σ_f) (frame placed at the base of the manipulator):

$${}^f\boldsymbol{T}_e({}^f\boldsymbol{p}_e, {}^f\boldsymbol{\phi}_e) = \left[\begin{array}{c|c} {}^f\boldsymbol{R}_e({}^f\boldsymbol{\phi}_e) & {}^f\boldsymbol{p}_e \\ \hline 0\,0\,0 & 1 \end{array}\right] = \left[\begin{array}{c|c} {}^f\boldsymbol{n}_e\ {}^f\boldsymbol{s}_e\ {}^f\boldsymbol{a}_e & {}^f\boldsymbol{p}_e \\ \hline 0\,0\,0 & 1 \end{array}\right] \tag{2.2}$$

where ${}^f\boldsymbol{R}_e({}^f\boldsymbol{\phi}_e) \in SO(3)$ is the rotation matrix representing the orientation of frame Σ_e expressed in frame Σ_f coordinates. ${}^f\boldsymbol{\phi}_e$ is the vector form of the ori-



Modified Denavit-Hartenberg parameters

i	a_i	α_i	d_i	θ_i
1	0	0	d_1	q_1
2	0	$-\frac{\pi}{2}$	0	q_2
3	0	$\frac{\pi}{2}$	d_3	q_3
4	a_4	$\frac{\pi}{2}$	0	q_4
5	a_5	$-\frac{\pi}{2}$	d_5	q_5
6	0	$\frac{\pi}{2}$	0	q_6
7	a_7	$\frac{\pi}{2}$	0	q_7
fl	0	0	d_{fl}	0
e	0	0	d_e	$-\frac{\pi}{4}$

Figure 2.1 – Panda robot with its kinematic parameters according to the modified Denavit-Hartenberg convention: $d_1 = 0.333$ m, $d_3 = 0.316$ m, $d_5 = 0.384$ m, $d_{fl} = 0.107$ m, $d_e = 0.1034$ m, $a_4 = 0.0825$ m, $a_5 = -a_4$ m, $a_7 = 0.088$ m.

entation representation which could be either one of the 12 possible *Euler angles* representations, or the *Axis/angle*, or again, the *unit quaternion* representation.

Each column of the ${}^f\boldsymbol{R}_e({}^f\boldsymbol{\phi}_e)$ is a unit vector of the end-effector frame Σ_e, which is placed conveniently from case to case. When the tool is a two finger gripper, the origin of the frame is placed at the mean point between the fingers and the unit vectors are chosen such that ${}^f\boldsymbol{s}_e$ belongs to the plane in which the fingers *slide*, ${}^f\boldsymbol{a}_e$ points in the *approach* direction towards the object to be grasped, while ${}^f\boldsymbol{n}_e$ is *normal* to the sliding plane (See Figure 2.1).

It is evident that the pose of the end-effetor frame depends on the manipulator's *configuration* ($\boldsymbol{q} \in \mathbb{R}^n$) and it can be systematically computed as a recursive product of homogeneous transformation matrices (${}^{i-1}\boldsymbol{T}_i(q_i)$) from the end-effector to the base, each of which is a function of a single joint variable (q_i). Therefore, the overall coordinate transformation that expresses the pose of the end-effector in base frame for the $n = 7$ joint manipulator in Figure 2.1 is given by

$$ {}^f\boldsymbol{T}_e(\boldsymbol{q}) = {}^f\boldsymbol{T}_1(q_1)\,{}^1\boldsymbol{T}_2(q_2)\ldots{}^6\boldsymbol{T}_7(q_7)\,{}^7\boldsymbol{T}_{fl}\,{}^{fl}\boldsymbol{T}_e \tag{2.3} $$

in which the two rightmost transformations, *i.e.,* from the last joint to the flange (${}^7\boldsymbol{T}_{fl}$) and from flange to end-effector frame (${}^{fl}\boldsymbol{T}_e$), are constant. The reference frames along the kinematic chain are often placed following the *Denavit-Hartenberg* [18] convention (DH). For the Panda robot from Franka Emika in our example, the frames placement follows the *Modified Denavit-Hartenberg* [19] convention, whose DH parameters are also reported in Figure 2.1.

The space in which the pose vector ($\boldsymbol{x}_e$) is defined is usually the space in which the operations (or tasks), that are requested to the manipulator, are specified; therefore it is called the *operational space* [11], while the space in which the vector of joint variables $\boldsymbol{q} = [q_1 \ldots q_n]^\top$ is defined is referred to as the *joint space* or *configuration space*.

The direct geometric model equation, in both its forms (2.1) or (2.2), expresses the (unique) relationship between the joint variables and the end-effector's pose. The inverse problem, on the other hand, refers to the problem of retrieving the vector of joint variables once an end-effector pose is given. The resolution of this problem is critical in order to transform the motion specifications defined in the operational space for the end-effector, into the corresponding motion in the joint space, allowing the required motion to be realized on the mechanical structure. Furthermore, the inverse problem has no unique solution for redundant manipulators (or, more broadly, when the number of joints is greater that the DoF of the task space).

2.2.1 Differential kinematics

We have seen the relationship between the end-effector's posture and the joint variables. In this subsection, we will focus on their differential relationship, or the relationship that exists between the end-effector's linear (${}^f\dot{\boldsymbol{p}}_e \in \mathbb{R}^3$) and angular (${}^f\boldsymbol{\omega}_e \in \mathbb{R}^3$) velocity with the joint velocities $\dot{\boldsymbol{q}} \in \mathbb{R}^n$. Both relationships are linear with respect to the joint velocities as follows:

$$\begin{aligned} {}^f\dot{\boldsymbol{p}}_e &= \boldsymbol{J}_{e_P}(\boldsymbol{q})\dot{\boldsymbol{q}} \\ {}^f\boldsymbol{\omega}_e &= \boldsymbol{J}_{e_O}(\boldsymbol{q})\dot{\boldsymbol{q}} \end{aligned} \tag{2.4}$$

being $\boldsymbol{J}_{e_P} \in \mathbb{R}^{3\times n}$ and $\boldsymbol{J}_{e_O} \in \mathbb{R}^{3\times n}$ respectively the matrices relative to the contribution of the joint velocities to the linear (${}^f\dot{\boldsymbol{p}}_e$) and angular (${}^f\boldsymbol{\omega}_e$) velocities of the end-effector. One can then write the *differential kinematic equation* in compact form as

$${}^f\boldsymbol{v}_e = \begin{bmatrix} {}^f\dot{\boldsymbol{p}}_e \\ {}^f\boldsymbol{\omega}_e \end{bmatrix} = \begin{bmatrix} {}^f\mathbf{v}_e \\ {}^f\boldsymbol{\omega}_e \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_{e_P}(\boldsymbol{q}) \\ \boldsymbol{J}_{e_O}(\boldsymbol{q}) \end{bmatrix} \dot{\boldsymbol{q}} = \boldsymbol{J}_e(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2.5}$$

where $\boldsymbol{J}_e(q) \in \mathbb{R}^{6\times n}$ is the configuration dependent *geometric Jacobian* matrix of the manipulator.

It is sometimes convenient to express the end-effector's velocity with respect to another frame u of the kinematic chain instead of the base frame; it is often the case for the end-effector frame ($u = e$). If the rotation matrix between the base and the desired frame (u) is known (${}^f\boldsymbol{R}_u$) and using the relationship between velocities expressed in two different frames

$$\begin{bmatrix} {}^u\mathbf{v}_e \\ {}^u\boldsymbol{\omega}_e \end{bmatrix} = \begin{bmatrix} {}^f\boldsymbol{R}_u{}^\top & \boldsymbol{O} \\ \boldsymbol{O} & {}^f\boldsymbol{R}_u{}^\top \end{bmatrix} \begin{bmatrix} {}^f\mathbf{v}_e \\ {}^f\boldsymbol{\omega}_e \end{bmatrix} = {}^f\mathbb{R}_u^\top \begin{bmatrix} {}^f\mathbf{v}_e \\ {}^f\boldsymbol{\omega}_e \end{bmatrix} \tag{2.6}$$

where $\boldsymbol{O}$ is the matrix of all zero elements of proper dimension. Exploiting this relationship, one can obtain the end-effector linear and angular velocities in its own frame from the joint velocities as

$${}^e\boldsymbol{v}_e = \begin{bmatrix} {}^e\mathbf{v}_e \\ {}^e\boldsymbol{\omega}_e \end{bmatrix} = {}^f\mathbb{R}_e^\top \boldsymbol{J}_e(\boldsymbol{q})\dot{\boldsymbol{q}} = {}^e\boldsymbol{J}_e(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2.7}$$

The method for computing the geometric Jacobian essentially follows a geometric procedure that identifies the individual contributions of each joint velocity to the linear and angular velocity components of the end-effector.

Another possibility to compute the Jacobian consists in deriving it directly from the *direct geometric model* functions ($\mathcal{K}(\boldsymbol{q})$) as

$$\begin{aligned} {}^f\dot{\boldsymbol{p}}_e &= \frac{\partial\, {}^f\boldsymbol{p}_e}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} = \boldsymbol{J}_{e_P}(\boldsymbol{q})\dot{\boldsymbol{q}} \\ {}^f\dot{\boldsymbol{\phi}}_e &= \frac{\partial\, {}^f\boldsymbol{\phi}_e}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} = \boldsymbol{J}_{e_\phi}(\boldsymbol{q})\dot{\boldsymbol{q}} \end{aligned} \tag{2.8}$$

the linear velocity of the end-effector can be obtained as the time derivative of its position vector in the base frame (${}^f\dot{\boldsymbol{p}}_e$), while for the orientation, the time derivative of the minimal orientation representation does not coincide, in general, with the vector of angular velocities (${}^f\dot{\boldsymbol{\phi}}_e \neq {}^f\boldsymbol{\omega}_e$). Furthermore, the computation of the Jacobian $\boldsymbol{J}_{e_\phi}(\boldsymbol{q}) = \frac{\partial\, {}^f\boldsymbol{\phi}_e}{\partial \boldsymbol{q}}$ is not straightforward since the function ${}^f\boldsymbol{\phi}_e$ is not usually available in explicit form and its computation requires to pass through the elements of the relative rotation matrix (${}^f\boldsymbol{R}_e({}^f\boldsymbol{\phi}_e)$). Formally deriving the direct geometric model equation (2.1) one yields to the following *differential kinematic equation*

$$\dot{\boldsymbol{x}}_e = \begin{bmatrix} {}^f\dot{\boldsymbol{p}}_e \\ {}^f\dot{\boldsymbol{\phi}}_e \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_{e_P}(\boldsymbol{q}) \\ \boldsymbol{J}_{e_\phi}(\boldsymbol{q}) \end{bmatrix} \dot{\boldsymbol{q}} = \boldsymbol{J}_A(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2.9}$$

where

$$\boldsymbol{J}_A(\boldsymbol{q}) = \frac{\partial\, \mathcal{K}(\boldsymbol{q})}{\partial \boldsymbol{q}} \tag{2.10}$$

is called the *analytic Jacobian*, which results to be different from the *geometric Jacobian* $\boldsymbol{J}_e(\boldsymbol{q})$ computed using the geometric procedure. Exploiting the existing relationship between angular velocity and time derivative of a minimal orientation representation

$${}^f\boldsymbol{\omega}_e = \boldsymbol{\Gamma}({}^f\boldsymbol{\phi}_e)\, {}^f\dot{\boldsymbol{\phi}}_e \tag{2.11}$$

where $\boldsymbol{\Gamma}({}^f\boldsymbol{\phi}_e)$ is a suitable representation dependent transformation matrix, it is possible to find the link between the two Jacobians. Taking into account (2.5) and (2.9)

$${}^f\boldsymbol{v}_e = \begin{bmatrix} \mathbb{I}_3 & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{\Gamma}({}^f\boldsymbol{\phi}_e) \end{bmatrix} \dot{\boldsymbol{x}}_e = \boldsymbol{\Gamma}_A({}^f\boldsymbol{\phi}_e)\dot{\boldsymbol{x}}_e = \boldsymbol{\Gamma}_A({}^f\boldsymbol{\phi}_e)\boldsymbol{J}_A(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2.12}$$

where $\mathbb{I}_3$ is the 3×3 identity matrix. We finally obtain

$$\boldsymbol{J}_e(\boldsymbol{q}) = \boldsymbol{\Gamma}_A({}^f\boldsymbol{\phi}_e)\boldsymbol{J}_A(\boldsymbol{q}) \tag{2.13}$$

2.2.1.1 Second-order differential kinematics

Kinematic inversion algorithms are first-order algorithms in the sense that they allow the inversion of a trajectory specified in terms of end-effector position and

velocity into the equivalent position and velocity of the joints. As we will see later, inverting a trajectory specified in terms of position, velocity, and acceleration in operational space may be necessary for control purposes. The manipulator, on the other hand, is a naturally second-order mechanical system, as the dynamic model will show. The derivation of the differential kinematic equation (2.5) allows to write

$$ {}^f\dot{\boldsymbol{v}}_e = \dot{\boldsymbol{J}}_e(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{J}_e(\boldsymbol{q})\ddot{\boldsymbol{q}} \tag{2.14} $$

while the derivation of equation (2.9) yields

$$ \ddot{\boldsymbol{x}}_e = \dot{\boldsymbol{J}}_A(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{J}_A(\boldsymbol{q})\ddot{\boldsymbol{q}} \tag{2.15} $$

2.2.2 Statics

The goal of this subsection is to find the relationship between the forces (${}^f\boldsymbol{f}_e \in \mathbb{R}^3$) and moments (${}^f\boldsymbol{\mu}_e \in \mathbb{R}^3$) acting on the end-effector (the *wrench* $\boldsymbol{h}_e = [{}^f\boldsymbol{f}_e{}^\top \ {}^f\boldsymbol{\mu}_e{}^\top]^\top$) and the forces/torques ($\boldsymbol{\tau} \in \mathbb{R}^n$) at the joints (force for *prismatic*, torque for *revolute* joints).

The considered manipulators are mechanical system with time-independent *holonomic constraints* whose configurations only depend on the joint variables $\boldsymbol{q}$ and not explicitly on time. This means that the *virtual* displacements (indicated with δ) of the structure coincide with the *elementary* displacements (indicated with d). Applying therefore the *Virtual Work Principle* we can obtain the desired relation.

Let us consider the elementary work done by the two force systems. On one hand, one has the force/torques acting on the joints, while in the other hand, one has to split the contribution to the work performed by the forces and the moments at the origin of the end-effector frame. The elementary works are then:

$$ \begin{aligned} dW_\tau &= \boldsymbol{\tau}^\top d\boldsymbol{q} \\ dW_h &= {}^f\boldsymbol{f}_e{}^\top d^f\boldsymbol{p}_e + {}^f\boldsymbol{\mu}_e{}^\top {}^f\boldsymbol{\omega}_e\, dt \end{aligned} \tag{2.16} $$

where dW_τ and dW_h are respectively the elementary work performed by the joint torques $\boldsymbol{\tau}$ and the wrench $\boldsymbol{h}_e$. $d^f\boldsymbol{p}_e$ and ${}^f\boldsymbol{\omega}_e\, dt$[1] are the linear and the angular displacements. Taking into account the differential kinematic equation (2.5)

$$ dW_h = {}^f\boldsymbol{f}_e{}^\top \boldsymbol{J}_{e_P}(\boldsymbol{q})d\boldsymbol{q} + {}^f\boldsymbol{\mu}_e{}^\top \boldsymbol{J}_{e_O}(\boldsymbol{q})d\boldsymbol{q} = \boldsymbol{h}_e^\top \boldsymbol{J}_e(\boldsymbol{q})d\boldsymbol{q} \tag{2.17} $$

[1] Due to the integrability problems related to the angular displacement it has been indicated by ${}^f\boldsymbol{\omega}_e\, dt$.

Because the virtual and elementary displacements coincide, the virtual work equals the elementary work, then

$$\begin{aligned} \delta W_\tau &= \boldsymbol{\tau}^\top \delta \boldsymbol{q} \\ \delta W_h &= \boldsymbol{h}_e^\top \boldsymbol{J}_e(\boldsymbol{q}) \delta \boldsymbol{q} \end{aligned} \tag{2.18}$$

and for the *Virtual Work Principle*, the manipulator is in *static equilibrium* if and only if

$$\delta W_\tau = \delta W_h \quad \forall \delta \boldsymbol{q} \tag{2.19}$$

yielding to the well-known relation

$$\boldsymbol{\tau} = \boldsymbol{J}_e^\top(\boldsymbol{q}) \boldsymbol{h}_e \tag{2.20}$$

which by virtue of the relation (2.13) can also be expressed as

$$\boldsymbol{\tau} = \boldsymbol{J}_e^\top(\boldsymbol{q}) \boldsymbol{h}_e = (\boldsymbol{\Gamma}_A({}^f\boldsymbol{\phi}_e)\boldsymbol{J}_A(\boldsymbol{q}))^\top \boldsymbol{h}_e = \boldsymbol{J}_A(\boldsymbol{q})^\top \boldsymbol{h}_A \tag{2.21}$$

where of course

$$\boldsymbol{h}_A = \boldsymbol{\Gamma}_A({}^f\boldsymbol{\phi}_e)^\top \boldsymbol{h}_e \tag{2.22}$$

2.2.3 Twist and Wrench transformations

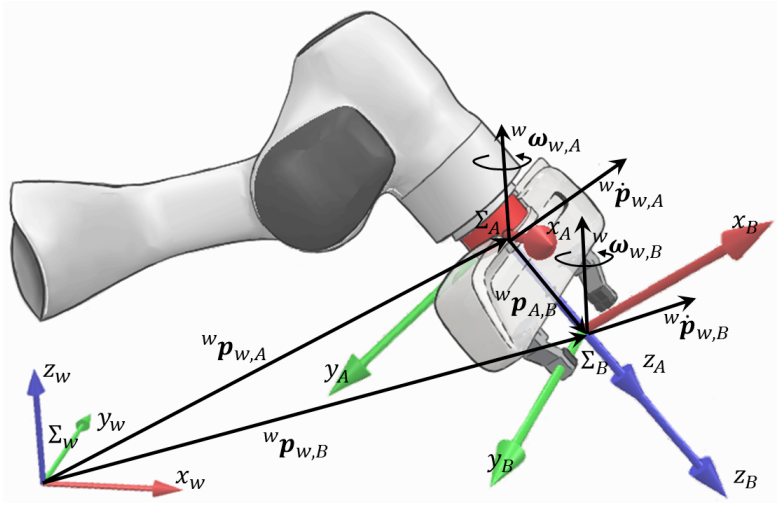


Figure 2.2 – Linear and angular velocity representations in different coordinate frames on the same rigid-body. The *Wrench-transformation* is used to express the measured contact *wrench* on the Force/Torque sensor frame (Σ_A) into the end-effector frame (Σ_B).

Let consider a fixed frame of reference Σ_w and a rigid-body in motion with respect to such frame. Let Σ_A and Σ_B be two coordinate frames attached to the

rigid-body as shown in Figure 2.2. The linear and angular velocity relationship between frames Σ_A and Σ_B with respect to the fixed frame Σ_w are given by:

$$\begin{aligned} {}^{w}\boldsymbol{\omega}_{w,B} &= {}^{w}\boldsymbol{\omega}_{w,A} \\ {}^{w}\dot{\boldsymbol{p}}_{w,B} &= {}^{w}\dot{\boldsymbol{p}}_{w,A} + {}^{w}\boldsymbol{\omega}_{w,A} \times {}^{w}\boldsymbol{p}_{A,B} \end{aligned} \tag{2.23}$$

which, exploiting the matrix form of the cross-product, can be written in compact form as

$$\begin{bmatrix} {}^{w}\dot{\boldsymbol{p}}_{w,B} \\ {}^{w}\boldsymbol{\omega}_{w,B} \end{bmatrix} = \begin{bmatrix} \mathbb{I} & -[{}^{w}\boldsymbol{p}_{A,B}]_{\times} \\ \boldsymbol{O} & \mathbb{I} \end{bmatrix} \begin{bmatrix} {}^{w}\dot{\boldsymbol{p}}_{w,A} \\ {}^{w}\boldsymbol{\omega}_{w,A} \end{bmatrix} \tag{2.24}$$

being $[\cdot]_{\times}$ the 3×3 *skew-symmetric* matrix operator. If the vectors are instead expressed in their own frames

$$\begin{gathered} {}^{w}\boldsymbol{p}_{A,B} = {}^{w}\boldsymbol{R}_{A}\,{}^{A}\boldsymbol{p}_{A,B} \\ {}^{w}\dot{\boldsymbol{p}}_{w,A} = {}^{w}\boldsymbol{R}_{A}\,{}^{A}\dot{\boldsymbol{p}}_{w,A} \qquad {}^{w}\dot{\boldsymbol{p}}_{w,B} = {}^{w}\boldsymbol{R}_{B}\,{}^{B}\dot{\boldsymbol{p}}_{w,B} = {}^{w}\boldsymbol{R}_{A}\,{}^{A}\boldsymbol{R}_{B}\,{}^{B}\dot{\boldsymbol{p}}_{w,B} \\ {}^{w}\boldsymbol{\omega}_{w,A} = {}^{w}\boldsymbol{R}_{A}\,{}^{A}\boldsymbol{\omega}_{w,A} \qquad {}^{w}\boldsymbol{\omega}_{w,B} = {}^{w}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{\omega}_{w,B} = {}^{w}\boldsymbol{R}_{A}\,{}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{\omega}_{w,B} \end{gathered}$$

and taking into account equation (2.24) and the properties of the *skew-symmetric* matrix operator [13], one yields the *Twist-transformation*

$${}^{B}\boldsymbol{v}_{w,B} = \begin{bmatrix} {}^{B}\dot{\boldsymbol{p}}_{w,B} \\ {}^{B}\boldsymbol{\omega}_{w,B} \end{bmatrix} = \begin{bmatrix} {}^{B}\boldsymbol{R}_{A} & -{}^{B}\boldsymbol{R}_{A}[{}^{A}\boldsymbol{p}_{A,B}]_{\times} \\ \boldsymbol{O} & {}^{B}\boldsymbol{R}_{A} \end{bmatrix} \begin{bmatrix} {}^{A}\dot{\boldsymbol{p}}_{w,A} \\ {}^{A}\boldsymbol{\omega}_{w,A} \end{bmatrix} = {}^{B}\mathbb{T}_{A}\,{}^{A}\boldsymbol{v}_{w,A} \tag{2.25}$$

being ${}^{B}\mathbb{T}_{A}$ the *Twist-transformation* matrix, which is a proper Jacobian matrix that expresses the linear and angular velocities from a reference frame into another. Due to the *kineto-static* duality [13], one can directly obtain the *Wrench-transformation* as

$${}^{A}\boldsymbol{h}_{A} = \begin{bmatrix} {}^{A}\boldsymbol{f}_{A} \\ {}^{A}\boldsymbol{\mu}_{A} \end{bmatrix} = \begin{bmatrix} {}^{A}\boldsymbol{R}_{B} & \boldsymbol{O} \\ [{}^{A}\boldsymbol{p}_{A,B}]_{\times}^{A}\boldsymbol{R}_{B} & {}^{A}\boldsymbol{R}_{B} \end{bmatrix} \begin{bmatrix} {}^{B}\boldsymbol{f}_{B} \\ {}^{B}\boldsymbol{\mu}_{B} \end{bmatrix} = {}^{B}\mathbb{T}_{A}^{\top}\,{}^{B}\boldsymbol{h}_{B} \tag{2.26}$$

being the *Wrench-transformation* matrix ${}^{A}\mathbb{F}_{B} = {}^{B}\mathbb{T}_{A}^{\top}$.

2.3 Dynamic model

The dynamic model of a robotic arm can be written using the *Lagrange* formulation in the joint space as described in [13]:

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{F}_{v}\dot{\boldsymbol{q}} + \boldsymbol{F}_{s}\boldsymbol{sgn}(\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{J}_{e}^{\top}(\boldsymbol{q})\boldsymbol{h}_{e} = \boldsymbol{\tau} \tag{2.27}$$

where $\dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} \in \mathbb{R}^{n}$ are respectively the generalized joint velocities and accelerations. $\boldsymbol{B}(\boldsymbol{q}) \in \mathbb{R}^{n \times n}$ is the symmetric and positive definite inertia matrix,

$\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n \times n}$ is the matrix of centrifugal and Coriolis effects, $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^n$ is the configuration dependent vector of gravitational forces, $\boldsymbol{\tau}_e = \boldsymbol{J}_e^\top \boldsymbol{h}_e \in \mathbb{R}^n$ is the joint vector corresponding to the external *wrench* $\boldsymbol{h}_e \in \mathbb{R}^6$ acting on the end-effector frame, $\boldsymbol{\tau} \in \mathbb{R}^n$ is the vector of joint actuation torques. Finally, $\boldsymbol{F}_v \dot{\boldsymbol{q}} \in \mathbb{R}^n$ and $\boldsymbol{F}_s \boldsymbol{sgn}(\dot{\boldsymbol{q}}) \in \mathbb{R}^n$ represents the joint vectors of viscous and coulomb friction.

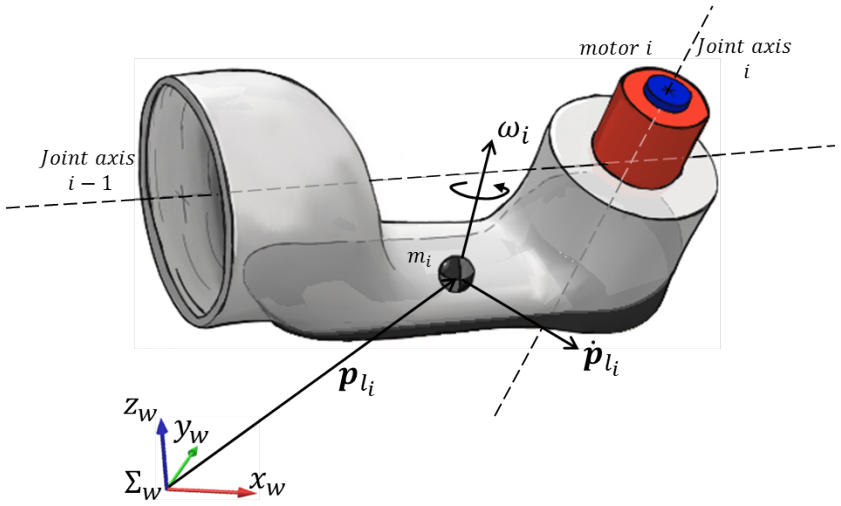


Figure 2.3 – Kinematic characterization of the i-th link for the Lagrangian formulation.

2.3.1 Parameter identification

The dynamic model in the form (2.27) includes typically nonlinear functions of $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$ and the dynamic parameters described here in detail further. For each link ℓ_i, $i = 1, \ldots, n$, let m_i be its mass and let

$$
{}^i\boldsymbol{r}_{i,ci} = \begin{bmatrix} c_{ix} \\ c_{iy} \\ c_{iz} \end{bmatrix}, \quad {}^i\boldsymbol{J}_{\ell_i} = \begin{bmatrix} J_{ixx} & J_{ixy} & J_{ixz} \\ J_{ixy} & J_{iyy} & J_{iyz} \\ J_{ixz} & J_{iyz} & J_{izz} \end{bmatrix}, \tag{2.28}
$$

be the position of the center of mass and the symmetric inertia tensor with respect to the i-th link frame, respectively, see Figure 2.3.

At this stage, if we collect the dynamic parameters of all the robot links in the three vectors

$$
\begin{aligned}
\boldsymbol{p}_1 &= \begin{bmatrix} m_1 & \ldots & m_n \end{bmatrix}^\top, \\
\boldsymbol{p}_2 &= \begin{bmatrix} c_{1x}m_1 & c_{1y}m_1 & c_{1z}m_1 & \ldots & c_{nx}m_n & c_{ny}m_n & c_{nz}m_n \end{bmatrix}^\top, \\
\boldsymbol{p}_3 &= \begin{bmatrix} \mathcal{J}_1^\top & \ldots & \mathcal{J}_n^\top \end{bmatrix}^\top,
\end{aligned} \tag{2.29}
$$

with $\boldsymbol{p}_1 \in \mathbb{R}^n$, $\boldsymbol{p}_2 \in \mathbb{R}^{3n}$, $\boldsymbol{p}_3 \in \mathbb{R}^{6n}$ and

$$\mathcal{J}_i = \begin{bmatrix} J_{ixx} & J_{ixy} & J_{ixz} & J_{iyy} & J_{iyz} & J_{izz} \end{bmatrix}^\top, \tag{2.30}$$

it is possible to rearrange (2.27) as

$$\boldsymbol{Y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\, \boldsymbol{\pi}(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3) = \boldsymbol{\tau}, \tag{2.31}$$

where the vector $\boldsymbol{\pi}(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3) = \begin{bmatrix} \boldsymbol{p}_1^\top & \boldsymbol{p}_2^\top & \boldsymbol{p}_3^\top \end{bmatrix}^\top \in \mathbb{R}^p$ [20]. Then, $\boldsymbol{\pi}$ appears linearly in the dynamic model (2.31), multiplied by the regressor matrix $\boldsymbol{Y}$ of known time-varying functions.

The dynamic identification procedure is performed by collecting $M \gg n\ p$ joint torque samples as well as M joint position samples, while the joint velocity and the acceleration are computed by off-line differentiation. For each numerical sample $(\boldsymbol{\tau}_k, \boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k)$, with $k = 1, \ldots, M$, we have

$$\boldsymbol{Y}_k(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k)\boldsymbol{\pi} = \boldsymbol{\tau}_k. \tag{2.32}$$

By stacking these quantities in vectors and matrices, one has

$$\overline{\boldsymbol{Y}}\boldsymbol{\pi} = \overline{\boldsymbol{\tau}}, \tag{2.33}$$

with $\overline{\boldsymbol{\tau}} \in \mathbb{R}^{Mn}$ and $\overline{\boldsymbol{Y}} \in \mathbb{R}^{Mn \times p}$. According to [21], we can prune the stacked regressor $\overline{\boldsymbol{Y}}$ so as to obtain a matrix with full column rank $\overline{\boldsymbol{Y}}_R$, and then identify the dynamic coefficients by solving an ordinary least-squares (OLS) problem via pseudoinversion

$$\hat{\boldsymbol{\pi}}_R = \overline{\boldsymbol{Y}}_R^\dagger \overline{\boldsymbol{\tau}}. \tag{2.34}$$

With the solution $\hat{\boldsymbol{\pi}}_R \in \mathbb{R}^\rho$ of regrouped dynamic parameters, i.e., the dynamic coefficients, we can provide a joint torque estimate as

$$\hat{\boldsymbol{\tau}} = \boldsymbol{Y}_R(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\hat{\boldsymbol{\pi}}_R \tag{2.35}$$

for validation on any new motion $\boldsymbol{q}(t)$. Finally, following [20], one can extract from the identified vector $\hat{\boldsymbol{\pi}}_R$ a feasible set of dynamic parameters $\hat{\boldsymbol{p}} = (\hat{\boldsymbol{p}}_1, \hat{\boldsymbol{p}}_2, \hat{\boldsymbol{p}}_3)$ (not necessarily the true ones) such that $\boldsymbol{\pi}_R(\hat{\boldsymbol{p}}_1, \hat{\boldsymbol{p}}_2, \hat{\boldsymbol{p}}_3) = \hat{\boldsymbol{\pi}}_R$ and the upper/lower bounds on the components of $\boldsymbol{p}_i$, $i = 1, 2, 3$, are also satisfied. However, the triangle inequality constraint of inertia tensors is not taken into account in [20], as instead done in Section 7.4.

Details on the physical consistency and the triangle inequality are provided in the next subsection.

2.3.2 Physical consistency of the dynamic parameters

The obtained estimation of the dynamic coefficients vector $\hat{\boldsymbol{\pi}}_R$ might be possibly physically inconsistent (e.g., a negative link mass), and this can be caused, for instance, by modeling errors or by noisy measurements. Recent works [22–24] highlighted these physical constraints and provided frameworks to consider them during the identification phase, by solving linear constrained optimization problems using the following cost function:

$$\min_{\boldsymbol{\pi}} f(\boldsymbol{\pi}) = \|\overline{\boldsymbol{Y}}\boldsymbol{\pi} - \overline{\boldsymbol{\tau}}\|. \tag{2.36}$$

Physical constraints regard the mass of each link, which has to be positive, and the barycentric inertia tensor of each link, which has to be positive definite. That is, for each link ℓ_i of the manipulator, one has:

$$m_i > 0 \tag{2.37}$$

and

$$\boldsymbol{I}_{\ell_i} = \begin{bmatrix} I_{ixx} & I_{ixy} & I_{ixz} \\ I_{ixy} & I_{iyy} & I_{iyz} \\ I_{ixz} & I_{iyz} & I_{izz} \end{bmatrix} \succ 0, \tag{2.38}$$

where $\boldsymbol{I}_{\ell_i}$ is the inertia tensor of link ℓ_i with respect to its center of mass. Moreover, it is always possible to express the barycentric inertia tensor in a diagonal form, exploiting a particular rotation matrix $\bar{\boldsymbol{R}}_i$, such as

$$\boldsymbol{I}_{\ell_i} = \bar{\boldsymbol{R}}_i \bar{\boldsymbol{I}}_{\ell_i} \bar{\boldsymbol{R}}_i^T, \tag{2.39}$$

where $\bar{\boldsymbol{I}}_{\ell_i}$ is the diagonal inertia tensor. Since the diagonal elements ($\bar{I}_{i,x}$, $\bar{I}_{i,y}$, $\bar{I}_{i,z}$) of $\bar{\boldsymbol{I}}_{\ell_i}$ are also the eigenvalues of $\boldsymbol{I}_{\ell_i}$, condition (2.38) can be rewritten as

$$\bar{I}_{i,x} > 0 \quad , \quad \bar{I}_{i,y} > 0 \quad , \quad \bar{I}_{i,z} > 0 \, . \tag{2.40}$$

These three inequalities are however included in the following triangle inequality:

$$\begin{cases} \bar{I}_{i,x} + \bar{I}_{i,y} > \bar{I}_{i,z} \\ \bar{I}_{i,y} + \bar{I}_{i,z} > \bar{I}_{i,x} \\ \bar{I}_{i,z} + \bar{I}_{i,x} > \bar{I}_{i,y} \end{cases} \tag{2.41}$$

which, with simple manipulations [24], leads to the following condition on $\boldsymbol{I}_{\ell_i}$:

$$\frac{\mathrm{tr}(\boldsymbol{I}_{\ell_i})}{2} - \lambda_{\max}(\boldsymbol{I}_{\ell_i}) > 0, \tag{2.42}$$

since $\bar{I}_{i,x} + \bar{I}_{i,y} + \bar{I}_{i,z} = \mathrm{tr}(\boldsymbol{I}_{\ell_i})$ and denoting with $\mathrm{tr}(\boldsymbol{I}_{\ell_i})$ and $\lambda_{\max}(\boldsymbol{I}_{\ell_i})$, respectively, the trace and the maximum eigenvalue of the inertia tensor $\boldsymbol{I}_{\ell_i}$.

Therefore, in order to guarantee physical consistency of the dynamic parameters, conditions (2.37) and (2.42) must be satisfied for each link and are therefore added as constraints to the optimization problem (see Section 7.4).

2.3.3 Dynamic model in operational space

To develop a dynamic model in operational space capable of describing the behavior of redundant and non-redundant manipulation structures, the dynamic model in joint space should be used as a starting point for the description, as its application is quite general. To that end, if one solves equation (2.27) in terms of joint accelerations while ignoring for simplicity the joint friction torques, one obtains

$$\ddot{\boldsymbol{q}} = -\boldsymbol{B}(\boldsymbol{q})^{-1}\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{B}(\boldsymbol{q})^{-1}\boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{B}(\boldsymbol{q})^{-1}\boldsymbol{J}_e^{\top}(\boldsymbol{q})(\boldsymbol{\gamma}_e - \boldsymbol{h}_e) \tag{2.43}$$

in which in virtue of equation (2.20), the contribution to the joint actuation torques has been expressed in terms of the equivalent forces at the end-effector as $\boldsymbol{\tau} = \boldsymbol{J}_e^{\top}(\boldsymbol{q})\boldsymbol{\gamma}_e$. We recall that the term $\boldsymbol{h}_e$ expresses the contribution to the end-effector forces due to contact with the environment while $\boldsymbol{\gamma}_e$ is the contribution to the end-effector forces due to the motor actuation torques.

The second-order differential kinematic equation (2.15) expresses the relationship between joint and operational space accelerations through the robot analytic Jacobian $\boldsymbol{J}_A$ while in the Lagrangian equation of motion (2.27) appears the geometric Jacobian $\boldsymbol{J}_e$. Similarly to relation (2.22), it is possible to write

$$\boldsymbol{\gamma}_A = \boldsymbol{\Gamma}_A({}^f\boldsymbol{\phi}_e)^{\top}\boldsymbol{\gamma}_e \tag{2.44}$$

Replacing (2.43) into the second-order differential kinematic equation (2.15) one obtains (omitting the dependencies on $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ for the sake of notation simplicity)

$$\ddot{\boldsymbol{x}}_e = \dot{\boldsymbol{J}}_A\dot{\boldsymbol{q}} - \boldsymbol{J}_A\boldsymbol{B}^{-1}\boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{J}_A\boldsymbol{B}^{-1}\boldsymbol{g} + \boldsymbol{J}_A\boldsymbol{B}^{-1}\boldsymbol{J}_A^{\top}(\boldsymbol{\gamma}_A - \boldsymbol{h}_A) \tag{2.45}$$

that rearranged and naming the quantities ($\dot{\boldsymbol{q}} = \boldsymbol{J}_A^{\dagger}\dot{\boldsymbol{x}}_e$)

$$\boldsymbol{B}_A = (\boldsymbol{J}_A\boldsymbol{B}^{-1}\boldsymbol{J}_A^{\top})^{-1} \tag{2.46a}$$

$$\boldsymbol{C}_A\dot{\boldsymbol{x}}_e = \boldsymbol{B}_A(\boldsymbol{J}_A\boldsymbol{B}^{-1}\boldsymbol{C} - \dot{\boldsymbol{J}}_A)\boldsymbol{J}_A^{\dagger}\dot{\boldsymbol{x}}_e \tag{2.46b}$$

$$\boldsymbol{g}_A = \boldsymbol{B}_A\boldsymbol{J}_A\boldsymbol{B}^{-1}\boldsymbol{g} \tag{2.46c}$$

yields to

$$\boldsymbol{B}_A(\boldsymbol{x}_e)\ddot{\boldsymbol{x}}_e + \boldsymbol{C}_A(\boldsymbol{x}_e,\dot{\boldsymbol{x}}_e)\dot{\boldsymbol{x}}_e + \boldsymbol{g}_A(\boldsymbol{x}_e) + \boldsymbol{h}_A = \boldsymbol{\gamma}_A \tag{2.47}$$

The obtained dynamic model in operational space is formally analogous to the one in joint space.

2.4 Force Control Schemes

The ability to manage the physical contact between a robot and its environment is a critical necessity for the accomplishment of a manipulation task. Pure motion control is insufficient given the inevitable modelling errors and uncertainties that can result in an increase in contact force, ultimately leading to unstable behaviour during interaction, especially in rigid environments. Feedback and force control are necessary to provide robust and flexible behaviour of the robotic system in unstructured environments and to have a safe and reliable operation in the presence of human operators.

In this section, we describe some of the most common force control methods in the literature, that have also been used as base controllers for the integrated vision-force methods we will discuss in Chapter 4.

2.4.1 Hybrid position/force control

Typically, a robotic task is separated into two main phases: a large motion, in which only position control is used, and fine quasi-static motion, in which force control is the primary concern. However, certain tasks may necessitate precise hybrid control of both velocity and force along a dynamic trajectory.

Following Mason's task description [15], Raibert and Craig [14] proposed a technique capable of controlling either the velocity or the force exchanged by each direction axis of a manipulator in compliant motion with the environment. The essential idea behind their control technique is an architectural concept conceived to disregard position errors along force-controlled axes and force errors along position-controlled axes. On the basis of the relative error, a typical PD/PID force or position controller is designed individually for each axis. This error is calculated using kinematic transformations then, *selection matrices* ($\boldsymbol{S}$ in Figure 2.4), a key element of this controller to keep force- and position-controlled directions separated, avoiding the arise of inconsistencies at the actuation level. In particular, given the task geometry and the constraints imposed by the environment, a subset of the controlled axis will be position/velocity controlled while the remaining ones will be force-controlled (those along the constraints imposed by the environment).

With reference to Figure 2.4, the motion of the workpiece which is rigidly handled by the manipulator's gripper, is constrained by the flat surface on which it slides; linear velocity along the end-effector's z-axis as well as angular velocities about the $x-$ and y-axis are constrained by the task geometry. Moreover, forces

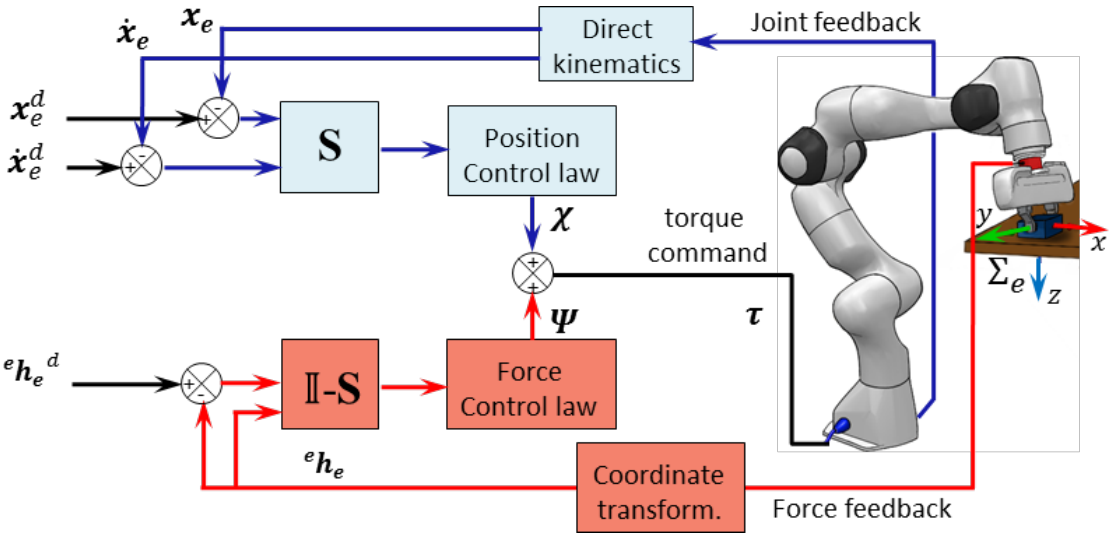


Figure 2.4 – Hybrid force/motion control scheme while executing a cube in plane sliding task. $\boldsymbol{S} = \boldsymbol{S_v}$ is the matrix that selects the motion controlled directions, while $\boldsymbol{S_f} = \mathbb{I} - \boldsymbol{S}$ selects the force controlled ones. The end-effector's current and reference pose are indicated with symbols $\boldsymbol{x}_e$ and $\boldsymbol{x}_e^d$ respectively while $\dot{\boldsymbol{x}}_e$ and $\dot{\boldsymbol{x}}_e^d$ represent its current and reference velocities. The measured and desired end-effector's external wrench are instead indicated with symbols ${}^e\boldsymbol{h}_e$ and ${}^e\boldsymbol{h}_e^d$. The position (2.51) and force (2.54) control laws produce the joint torque contributions that are summed up in $\boldsymbol{\tau}$ to servo the robot along the free motion and constrained Cartesian directions respectively.

along $x-$ and y-axis and moments about the z-axis cannot be arbitrarily imposed; all those are called *natural constraints*. Conversely, the manipulator can be commanded to move along $x-$ and y-axis and about z-axis while arbitrary moments about $x-$ and y-axis and forces along the z-axis can be required to the controller. The latter are named *Natural constraints* and are used to determine the selection matrix for the velocity subspace $\boldsymbol{S_v}$. The selection matrix of the force subspace is quickly obtained as $\boldsymbol{S_f} = \mathbb{I} - \boldsymbol{S_v}$. For this particular example, selection matrices are:

$$\boldsymbol{S_v} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{S_f} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.48}$$

The joint torque controller has the following expression

$$\boldsymbol{\tau} = \chi(\boldsymbol{S_v}, \boldsymbol{\Delta x}, \boldsymbol{\Delta \dot{x}}) + \Psi(\boldsymbol{S_f}, \boldsymbol{\Delta f}) \tag{2.49}$$

$$\chi(\boldsymbol{S_v}, \boldsymbol{\Delta x}, \boldsymbol{\Delta \dot{x}}) = \boldsymbol{K}_{p_P} {\boldsymbol{J}_A}^{-1} \boldsymbol{\Delta x} + \boldsymbol{K}_{p_D} {\boldsymbol{J}_A}^{-1} \boldsymbol{\Delta \dot{x}} + \boldsymbol{K}_{p_I} \int {\boldsymbol{J}_A}^{-1} \boldsymbol{\Delta x} \tag{2.50}$$

$$\boldsymbol{\Delta x} = \boldsymbol{S_v}({}^f\boldsymbol{x}_e^d - \mathcal{K}(\boldsymbol{q})) \tag{2.51}$$

$$\boldsymbol{\Delta \dot{x}} = \boldsymbol{S_v}({}^f\dot{\boldsymbol{x}}_e^d - \boldsymbol{J}_A(\boldsymbol{q})\dot{\boldsymbol{q}}) \tag{2.52}$$

$$\Psi(\boldsymbol{S_f}, \boldsymbol{\Delta f}) = \boldsymbol{J}_e^\top \boldsymbol{S_f}\, {}^e\boldsymbol{h}_e^d + \boldsymbol{K}_{f_P} \boldsymbol{J}_e^\top \boldsymbol{\Delta f} + \boldsymbol{K}_{f_I} \int \boldsymbol{J}_e^\top \boldsymbol{\Delta f} \tag{2.53}$$

$$\boldsymbol{\Delta f} = \boldsymbol{S_f}({}^e\boldsymbol{h}_e^d - {}^e\mathbb{F}_{F/T}\, {}^{F/T}\boldsymbol{h}_{F/T}) \tag{2.54}$$

where ${\boldsymbol{J}_A}^{-1}\boldsymbol{\Delta x}$ is a differential approximation valid only for small $\boldsymbol{\Delta x}$. Otherwise it can be seen as the inverse kinematic problem which provides as solution the joint space error $\boldsymbol{q}_e$; $\chi(\boldsymbol{S_v}, \boldsymbol{\Delta x}, \boldsymbol{\Delta \dot{x}})$ implements a PID controller on the positional error $\boldsymbol{\Delta x}$ while $\Psi(\boldsymbol{S_f}, \boldsymbol{\Delta f})$ implements a force reference *feedforward* plus a PI on the wrench error $\boldsymbol{\Delta f}$. Both force and position contributions to the joint torques are "filtered out" by the relative selection matrices.

Several improvements to the presented hybrid control of manipulators have been proposed since its formulation in 1981 for: compensating the interaction effects due to dynamic coupling like in [11] or in [25] which also preserves the dynamic orthogonality between force and velocity; for accounting for the followed geometric curvature of the object surface [26, 27]; a robust version to dynamic parameters is given in [28] while a generalization for controlling robots in contact with dynamic environments is given in [29]. Kinematic stability issues due to the inverse of the manipulator Jacobian and selection matrices are treated in [30] and [31] respectively.

The presented hybrid position/force controller generates the torque commands for the joints; nevertheless, following the same reasoning, a controller that generates the joint velocities instead of the torques can be synthesized. What matters is the preservation of the orthogonal subspaces through the use of the selection matrices.

2.4.2 Impedance control

The ratio of force output to motion input is known as mechanical impedance; regulating the impedance of a mechanism means adjusting the force of resistance to external motion imposed by the environment. Impedance control seeks to implement a dynamic relationship between manipulator variables such as end-effector position and wrench, rather than simply controlling these variables separately. This concept was originally applied to manipulators by Hogan [12] in 1985. The

strategy that has been chosen to deal with the inertial behaviour of the manipulator is to "hide" the underlying non-linear inertial dynamics of the manipulator and impose a simpler one; that of a rigid body.

A control technique for such mechanical systems, which aims at linearizing and decoupling the manipulator dynamics via feedback, is the well-known *inverse dynamics control*. Nonlinearities such as Coriolis and centrifugal forces as well as gravitational forces, can be cancelled by adding those terms in the control input, while decoupling is achieved through the inertia matrix. The following development follows more closely the development in [13].

From the manipulator's equation of motion (2.27), we want to synthesize an inverse dynamics controller in operational space that minimizes the error $\tilde{\boldsymbol{x}}_e = \boldsymbol{x}_e^d - \boldsymbol{x}_e$. Choosing a joint space control input

$$\boldsymbol{u} = \boldsymbol{B}(\boldsymbol{q})\boldsymbol{\alpha} + \boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{2.55}$$

where $\boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{F}_v\dot{\boldsymbol{q}} + \boldsymbol{F}_s\boldsymbol{sgn}(\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q})$ and $\boldsymbol{\alpha}$ is the new resolved joint space acceleration controller to be properly designed. The manipulator (2.27) under the action of such controller, is described by

$$\ddot{\boldsymbol{q}} = \boldsymbol{\alpha} - \boldsymbol{B}^{-1}\boldsymbol{J}_e^{\top}(\boldsymbol{q})\boldsymbol{h}_e \tag{2.56}$$

which, exploiting the second-order differential kinematic equation (2.15) to link joint and end-effector accelerations, suggest the expression of $\boldsymbol{\alpha}$

$$\boldsymbol{\alpha} = \boldsymbol{J}_A(\boldsymbol{q})^{-1}\boldsymbol{M}_d^{-1}(\boldsymbol{M}_d\ddot{\boldsymbol{x}}_e^d + \boldsymbol{D}\dot{\tilde{\boldsymbol{x}}}_e + \boldsymbol{K}\tilde{\boldsymbol{x}}_e - \boldsymbol{M}_d\dot{\boldsymbol{J}}_A(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}) \tag{2.57}$$

which includes a PD with acceleration *feedforward* in operational space, being $\boldsymbol{M}_d, \boldsymbol{D}$ and $\boldsymbol{K}$ suitable positive definite matrices. Replacing equation (2.57) into (2.56) and accounting for the second-order differential equation, one obtains the *closed-loop* system equation

$$\boldsymbol{M}_d\ddot{\tilde{\boldsymbol{x}}}_e + \boldsymbol{D}\dot{\tilde{\boldsymbol{x}}}_e + \boldsymbol{K}\tilde{\boldsymbol{x}}_e = \boldsymbol{M}_d\boldsymbol{B}_A^{-1}(\boldsymbol{q})\boldsymbol{h}_A \tag{2.58}$$

where $\boldsymbol{B}_A(\boldsymbol{q}) = \boldsymbol{J}_A(\boldsymbol{q})^{-\top}\boldsymbol{B}(\boldsymbol{q})\boldsymbol{J}_A(\boldsymbol{q})^{-1}$ is the inertia matrix of the manipulator in operational space (see Section 2.3.3). The *closed-loop* system equation defines a generalized *mechanical impedance* between the displacement vector $\tilde{\boldsymbol{x}}_e$ and the vector of resultant forces $\boldsymbol{M}_d\boldsymbol{B}_A^{-1}(\boldsymbol{q})\boldsymbol{h}_A$ in the operational space. This impedance is equivalent to a mechanical system characterized by the matrices of *inertia* ($\boldsymbol{M}_d$), *damping* ($\boldsymbol{D}$) and *stiffness* ($\boldsymbol{K}$).

The presence of the configuration dependent inertia matrix $\boldsymbol{B}_A(\boldsymbol{q})^{-1}$ keeps the system dynamically coupled during the interaction. Using a Force/Torque

sensor, it is possible to measure the *contact wrench* ($\boldsymbol{h}_e$) and use it to preserve the linearity and decoupled properties of the system. Adding it then in the input control law

$$\boldsymbol{u} = \boldsymbol{B}(\boldsymbol{q})\boldsymbol{\alpha} + \boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{J}_e^{\top}\, \boldsymbol{h}_e, \tag{2.59}$$

fully compensates for the external forces, making the manipulator infinitely stiff with respect to those forces; then with

$$\boldsymbol{\alpha} = \boldsymbol{J}_A(\boldsymbol{q})^{-1}\boldsymbol{M}_d^{-1}(\boldsymbol{M}_d\ddot{\boldsymbol{x}}_e^d + \boldsymbol{D}\dot{\tilde{\boldsymbol{x}}}_e + \boldsymbol{K}\tilde{\boldsymbol{x}}_e - \boldsymbol{M}_d\dot{\boldsymbol{J}}_A(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{h}_A) \tag{2.60}$$

in which $-\boldsymbol{J}_A(\boldsymbol{q})^{-1}\boldsymbol{M}_d^{-1}\boldsymbol{h}_A$ has been added to endow the system with a linear impedance behaviour with respect to the vector $\boldsymbol{h}_A$, as the closed loop equation shows

$$\boldsymbol{M}_d\ddot{\tilde{\boldsymbol{x}}}_e + \boldsymbol{D}\dot{\tilde{\boldsymbol{x}}}_e + \boldsymbol{K}\tilde{\boldsymbol{x}}_e = \boldsymbol{h}_A \tag{2.61}$$

The control scheme of the impedance control in interaction with the environment is depicted in Figure 2.5.

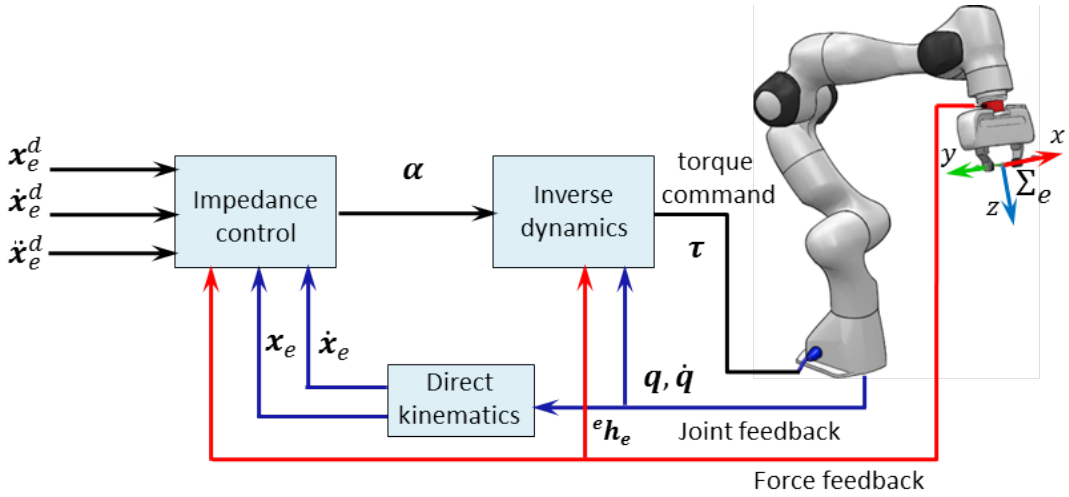


Figure 2.5 – Impedance control scheme. The simplest version of the controller (eq. (2.57)) does not require any Force/Torque measurement. This scheme relates to the controller eq. (2.60) and inverse dynamics eq. (2.59).

At equilibrium, *i.e.*, when the manipulator is at rest ($\boldsymbol{q} = \dot{\boldsymbol{q}} = 0$), the position of the end-effector is given by

$$\boldsymbol{x}_e = \boldsymbol{x}_e^d - \boldsymbol{K}^{-1}\boldsymbol{h}_A \tag{2.62}$$

When the manipulator is not in contact with the environment ($\boldsymbol{h}_A = 0$) the actual position of the end-effector reaches the desired one ($\boldsymbol{x}_e = \boldsymbol{x}_e^d$). When in

contact with the environment, the positioning error is proportional to the inverse of the chosen stiffness $\boldsymbol{K}$ and the equivalent contact wrench $\boldsymbol{h}_A$. It should be noted that, with respect to the contact wrench $\boldsymbol{h}_e$, the impedance is dependent on the orientation of the end-effector due to the presence of the matrix $\boldsymbol{\Gamma}_A({}^f\boldsymbol{\phi}_e)$ (see (2.22)), making the choice of the impedance parameters difficult.

To avoid this problem it is sufficient to redesign the control input $\boldsymbol{\alpha}$ with respect to a different error vector in operational space

$$\tilde{\boldsymbol{x}} = -\begin{bmatrix} {}^d\boldsymbol{p}_{d,e} \\ {}^d\boldsymbol{\phi}_{d,e} \end{bmatrix} \tag{2.63}$$

where ${}^d\boldsymbol{p}_{d,e}$ is the position error between the origin of the current and desired end-effector frames and ${}^d\boldsymbol{\phi}_{d,e}$ is the orientation vector extracted from the rotation matrix ${}^d\boldsymbol{R}_e({}^d\boldsymbol{\phi}_{d,e})$. For further details refer to [13].

2.4.3 Admittance control

The choice of the impedance parameters in the *impedance controller* determines its dynamic behaviour, both in free space and when in contact with the environment. Indeed if the interaction wrench is generated from contact with an environment described by a *mass-spring-damper*, the interacting system manipulator-environment is equivalent to a mechanical system made by the parallel of the two impedances, and the resulting closed-loop dynamics along the constrained directions will differ from those in free space. Since the dynamic behavior of the system is influenced by environmental rigidity, its stability is concerned as well.

Among the objectives to pursue when choosing the impedance parameters, there is the need to ensure a high disturbance rejection factor given the model uncertainties and the approximations induced in the computation of the inverse dynamics. This factor is proportional to the gain $\boldsymbol{K}$ which also impacts the trajectory tracking performances. Indeed, the impedance controller in free space is nothing but an inverse dynamics position controller. It follows that tracking performance will be more degraded the lower this parameter is chosen. It is therefore evident that there is a trade-off between the tracking performance obtainable and the compliance assigned to the manipulator. The more rigid the manipulator is commanded to be, the better it will follow the reference trajectory and vice versa.

Separating the motion control and impedance control problems, as done in the control scheme reported in Figure 2.6, could be a solution to the aforementioned issues.

Let us suppose to have three reference frames, the current Σ_c, the desired Σ_d and the compliant frame Σ_{c^*} and that the compliant frame is coincident with the

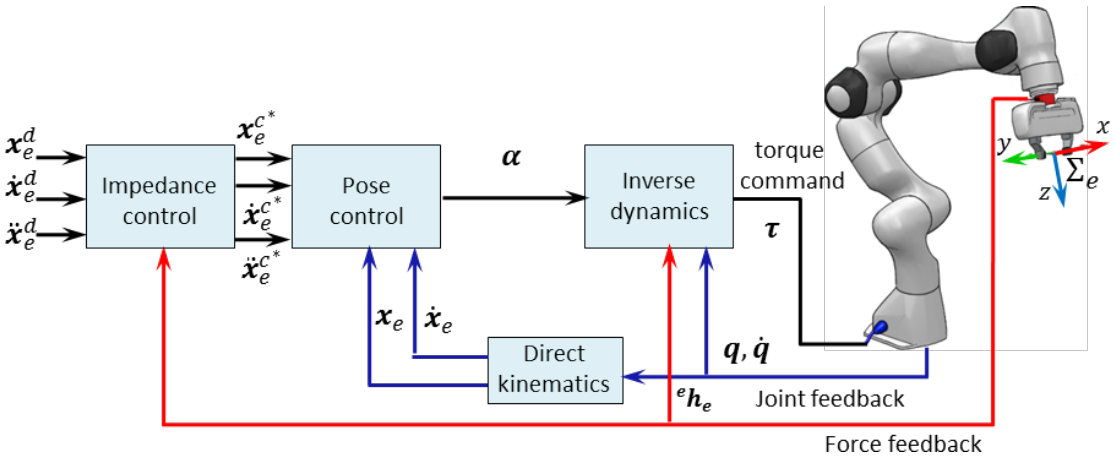


Figure 2.6 – Admittance control scheme. It is composed by an internal "stiff" impedance controller (pose control eq. (2.57)) and inverse dynamics eq. (2.55)) and an external "impedance law" (eq. (2.66)).

desired frame. The pose of those frames with respect to a fixed frame are given by

$$ {}^{f}\boldsymbol{T}_c = \left[\begin{array}{c|c} {}^{f}\boldsymbol{R}_c & {}^{f}\boldsymbol{p}_c \\ \hline 0\ 0\ 0 & 1 \end{array}\right]; {}^{f}\boldsymbol{T}_d = \left[\begin{array}{c|c} {}^{f}\boldsymbol{R}_d & {}^{f}\boldsymbol{p}_d \\ \hline 0\ 0\ 0 & 1 \end{array}\right]; {}^{f}\boldsymbol{T}_{c^*} = \left[\begin{array}{c|c} {}^{f}\boldsymbol{R}_{c^*} & {}^{f}\boldsymbol{p}_{c^*} \\ \hline 0\ 0\ 0 & 1 \end{array}\right] \tag{2.64} $$

Let us also suppose that there are two pose errors, $\boldsymbol{e}_c$ between the current and the compliant frames and $\bar{\boldsymbol{e}}_c$ between the desired and the compliant frame

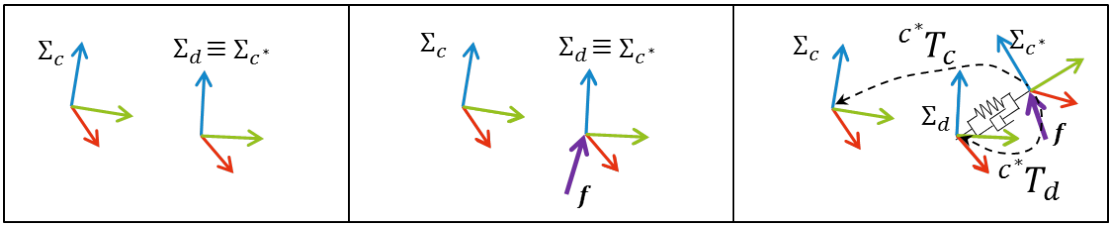
$$ \begin{aligned} {}^{c^*}\boldsymbol{T}_c(\boldsymbol{e}_c) = {}^{f}\boldsymbol{T}_{c^*}{}^{-1}{}^{f}\boldsymbol{T}_c \rightarrow \boldsymbol{e}_c = \begin{bmatrix} {}^{c^*}\boldsymbol{p}_{c^*,c} \\ {}^{c^*}\boldsymbol{\phi}_{c^*,c} \end{bmatrix} \\ {}^{d}\boldsymbol{T}_{c^*}(\bar{\boldsymbol{e}}_c) = {}^{f}\boldsymbol{T}_{d}{}^{-1}{}^{f}\boldsymbol{T}_{c^*} \rightarrow \bar{\boldsymbol{e}}_c = \begin{bmatrix} {}^{d}\boldsymbol{p}_{d,c^*} \\ {}^{d}\boldsymbol{\phi}_{d,c^*} \end{bmatrix} \end{aligned} \tag{2.65} $$

The latter error is used as displacement vector in an impedance equation subjected to the action of a wrench $\boldsymbol{h}_e$ which makes the compliant frame move according to the impedance law (2.66) (see Figure 2.7).

The *admittance* controller is based on this concept of *compliant frame*, which is a Cartesian reference frame that describes the ideal end-effector's behavior as a result of impedance control. The linear and angular velocity and acceleration of the compliant frame are computed integrating impedance equations of the form

$$ \boldsymbol{M}_t\ddot{\bar{\boldsymbol{e}}}_c + \boldsymbol{D}_t\dot{\bar{\boldsymbol{e}}}_c + \boldsymbol{K}_t\bar{\boldsymbol{e}}_c = \boldsymbol{h}_e \tag{2.66} $$

where $\boldsymbol{M}_t$, $\boldsymbol{D}_t$ and $\boldsymbol{K}_t$ are the parameters of a mechanical impedance between the desired frame and the compliant frame.



(a) (b) (c)

Figure 2.7 – Compliant frame concept: (a) Desired and Compliant frames are coincident, the Current frame tracks the compliant frame. (b) The system undergoes the action of an external force. (c) The compliant frame is linked to the desired one with a *mass-spring-damper* relationship.

The position, velocity and acceleration of the compliant frame are then used as reference input to a "stiff" impedance controller, see Figure 2.6. The control parameters for both the motion and the compliance controller can be designed separately; ensuring high rejection factor and trajectory tracking performance for the motion controller on one hand, and imposing the desired interaction behaviour opportunely tuning the impedance parameters on the other. For stability purposes of the whole controlled system, the bandwidth of the motion controller must be greater than the one of the impedance.

2.5 Conclusion

In this introductory chapter, we examined the fundamental concepts of robot kinematics, its dynamic model, dynamic parameter identification, and force control of manipulators. These notions form the foundation of knowledge required to comprehend the content presented here. Furthermore, this chapter has allowed us to illustrate the fundamental concepts, terminology and symbols that will be used in the subsequent chapters.

On the other hand, as will be seen in Chapter 4, the motion/force control schemes discussed above will constitute the basic schemes upon which the main vision/force control schemes have been built, essentially replacing the motion loop with a vision-based control loop.

The focus of the next chapter is on vision control, the second essential brick on which this work is built on.

Chapter 3

Vision Control

Contents

3.1 Introduction

Motivated by the desire of reducing or avoiding accurate environmental modeling, robotic systems are more and more often equipped with external exteroceptive sensors, like cameras, that allow them to acquire geometric information on the environment in which they operate, without direct contact with it. This allows such systems to perform the required tasks without having an accurate preliminary knowledge of the environment.

The visual perception is a very versatile sensing modality and the extracted information can be used for many purposes ranging from motion planning, localization and mapping, scene segmentation, control, just to name a few. This kind of sensors becomes essential when a task has to be performed in a partially known/unknown environment or when it is not completely rigid. Cameras make then possible to account for environmental modeling errors as well as for inaccuracies in the identified geometric model of the manipulator.

A fundamental aspect that characterizes classical visual control, with respect to motion and force control, is the fact that the controlled quantities are not directly measured by the sensor, but are obtained from these downstream of rather complex image processing and computer vision algorithms. Such algorithms allow the extraction of digital information, known as visual features, relating to the objects present in the images of a scene, that can be used to estimate the position and orientation (*pose*) of the camera with respect to those objects.

Different other paradigms of visual control exist that do not make use of classical geometric features. For instance, [32] considers the luminance of the pixels of the entire image as visual feature; this approach is known as photometric visual servoing and belongs to the so called *direct visual servoing* methods. [33] proposed a generic framework to consider histograms as visual features while [34] presented a deep neural network-based method to perform 6 DOF visual servoing in which a Convolutional Neural Network was trained to estimate the relative pose between two images of the same scene.

The algorithmic and technological progress made over the past decades has led to tightly intertwine the aspects of perception with those of action, by directly integrating the measurements provided by a vision system into closed-loop control laws that deal with the extracted visual information. When visual measurements are used in feedback in a control algorithm, we speak of visual control or *Visual-Servoing*, and it is the main focus of this chapter. This term seems to be firstly introduced by Hill and Park [35] in 1979 to distinguish their approach from earlier “blocks world” experiments where the system alternated between picture

taking and moving, referred as *look-then-move*. That approach uses open-loop visual measurements and therefore makes the system not very robust against uncertainties due, for example, to the fact that the object may have moved while the end-effector reaches the desired pose. Prior to the adoption of its current term, Visual-servoing was generally referred to by the less specific term *visual feedback*. The difference between visual-servoing and classical look-then-move systems can be also seen from the data abstraction flow perspective from perception and action [36] as shown in Figure 3.1.

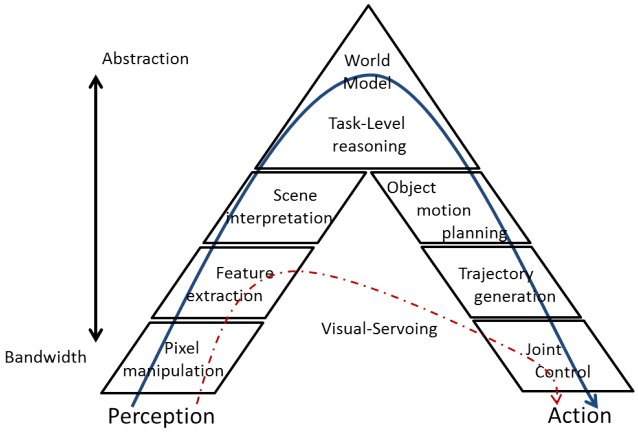


Figure 3.1 – General hierarchical control structure of a classical *look-then-move* approach (blue line) in contrast with a visual servoing system (dashed red line). Visual-servoing can be considered as a low-level "shortcut" through the hierarchy, characterized by high bandwidth but moderate image processing. Higher levels correspond to more abstract data representation and lower bandwidth.

3.2 Camera model

Cameras are at the heart of any visual system and it is worth delving into their mathematical characterization. A camera model describes the mathematical relationship between the coordinates of a point $\boldsymbol{X} = (X, Y, Z)^\top$ in three-dimensional space and its projection on the image plane $\boldsymbol{x} = (x, y)$, see Figure 3.2. By far, the most common camera model is the *pinhole* model, which makes the fundamental assumption that rays of light enter the camera through an infinitely small aperture.

A scene's 3D point $\boldsymbol{X}$ is firstly transformed in camera frame Σ_c coordinates $({}^c\boldsymbol{X} = [{}^cX, {}^cY, {}^cZ]^\top)$ and then is projected, using a perspective transformation,

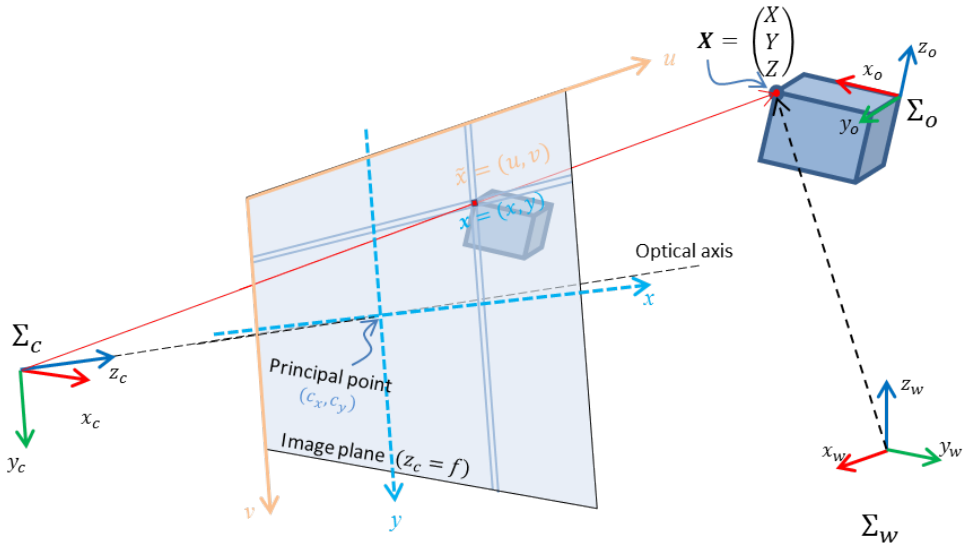


Figure 3.2 – Frontal pinhole imaging model: the image of a 3-D point ${}^c\boldsymbol{X}$ is the point $\boldsymbol{x}$ at the intersection of the ray going through the optical center O_c and the image plane at a distance f in front of the optical center.

into the image plane which forms the corresponding point $\boldsymbol{x} = (x, y)$, we have:

$$\begin{cases} x &= {}^cX/{}^cZ = (u - c_x)/f\alpha \\ y &= {}^cY/{}^cZ = (v - c_y)/f \end{cases} \tag{3.1}$$

being f the ratio between the *focal length*, which is the distance between the origin of the camera frame and the image plane, and the pixel vertical dimension, α the ratio of the pixel horizontal and vertical dimensions, c_x and c_y the coordinates of the *principal point* in pixels. $\bar{\boldsymbol{x}} = (u, v)$ gives the coordinates of the image point expressed in pixel units, and $\boldsymbol{a} = (c_x, c_y, f, \alpha)$ is the set of camera intrinsic parameters. Further details on imaging geometry and perspective projection can be found in many computer vision text books [37, 38].

3.3 Visual Servoing

Visual servoing techniques consist of using the information provided by one or several cameras to control the movements of a robotic system. This allows to execute a wide range of tasks spanning from locating the system with respect to its environment, or tracking mobile objects, by controlling one or as many as all of the camera's $n \leq 6$ degrees of freedom. Regardless the sensor's configuration, which can be both a mounted camera on the robot's end-effector (*eye-in-hand*) or one (several) scene camera(s) (*eye-to-hand*), the objective is twofold; on the

one hand, one must select a set of k possible visual data, in order to control the desired $m \leq n$ degrees of freedom, while on the other hand, one must develop a control law capable of carrying these data $\boldsymbol{s}(t)$ to reach the desired value $\boldsymbol{s}^d$ which defines when the task has been properly executed. It is also possible to track a desired trajectory $\boldsymbol{s}^d(t)$. The aim of all vision-based control schemes is to minimize an error vector $\boldsymbol{e_s} = \boldsymbol{s}^d(t) - \boldsymbol{s}(t)$, that is, making this value reach zero and maintaining it there [39].

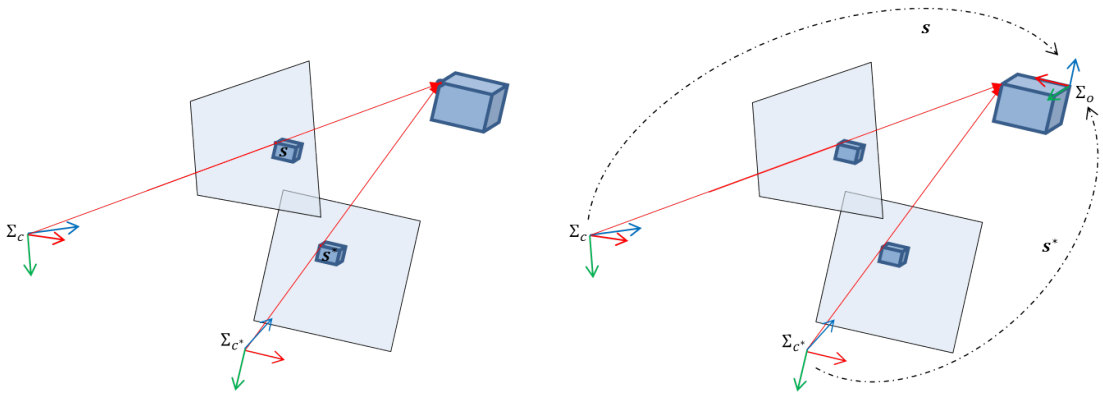


Figure 3.3 – 2-D and 3-D visual servoing: 2-D visual servoing is used to bring the camera's frame of reference from Σ_{c^*} to Σ_c based on measurements $\boldsymbol{s}$ extracted directly from the image (left). With 3-D visual servoing, the measurements $\boldsymbol{s}$ represent 3-D data estimated after a 3D localization phase (right)

To study the system's behavior, a model describing the relationship between the chosen visual data $\boldsymbol{s}(t) \in \mathbb{R}^k$ and the control variables is needed. The vector $\boldsymbol{s}^d$ contains the desired values of the features. Let us suppose the vector of visual data to be defined by a differentiable application $\boldsymbol{s} : SE(3) \rightarrow \mathbb{R}^k$:

$$\boldsymbol{s} = \boldsymbol{s}(\boldsymbol{m}(t), \boldsymbol{a}) \tag{3.2}$$

the vector $\boldsymbol{m}(t)$ represents a set of image measurements (e.g., the image coordinates of interest points or the image coordinates of the centroid of an object). These image data are utilized to generate a $\boldsymbol{s}(\boldsymbol{m}(t), \boldsymbol{a})$ vector of k visual features, where $\boldsymbol{a}$ is a set of parameters that reflect further information (e.g., coarse camera intrinsic parameters or 3-D models of objects). The visual datum can therefore be modified by the motion of either the camera or the object it perceives. Once $\boldsymbol{s}$ is chosen, the control scheme design can be quite simple designing, for example, a velocity controller. The relationship that links the relative twist ${}^{\boldsymbol{c}}\boldsymbol{v} \in \mathbb{R}^6$ between the camera and the object with the rate of change of the visual data can be obtained differentiating equation (3.2), leading to:

$$\dot{\boldsymbol{s}} = \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{p}} \dot{\boldsymbol{p}} = \boldsymbol{L_s}\, {}^{\boldsymbol{c}}\boldsymbol{v} = \boldsymbol{L_s}({}^{\boldsymbol{c}}\boldsymbol{v_c} - {}^{\boldsymbol{c}}\boldsymbol{v_o}) = \boldsymbol{L_s}\, {}^{\boldsymbol{c}}\boldsymbol{v_c} + \dot{\boldsymbol{s}}_o \tag{3.3}$$

being $\boldsymbol{L_s} \in \mathbb{R}^{k\times 6}$ the well-known *interaction matrix* [40] related to $\boldsymbol{s}$, ${}^{c}\boldsymbol{v_c} = [{}^{c}\mathbf{v}_c^\top \quad {}^{c}\boldsymbol{\omega}_c^\top]^\top$ and ${}^{c}\boldsymbol{v_o} = [{}^{c}\mathbf{v}_o^\top \quad {}^{c}\boldsymbol{\omega}_o^\top]^\top$ the camera and object absolute twists in camera frame respectively, while $\dot{\boldsymbol{s}}_o$ represents the contribution to the features' velocity due to the target motion.

From the previous equation (3.3) and considering the time derivative of the feature error for static target (${}^{c}\boldsymbol{v_o} = 0$) and constant desired features $\dot{\boldsymbol{s}}^d = 0$, $\dot{\boldsymbol{e}}_{\boldsymbol{s}}(t) = \dot{\boldsymbol{s}}^d - \dot{\boldsymbol{s}}(t) = -\dot{\boldsymbol{s}}(t)$, we immediately obtain

$$\dot{\boldsymbol{e}}_{\boldsymbol{s}}(t) = -\boldsymbol{L_s}\ {}^{c}\boldsymbol{v_c} \tag{3.4}$$

Considering ${}^{c}\boldsymbol{v_c}$ as the input to the manipulator controller (${}^{c}\boldsymbol{v}_{\boldsymbol{c}}^{cmd}$), and to ensure an exponential decoupled decrease of the error, we ask as a control objective to have $\dot{\boldsymbol{e}}_{\boldsymbol{s}} = -\lambda \boldsymbol{e_s}$, that together with equation (3.4) we obtain

$${}^{c}\boldsymbol{v}_{\boldsymbol{c}}^{cmd} = \lambda \boldsymbol{L}_{\boldsymbol{s}}^{\dagger} \boldsymbol{e_s} \tag{3.5}$$

where $\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}$ is chosen as the Moore-Penrose pseudo-inverse of $\boldsymbol{L_s}$, that is, $\boldsymbol{L}_{\boldsymbol{s}}^{\dagger} = (\boldsymbol{L}_{\boldsymbol{s}}^\top \boldsymbol{L_s})^{-1}\boldsymbol{L}_{\boldsymbol{s}}^\top$ when $\boldsymbol{L_s}$ is of full-rank ($\mathrm{rank}(\boldsymbol{L_s}) = 6$). This choice allows $\|\dot{\boldsymbol{e}}_{\boldsymbol{s}} - \lambda \boldsymbol{L_s}\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}\boldsymbol{e_s}\|$ and $\|{}^{c}\boldsymbol{v}_{\boldsymbol{c}}^{cmd}\|$ to be minimal. When $k = 6$, if $det(\boldsymbol{L_s}) \neq 0$ it is possible to invert $\boldsymbol{L_s}$, giving the control ${}^{c}\boldsymbol{v}_{\boldsymbol{c}}^{cmd} = \lambda \boldsymbol{L}_{\boldsymbol{s}}^{-1}\boldsymbol{e_s}$. In real visual servo systems, it is impossible to know perfectly in practice either $\boldsymbol{L_s}$ or $\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}$. So an approximation or an estimation of one of these two matrices must be realized. In the sequel, we denote both the pseudoinverse of the approximation of the interaction matrix and the approximation of the pseudoinverse of the interaction matrix by the symbol $\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}$. Using this notation, the control law is in fact:

$${}^{c}\boldsymbol{v}_{\boldsymbol{c}}^{cmd} = \lambda \widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}} \boldsymbol{e_s} \tag{3.6}$$

This is the basic design implemented by most visual servo controllers.

The main difference between visual servoing schemes is how features $\boldsymbol{s}$ are designed. We will take a glance at two very different approaches in the next two subsections of this chapter. First, we describe image-based visual servo control (IBVS), which consists of a set of features that are readily available as image data. Then, we describe pose-based visual servo (PBVS) control, in which $\boldsymbol{s}$ is a set of three-dimensional parameters that must be estimated from image measurements.

3.3.1 IBVS: Image-Based Visual Servoing

In Image-Based Visual Servoing (IBVS) the visual features consist of 2D primitives directly extracted from the image plane. Traditional control schemes use a set of points in image plane coordinates, but many other primitives can be adopted [40],

like straight lines or image moments [41]. The image measurements are usually obtained in pixel coordinates and the intrinsic parameters are used to go from pixels to the features $\boldsymbol{s}$.

For the case of image points features, it is possible to obtain the relative interaction matrix by time derivation of the perspective model equation (3.1) (see also Figure 3.2) and relating then the velocity of the 3D point to the camera spatial velocity using the well-known equation ${}^c\dot{\boldsymbol{X}} = -{}^c\mathbf{v}_c - {}^c\boldsymbol{\omega}_c \times {}^c\boldsymbol{X}$, finally yielding $\dot{\boldsymbol{x}} = \boldsymbol{L}_{\boldsymbol{x}}\, {}^c\boldsymbol{v}_{\boldsymbol{c}}$, with

$$\boldsymbol{L}_{\boldsymbol{x}}(\boldsymbol{x},\, {}^cZ) = \begin{bmatrix} -1/{}^cZ & 0 & x/{}^cZ & xy & -(1+x^2) & y \\ 0 & -1/{}^cZ & y/{}^cZ & 1+y^2 & -xy & -x \end{bmatrix} \tag{3.7}$$

in which the value ${}^cZ$ is the depth of the point relative to the camera frame; therefore, any control scheme that uses this form of the interaction matrix must estimate or approximate this value. Similarly, the camera intrinsic parameters are involved in the computation of x and y. Thus, $\boldsymbol{L}_{\boldsymbol{x}}$ cannot be directly used and has to be estimated or approximated $\widehat{\boldsymbol{L}_{\boldsymbol{x}}}$.

To control the 6 degrees of freedom we need at least six independent features to fully constrain the system. We can use then three image points ($k = 6$ two coordinates per image point), but given the singularity problems related to this choice, usually more than three points are used ($k \geq 6$). The feature vector is built as $\boldsymbol{s} = [\boldsymbol{x}_1^\top\ \boldsymbol{x}_2^\top\ \boldsymbol{x}_3^\top\ \dots]^\top$, while the interaction matrix is obtained by stacking the individual interaction matrices of each point of the type (3.7)

$$\boldsymbol{L}_{\boldsymbol{s}}(\boldsymbol{s}, \boldsymbol{Z}) = \begin{bmatrix} \boldsymbol{L}_{\boldsymbol{x}}(\boldsymbol{x}_1,\, {}^cZ_1) \\ \boldsymbol{L}_{\boldsymbol{x}}(\boldsymbol{x}_2,\, {}^cZ_2) \\ \boldsymbol{L}_{\boldsymbol{x}}(\boldsymbol{x}_3,\, {}^cZ_3) \\ \vdots \end{bmatrix} \tag{3.8}$$

The same idea holds for any other primitive we want to use, like straight lines, which are represented in polar coordinates as shown in Figure 3.4

Results for more complex primitives (circles, spheres, and cylinders) are given in [42].

Once enough features have been chosen to constrain all the task's degrees of freedom, we can apply control equation (3.6).

3.3.2 PBVS: Pose-Based Visual Servoing

Pose-Based Visual Servoing (PBVS), as previously stated, uses the pose parameters of the frame attached to the camera with respect to another frame, usually

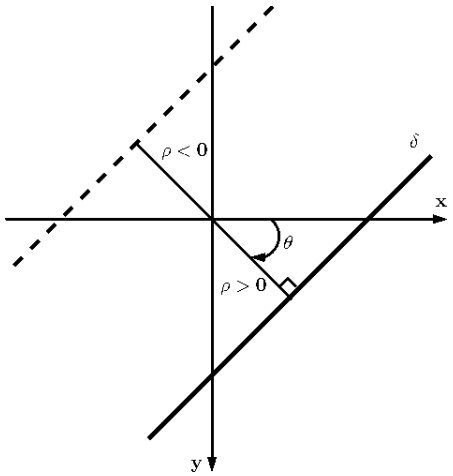


Figure 3.4 – 2D line representation in polar coordinates (ρ, θ).

the object frame, to define the vector of visual features $\boldsymbol{s}$. The camera intrinsic parameters and the 3-D model of the observed object must be known in order to compute its pose from a set of measurements in a single image. This is a classical problem in computer vision known as *3D localization problem*. The parameterization used to define $\boldsymbol{s}$ is thus determined by the representation adopted for the camera pose. We consider three different coordinate frames, the current camera frame Σ_c, the desired camera frame Σ_d, and a reference frame Σ_o associated to the object. If ${}^c\boldsymbol{T}_o$ is the homogeneous transformation matrix that expresses the pose of the object in camera frame coordinates (Σ_c), ${}^d\boldsymbol{T}_o$ the object's pose in desired camera frame (Σ_d), one can express the pose error between the current and the desired pose of the camera in the desired frame as

$$ {}^d\boldsymbol{T}_o \, {}^c\boldsymbol{T}_o^{-1} = {}^d\boldsymbol{T}_c = \left[\begin{array}{c|c} {}^d\boldsymbol{R}_c({}^d\boldsymbol{\phi}_c) & {}^d\boldsymbol{p}_c \\ \hline 0\,0\,0 & 1 \end{array}\right] \tag{3.9} $$

where for the orientation representation ${}^d\boldsymbol{\phi}_c = \theta\boldsymbol{u}$, the axis/angle is adopted. The feature vector is then $\boldsymbol{s} = ({}^d\boldsymbol{p}_c, \theta\boldsymbol{u})$. In this case $\boldsymbol{s}^d = 0$ and $\boldsymbol{e}_{\boldsymbol{s}} = \boldsymbol{s}$, while the interaction matrix is

$$ \boldsymbol{L}_{\boldsymbol{X}} = \begin{bmatrix} {}^d\boldsymbol{R}_c({}^d\boldsymbol{\phi}_c) & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{L}_{\theta\boldsymbol{u}} \end{bmatrix} \tag{3.10} $$

with

$$ \boldsymbol{L}_{\theta\boldsymbol{u}} = \mathbb{I}_3 - \frac{\theta}{2}[\boldsymbol{u}]_\times + \left(1 - \frac{sinc(\theta)}{sinc^2(\frac{\theta}{2})}\right)[\boldsymbol{u}]_\times^2 \tag{3.11} $$

where $sinc(\theta) = \sin(\theta)/\theta$.

This feature representation fully decouples the translational and rotational motions allowing to achieve straight line motions of the end-effector in Cartesian

space using the kinematic control law equation (3.6)

$$^{c}\boldsymbol{v}_{\boldsymbol{c}}^{cmd} = \lambda \widehat{\boldsymbol{L}_{\boldsymbol{X}}^{-1}} \boldsymbol{e}_{\boldsymbol{s}} \tag{3.12}$$

3.3.3 Stability analysis

In order to assess the stability of the closed-loop visual servo systems, we can resort to the Lyapunov analysis by considering the candidate function defined by the squared error norm $\boldsymbol{V} = \frac{1}{2}\boldsymbol{e}_{\boldsymbol{s}}^{\top}\boldsymbol{e}_{\boldsymbol{s}}$, whose derivative is given by

$$\dot{\boldsymbol{V}} = \boldsymbol{e}_{\boldsymbol{s}}^{\top}\dot{\boldsymbol{e}}_{\boldsymbol{s}} = -\lambda \boldsymbol{e}_{\boldsymbol{s}}^{\top}\boldsymbol{L}_{\boldsymbol{s}}\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}\boldsymbol{e}_{\boldsymbol{s}} \tag{3.13}$$

to which the error dynamics $\dot{\boldsymbol{e}}_{\boldsymbol{s}} = \dot{\boldsymbol{s}} = \boldsymbol{L}_{\boldsymbol{s}}\,^{c}\boldsymbol{v}_{\boldsymbol{c}}$ and the controller $^{c}\boldsymbol{v}_{\boldsymbol{c}}^{cmd} = \lambda\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}\boldsymbol{e}_{\boldsymbol{s}}$ have been replaced revealing that the global asymptotic stability of the system is obtained when the following sufficient condition is ensured:

$$\boldsymbol{L}_{\boldsymbol{s}}\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}} > 0 \tag{3.14}$$

If the camera degrees of freedom equals the number of features (*i.e.*, $k = 6$), and if the features are chosen so that $\boldsymbol{L}_{\boldsymbol{s}}$ and $\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}$ have full rank 6, then the previous condition is guaranteed if the approximation $\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}$ is sufficiently accurate.

3.3.3.1 Stability analysis in the IBVS case

Most of the IBVS cases consider a redundant number of features ($k > 6$) and therefore the stability condition stated before can never be attained. Since $\boldsymbol{L}_{\boldsymbol{s}}\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}} \in \mathbb{R}^{k\times k}$ can at most have $\text{rank}(\boldsymbol{L}_{\boldsymbol{s}}\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}) = 6$, it presents a non trivial null space. In this case, feature configurations $\boldsymbol{e}_{\boldsymbol{s}} \in \ker(\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}})$ corresponds to local minima, thus only local asymptotic stability can be obtained and further analysis is required.

Let us define a new error $\boldsymbol{e}'_{\boldsymbol{s}} = \widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}\boldsymbol{e}_{\boldsymbol{s}}$ whose time derivative is

$$\dot{\boldsymbol{e}}'_{\boldsymbol{s}} = \widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}\dot{\boldsymbol{e}}_{\boldsymbol{s}} + \frac{d\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}}{dt}\boldsymbol{e}_{\boldsymbol{s}} \tag{3.15}$$

and since, following [43], $\frac{d\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}}{dt}\boldsymbol{e}_{\boldsymbol{s}} = \boldsymbol{\mathcal{O}}(\boldsymbol{e}_{\boldsymbol{s}})^{c}\boldsymbol{v}_{\boldsymbol{c}}$, it results

$$\dot{\boldsymbol{e}}'_{\boldsymbol{s}} = (\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}\boldsymbol{L}_{\boldsymbol{s}} + \boldsymbol{\mathcal{O}}(\boldsymbol{e}_{\boldsymbol{s}}))^{c}\boldsymbol{v}_{\boldsymbol{c}} = -\lambda(\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}\boldsymbol{L}_{\boldsymbol{s}} + \boldsymbol{\mathcal{O}}(\boldsymbol{e}_{\boldsymbol{s}}))\boldsymbol{e}'_{\boldsymbol{s}} \tag{3.16}$$

where $\boldsymbol{\mathcal{O}}(\boldsymbol{e}_{\boldsymbol{s}}) \to 0$ when $\boldsymbol{s} \to \boldsymbol{s}^{d}$, resulting in the condition

$$\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}\boldsymbol{L}_{\boldsymbol{s}} > 0 \tag{3.17}$$

for ensuring local asymptotic stability. Indeed, only the linearized system $\dot{\boldsymbol{e}}'_{\boldsymbol{s}} = -\lambda \widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}} \boldsymbol{L}_{\boldsymbol{s}} \boldsymbol{e}'_{\boldsymbol{s}}$ has to be considered if we are interested in the local asymptotic stability.

Even though local asymptotic stability can be ensured when $k > 6$, we cannot ensure global asymptotic stability. Determining the size of the neighborhood where the stability and the convergence are ensured is still an open issue, even if this neighborhood is surprisingly large in practice [44].

On the other hand, when dealing with a large number of visual features ($k > 6$), their selection and configuration is an important aspect to not underestimate as they must be chosen carefully to avoid potential local minima in the workspace. It is indeed possible to get stuck into a given configuration when

$$(\boldsymbol{s}^d - \boldsymbol{s}) \in \ker(\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}) \tag{3.18}$$

since a zero velocity command will be generated even if $\boldsymbol{s} \neq \boldsymbol{s}^d \quad (\boldsymbol{e}_{\boldsymbol{s}} \neq 0)$.

3.3.3.2 Stability analysis in the PBVS case

The stability properties of PBVS are quite appealing. Since $\boldsymbol{L}_{\theta u}$ given in equation (3.11) is singular only for $\theta = 2k\pi$, stability condition equation (3.14) guarantees the global asymptotic stability under the strong hypothesis that all the pose parameters are perfectly known, leading to $\boldsymbol{L}_{\boldsymbol{X}} \widehat{\boldsymbol{L}_{\boldsymbol{X}}^{-1}} = \mathbb{I}_6$. In practice, however, even small errors in computing the points position in the image can lead to pose errors that can impact significantly the accuracy and the stability of the system [44].

3.3.4 Mounted camera

If we consider a camera mounted on the end-effector of a manipulator observing a (possibly moving) object, the relation between the rate of change of the visual features $\dot{\boldsymbol{s}}$, the joint velocities of the robot $\dot{\boldsymbol{q}}$ and the target motion can easily be obtained as:

$$\dot{\boldsymbol{s}} = \boldsymbol{L}_{\boldsymbol{s}}\, {}^{c}\boldsymbol{v} = \boldsymbol{L}_{\boldsymbol{s}}\, {}^{c}\mathbb{T}_{e}\, {}^{e}\boldsymbol{J}_{e}(\boldsymbol{q})\, \dot{\boldsymbol{q}} - \boldsymbol{L}_{\boldsymbol{s}}\, {}^{c}\boldsymbol{v}_{\boldsymbol{o}} = \boldsymbol{L}_{\boldsymbol{s}}\, {}^{c}\mathbb{T}_{e}\, {}^{e}\boldsymbol{J}_{e}(\boldsymbol{q})\, \dot{\boldsymbol{q}} - \dot{\boldsymbol{s}}_{o} \tag{3.19}$$

being ${}^{e}\boldsymbol{J}_{e}(\boldsymbol{q})$ the robot Jacobian in end-effector coordinates as saw in Section 2.2.1, and ${}^{c}\mathbb{T}_{e}$ the twist-transformation matrix that transforms the end-effector twist from the end-effector frame to the camera frame (see Section 2.2.3). For the sake of readability, from now on the Jacobian dependency on the robot's configuration ($\boldsymbol{q}$) will be omitted.

3.3.5 Feature trajectory tracking

Let now consider the case in which a desired feature trajectory $\dot{\boldsymbol{s}}^d(t)$ is planned. When path planning and trajectory following are combined, the visual servoing robustness with respect to modeling errors is considerably improved. The control laws seen in the previous subsections can be adapted to take into account the time-varying nature of the given trajectory such that the error $\boldsymbol{e_s} = \boldsymbol{s}^d - \boldsymbol{s}$ remain small. In particular, deriving the error we have now that

$$\dot{\boldsymbol{e}}_{\boldsymbol{s}} = \dot{\boldsymbol{s}}^d - \dot{\boldsymbol{s}} = \dot{\boldsymbol{s}}^d - \boldsymbol{L}_{\boldsymbol{s}}^{\boldsymbol{c}} \boldsymbol{v_c} \tag{3.20}$$

from which, by setting our control objective as before $\dot{\boldsymbol{e}}_{\boldsymbol{s}} = -\lambda \boldsymbol{e_s}$, one obtains the following control law

$${}^{c}\boldsymbol{v}_{\boldsymbol{c}}^{cmd} = \lambda \widehat{\boldsymbol{L}_{\boldsymbol{s}}}^{\dagger} \boldsymbol{e_s} + \widehat{\boldsymbol{L}_{\boldsymbol{s}}}^{\dagger} \dot{\boldsymbol{s}}^d \tag{3.21}$$

The control law's new second term anticipates the variation of $\dot{\boldsymbol{s}}^d$, eliminating the tracking error it would cause. When tracking a moving target, a similar form of the control law will be attained, as we will see in the next.

3.3.6 Target tracking

Let consider now the case of regulation to a constant desired value $\boldsymbol{s}^d$ of the features following a moving target. Deriving again the error vector we obtain this time

$$\dot{\boldsymbol{e}}_{\boldsymbol{s}} = \dot{\boldsymbol{s}}^d - \dot{\boldsymbol{s}} = -\boldsymbol{L}_{\boldsymbol{s}}^{\boldsymbol{c}} \boldsymbol{v_c} + \frac{\partial \boldsymbol{e_s}}{\partial t} \tag{3.22}$$

where the term $\partial \boldsymbol{e_s}/\partial t = \dot{\boldsymbol{s}}_o$ expresses the error variation due to the generally unknown motion of the target. Using the same control objective as usual, we obtain

$${}^{c}\boldsymbol{v}_{\boldsymbol{c}}^{cmd} = \lambda \widehat{\boldsymbol{L}_{\boldsymbol{s}}}^{\dagger} \boldsymbol{e_s} + \widehat{\boldsymbol{L}_{\boldsymbol{s}}}^{\dagger} \widehat{\frac{\partial \boldsymbol{e_s}}{\partial t}} \tag{3.23}$$

where $\widehat{\frac{\partial \boldsymbol{e_s}}{\partial t}}$ is an estimation or approximation of the features velocity due to the target motion. We proposed an estimation model within an Extended Kalman Filter to estimate this term, which will be illustrated in Chapter 5. More details on this subject can be found in [45].

3.4 Second-order Visual Servoing

Visual Servoing has been widely investigated in the last decades as it provides a powerful strategy for robot control. Thanks to the direct feedback from spatial sensors, it allows to reduce the impact of some modeling errors. The main

drawbacks with visual servoing, and more generally sensor-based control, is that robots can only be controlled through its end-effector twist or its joints velocities. These schemes provide kinematic control laws that do not take the robot dynamics into account. These control laws ensure an exponential decrease of the monitored values in sensor-space, under the assumptions given in Section 3.3.3, but when they are not met, tracking performance and convergence behavior can be poor. Furthermore, despite the linear convergence in sensor space, the end-effector's motion is not predictable. Despite there exist varying gain strategies that modify the temporal evolution of the system, the operational space dynamic behaviour could not be satisfactory when the robot dynamics starts playing a role (*e.g.,* for fast motions or when in contact with the environment).

Another approach that can be exploited consists in going one step further and to consider an acceleration model for the features. This strategy allows also to obtain a natural and direct link with the dynamic model of the controlled system, enabling full trajectory tracking in sensor-space.

In this section we briefly discuss the derivation of the second-order model for the visual-servoing that can be found in [46, 47].

Equation (3.3) expresses the link between the rate of change of the features and the sensor's velocity and differentiating it, one obtains the expression relating the features acceleration and the sensor and object velocity and acceleration as:

$$\ddot{\boldsymbol{s}} = \boldsymbol{L_s}\,({}^c\dot{\boldsymbol{v}}_{\boldsymbol{c}} - {}^c\dot{\boldsymbol{v}}_{\boldsymbol{o}}) + \dot{\boldsymbol{L}}_{\boldsymbol{s}}\,({}^c\boldsymbol{v_c} - {}^c\boldsymbol{v_o}) \tag{3.24}$$

or, when differentiating (3.19) (*eye-in-hand* mount), the link with the joint variables can be done:

$$\ddot{\boldsymbol{s}} = \dot{\boldsymbol{L}}_{\boldsymbol{s}}\,{}^c\mathbb{T}_e\,{}^e\boldsymbol{J}_e\,\dot{\boldsymbol{q}} + \boldsymbol{L_s}\,{}^c\dot{\mathbb{T}}_e\,{}^e\boldsymbol{J}_e\,\dot{\boldsymbol{q}} + \boldsymbol{L_s}\,{}^c\mathbb{T}_e\,{}^e\dot{\boldsymbol{J}}_e\,\dot{\boldsymbol{q}} + \boldsymbol{L_s}\,{}^c\mathbb{T}_e\,{}^e\boldsymbol{J}_e\,\ddot{\boldsymbol{q}} - \ddot{\boldsymbol{s}}_o. \tag{3.25}$$

Equation (3.25) can be rearranged in a more compact form as:

$$\ddot{\boldsymbol{s}} = \boldsymbol{J_s}\,\ddot{\boldsymbol{q}} + \boldsymbol{h_q} - \ddot{\boldsymbol{s}}_o \tag{3.26}$$

where $\boldsymbol{J_s} = \boldsymbol{L_s}\,{}^c\mathbb{T}_e\,{}^e\boldsymbol{J}_e$ denotes the so called *Feature Jacobian* [48] and:

$$\boldsymbol{h_q} = (\dot{\boldsymbol{L}}_{\boldsymbol{s}}\,{}^c\mathbb{T}_e\,{}^e\boldsymbol{J}_e + \boldsymbol{L_s}\,{}^c\dot{\mathbb{T}}_e\,{}^e\boldsymbol{J}_e + \boldsymbol{L_s}\,{}^c\mathbb{T}_e\,{}^e\dot{\boldsymbol{J}}_e)\dot{\boldsymbol{q}}. \tag{3.27}$$

Notice that for a *eye-in-hand* mount configuration the relative pose between the end-effector and the camera is fixed, then the twist-transformation matrix is constant, leading to ${}^c\dot{\mathbb{T}}_e = 0$.

Consider the manipulator model equation (2.27) in free space ($\boldsymbol{h}_e = 0$)

$$\boldsymbol{\tau} = \boldsymbol{B}\ddot{\boldsymbol{q}} + \boldsymbol{N} \tag{3.28}$$

in which we have compacted in $\boldsymbol{N} = \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{g} + \boldsymbol{F}_v\dot{\boldsymbol{q}} + \boldsymbol{F}_s\boldsymbol{sgn}(\dot{\boldsymbol{q}})$. The inertia matrix of the manipulator is a positive definite square matrix and is then always possible to invert. We can then explicit the joint acceleration from the previous $\ddot{\boldsymbol{q}} = \boldsymbol{B}^{-1}(\boldsymbol{\tau} - \boldsymbol{N})$ equation and inject it in equation (3.26). For a motionless target $\ddot{\boldsymbol{s}}_o = 0$, one obtains the link between the features acceleration and the joint actuation torques of the manipulator

$$\boldsymbol{J}_s\boldsymbol{B}^{-1}\boldsymbol{\tau} = \ddot{\boldsymbol{s}} - \boldsymbol{h}_{\boldsymbol{q}} + \boldsymbol{J}_s\boldsymbol{B}^{-1}\boldsymbol{N} \tag{3.29}$$

3.5 Conclusion

In this chapter we have introduced the basics of visual control, also known as visual servoing, which represents the second cardinal point of this work. On the one hand, we illustrated the terminology and nomenclature used, while on the other, we highlighted the first and second order relationships that exist between the relative motion of the visual features and the camera. The concepts shown in this chapter are of paramount importance in regards of the development of the results carried out in this thesis, as we will see from Part II onwards.

The next chapter lays the groundwork for the vision/force coupling discussed in the literature, which we used as a starting point for our research.

Chapter 4

Vision-Force control

Contents

4.1 Introduction

The sensory integration of vision and force, which combines visual servoing with force control, is at the heart of this thesis work. Over the last two decades, much research has been done on combining visual servoing techniques with force control to take advantage of the complementarity of vision and force sensing. Cameras are capable of providing a rich description of the scene, while force sensing can provide local information about the contact itself. A combined vision/force arrangement is becoming increasingly common, as in the context of physical human-robot interaction [2], due to the greater computing power available and the ever lower cost of vision systems, greatly increasing the spectrum of robotic tasks that may be accomplished. However, due to the very different nature of these two sensing modalities (*e.g.*, difference in the measured physical quantities, data rates, delays, etc.), obtaining an effective combined use of vision and force is not straightforward. Sensor integration approaches necessitate a shared representation of the sensory input being fused. This common data representation is not given for vision and force sensors [1]; the two sensing systems produce, as said before, fundamentally distinct quantities, namely force and position. Despite that, examples in which a combined use of vision and force sensing has been successfully achieved within an estimation (Bayesian [1, 49, 50] or incremental smoothing and mapping (iSAM) [51, 52]) framework are not isolated.

Transferring the problem from the perception level to the control level is another way to approach the problem of visual force sensor integration. Nelson [53] suggested three different ways for combining force and vision in a manipulator's feedback loop, which are: *traded*, *hybrid*, and *shared* control. For a given direction, traded control alternates between force control and visual servo in a mutually exclusive manner. Hybrid control, as for its position/force counterpart saw in Section 2.4.1, partitions the task directions into two orthogonal subspaces controlled separately by vision or force. Shared control offers the highest level of sensor integration allowing for simultaneous control with both vision and force along any task direction.

Most of the motion/force control strategies discussed in Chapter 2 have been revisited by researchers and expanded by essentially replacing the motion control loop with vision. As a result, those schemes may have the same strengths and limitations as their position-based counterparts, which we are about to discuss.

4.2 Hybrid Vision-Force control

Several hybrid vision/force control schemes have been proposed and applied [53]. All of them follow the same architecture shown in Figure 4.1, which is equivalent to the hybrid position/force control [14] presented in Section 2.4.1. The input to the vision control law (VCL) is computed from desired $\boldsymbol{s}^d$ and current $\boldsymbol{s}$ visual features. As for the position/force case, it is necessary to ensure orthogonality between vision $\boldsymbol{v}_v^d$ and force controlled subspaces $\boldsymbol{v}_f^d$ to avoid any conflict at the actuation level. Hybrid control separates vision control and force control into two separate control loops, that operate in orthogonal directions, as shown in Figure 4.1. A diagonal selection matrix $\boldsymbol{S}$ and its complementary $(\mathbb{I} - \boldsymbol{S})$ are introduced in the two control loops.

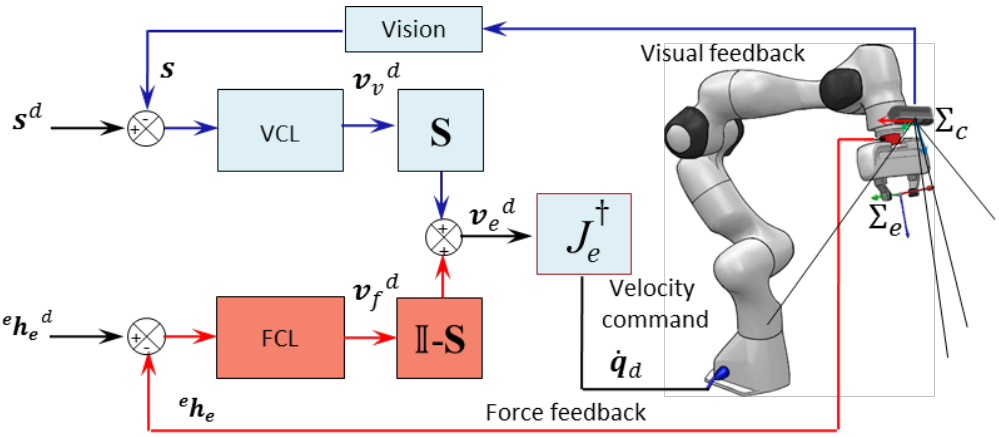


Figure 4.1 – Hybrid-based vision/force control scheme

The selection matrices are applied to vision and force errors defined in a common frame in operational space. The resulting motion is given by the output contributions of vision and force as follows:

$$\boldsymbol{v}_e^d = \boldsymbol{S}\boldsymbol{v}_v^d + (\mathbb{I} - \boldsymbol{S})\boldsymbol{v}_f^d \tag{4.1}$$

where $\boldsymbol{v}_v^d$ is the output of the Vision Control Law (VCL) according to the classical visual servoing kinematic controller seen in Chapter 3, and $\boldsymbol{v}_f^d$ is the output of the Force Control Law (FCL) taking into account the desired behavior of the system according to the relationship between force and differential displacement by means of the stiffness matrix ($\boldsymbol{h} = \boldsymbol{K}\, d\boldsymbol{X}_e$; $\dot{\boldsymbol{X}}_e = \dot{\boldsymbol{s}}_{3D} = \boldsymbol{L}_{\boldsymbol{X}}\, \boldsymbol{v}$, being $\boldsymbol{L}_{\boldsymbol{X}}$ the interaction matrix corresponding to the 3D pose definition $\boldsymbol{X}_e$, see Section 3.3.2). Visual servoing is only possible with hybrid control in directions that are orthogonal to the force feedback directions. The positioning error generated by a force perturbation in the force controlled direction cannot be corrected by the vision-based

controller. The range of possible tasks for this controller, on the other hand, is limited to those that can be described in terms of constraint surfaces [15].

Several improvements to this scheme were made by Baeten and De Schutter [3] who have allowed *shared* control in a hybrid scheme using, whether possible, vision-based feedforward terms along the force controlled directions, and vice versa, to improve the overall performance of the task.

4.3 Visual-Impedance control

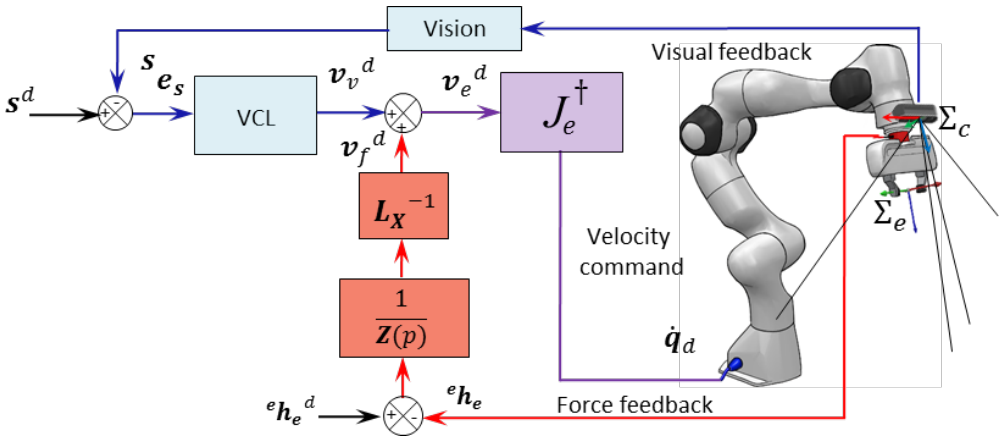


Figure 4.2 – Impedance-based vision/force control scheme

As shown in Section 2.4.2, the impedance position/force control framework allows to *a priori* define the way the manipulator shall react with respect to external force disturbance. Using such a scheme, the six degrees of freedom can be simultaneously position- and force-controlled. A programmable mechanical impedance $\boldsymbol{Z}$ is thus provided to the end-effector:

$$\boldsymbol{F}(p) = p\boldsymbol{Z}(p)\boldsymbol{X}(p) \tag{4.2}$$

The robot is supposed to be equivalent to a mass-spring-damper second-order system whose transfer function is:

$$p\boldsymbol{Z}(p) = \boldsymbol{M}p^2 + \boldsymbol{D}p + \boldsymbol{K} \tag{4.3}$$

where $\boldsymbol{M}, \boldsymbol{D}$ and $\boldsymbol{K}$ are respectively the desired inertia, damping and stiffness matrices.

Impedance control can be accomplished in a variety of ways. Torque-based impedance control, in which the low-level loop is a torque loop, and position-based impedance control, in which a low-level position controller is used, are the two

main categories. Torque-based impedance control should be preferred for general purposes in terms of performance because it can provide a wide range of target impedances, including soft behaviors [54]. When this method was proposed [55], the authors stated that *"torque-based impedance control was not relevant to most industrial applications, that used built-in joint position controllers in the robot's hardware controller. Thus, a position based impedance controller has been selected"*. By considering only a damping matrix in the position-based impedance control, the mechanical impedance becomes $\boldsymbol{Z}(p) = \boldsymbol{D}$, which is also known as *accommodation control* [56]. For vision/force impedance control (see Figure 4.2), the final control vector can be written as $\boldsymbol{v}_e^d = \boldsymbol{v}_v^d + \boldsymbol{v}_f^d$, where $\boldsymbol{v}_v^d$ is the kinematic screw computed by the vision control law, and $\boldsymbol{v}_f^d$ is computed by the force control loop. Thus, the expression for impedance vision/force control can be written as:

$$\boldsymbol{v}_e^d = -\lambda \boldsymbol{L}_{\boldsymbol{s}}^{\dagger}(\boldsymbol{s} - \boldsymbol{s}^d) + \boldsymbol{L}_{\boldsymbol{X}}^{-1}\boldsymbol{D}^{-1}({}^e\boldsymbol{h}_e^d - {}^e\boldsymbol{h}_e) \tag{4.4}$$

so that $\dot{\boldsymbol{X}}_e = \dot{\boldsymbol{s}}_{3D} = \boldsymbol{L}_{\boldsymbol{X}}\boldsymbol{v}_f^d$.

With such a serial scheme, it is possible that $\boldsymbol{s} \neq \boldsymbol{s}^d$ and ${}^e\boldsymbol{h}_e \neq {}^e\boldsymbol{h}_e^d$ while $\boldsymbol{v}_e^d = 0$. A local minimum can thus be attained or oscillatory phenomena can occur. Consider moving the end-effector parallel to the contact surface in order to reach the target. The vision system slips over the surface, guiding the end-effector to its final position without friction. The controller, however, would fail if friction were present. Indeed, the vision-based controller tangential velocity decreases as it gets closer to the final position and, because of the friction component, it will eventually become smaller than the opposite tangential velocity generated by the force feedback. To avoid the system from blocking with a pure damping impedance, the following condition must be met:

$$\|\boldsymbol{v}_v^d\| > \|\boldsymbol{v}_f^d\| = \boldsymbol{D}\|{}^e\boldsymbol{h}_e\| \tag{4.5}$$

Friction produces static positioning error when using a proportional visual control law like the one in equation (4.4). Although a proportional integral controller could be used, combining integral correction with vision feedback and friction non linearities could result in instabilities or in the emergence of limit cycles.

4.4 External Hybrid Vision-Force control

The External Hybrid Vision-Force control scheme was proposed by Mezouar et al. [57] in 2007. This scheme is based on the concept of external control, proposed by Perdereau et al. [58] in 1993. In this control scheme, force and vision

controllers are superposed in a hierarchical way. The juxtaposition of the force control loop over the vision control loop provides various benefits over hybrid- and impedance-based vision/force control, therefore addressing their shortcomings. The manipulator is equipped with a wrist-mounted force/torque sensor and a mounted camera (*eye-in-hand*).

More specifically, the force control loop is closed around an internal vision control loop. The desired vector of image observation $\boldsymbol{s}^d$, used as input to the vision-based controller, is modified by the force controller output, which is a relative position $d\boldsymbol{X}_e$, projected on sensor space by means of the interaction matrix $\boldsymbol{L_s}$. When the end-effector moves in free space, the output of the force controller is null and the robot is controlled according to the vision-based controller output. Otherwise the force controller modifies the reference trajectory of visual observations by adding a relative displacement $d\boldsymbol{s}$ to the reference. Let us see how this feature displacement $d\boldsymbol{s}$ is obtained.

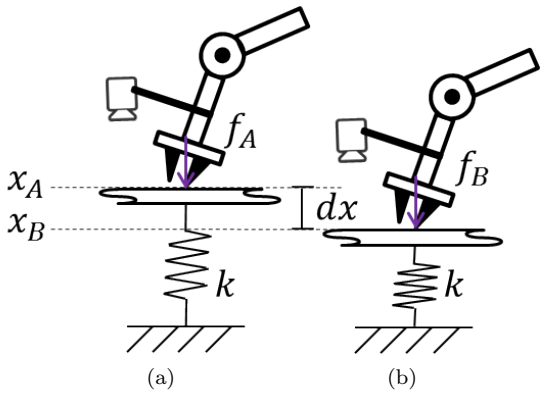


Figure 4.3 – 1D contact on a surface. Contact is modeled as a spring of stiffness k. (a) The manipulator tip applies a force f_A on a surface reaching position x_A. (b) The manipulator applies a greater force f_B on the contact surface penetrating in the material x_B. The force variation $(f_B - f_A)$ produces a displacement variation $dx = x_B - x_A$ proportional to the stiffness k.

Considering the 1D case in Figure 4.3, the force variation $(f_B - f_A)$ between the two configurations ((a) and (b)) is proportional to the environment stiffness k and the displacement dx according to the equation

$$f_B - f_A = k(x_B - x_A) = k\,dx \tag{4.6}$$

then, the displacement variation can be obtained as $dx = \frac{1}{k}(f_B - f_A)$. For the 3D case, one has to consider the *pose* variation (error $d\boldsymbol{X}_e$) of the manipulator's end-effector in contact with the environment, while forces translate to *wrenches*. In order to properly project afterwards the 3D displacement in the feature space, one need to express all the involved *twists* and *wrenches* in the same reference

frame. Since the camera is rigidly attached to the last link of the manipulator, the relative pose between the camera and the end-effector frames is constant, allowing to express the *wrench* acting on the end-effector into the camera frame thanks to the wrench transformation matrix (${}^c\boldsymbol{h}_c = {}^c\mathbb{F}_e\,{}^e\boldsymbol{h}_e$) [13]. So, equation (4.6) in 3D becomes

$$\boldsymbol{h}_B - \boldsymbol{h}_A = \boldsymbol{K}\, d\boldsymbol{X}_e \tag{4.7}$$

where $d\boldsymbol{X}_e$ is the end-effector pose error vector associated to the Homogeneous transformation matrix ${}^A\boldsymbol{T}_B(\boldsymbol{X}_A, \boldsymbol{X}_B) = {}^w\boldsymbol{T}_A^{-1}(\boldsymbol{X}_A){}^w\boldsymbol{T}_B(\boldsymbol{X}_B)$, and depends on the chosen orientation representation (in our case, the Axis/angle representation have been used). Since we are expressing all the quantities in the camera frame, in analogy with the pose-based visual servoing with 3D pose errors as features, the differential relationship between the error variation and the *twist* of the camera is

$$\dot{\boldsymbol{s}}_{3D} = \boldsymbol{L}_{\boldsymbol{X}}\,{}^c\boldsymbol{v}_c \rightarrow \frac{d\boldsymbol{X}_e}{dt} = \boldsymbol{L}_{\boldsymbol{X}}\frac{d\boldsymbol{X}}{dt} \rightarrow d\boldsymbol{X}_e = \boldsymbol{L}_{\boldsymbol{X}} d\boldsymbol{X} \tag{4.8}$$

which replaced in equation (4.7) and isolating the infinitesimal twist $d\boldsymbol{X}$ yields to

$$d\boldsymbol{X} = \boldsymbol{L}_{\boldsymbol{X}}^{-1}\,\boldsymbol{K}^{-1}(\boldsymbol{h}_B - \boldsymbol{h}_A) \tag{4.9}$$

Equation (4.8) holds true no matter which orientation representation is used, but the Jacobian that projects the velocities (in our case $\boldsymbol{L}_{\boldsymbol{X}}$) will change accordingly. One obtains then an infinitesimal displacement of the camera frame due to the forces/torques acting on this frame which have to be now projected in the feature space. By using the infinitesimal relationship in equation (4.8) but for a generic image feature ($\dot{\boldsymbol{s}} = \boldsymbol{L}_{\boldsymbol{s}}\,{}^c\boldsymbol{v}_c$) and replacing equation (4.9), one finally obtain the projected displacement in image space as:

$$d\boldsymbol{s} = \boldsymbol{L}_{\boldsymbol{s}}\,\boldsymbol{L}_{\boldsymbol{X}}^{-1}\,\boldsymbol{K}^{-1}(\boldsymbol{h}_B - \boldsymbol{h}_A) \tag{4.10}$$

which is the Force Control Law (FCL) used in the control scheme in Figure 4.4, where $\boldsymbol{h}_A$ and $\boldsymbol{h}_B$ are respectively the measured ${}^c\boldsymbol{h}_c = {}^c\mathbb{F}_e\,{}^e\boldsymbol{h}_e$ and the desired end-effector *wrenchs* ${}^c\boldsymbol{h}_c^d = {}^c\mathbb{F}_e\,{}^e\boldsymbol{h}_e^d$ projected in camera frame. On the other hand, the Vision Control Law (VCL) is the classical visual-servoing controller, saw in Chapter 3, that minimizes the error between the vector of the current features $\boldsymbol{s}$ and the vector of visual reference which was modified $\boldsymbol{s}^* = \boldsymbol{s}^d + d\boldsymbol{s}$. It finally outputs the desired end-effector *twist* $\boldsymbol{v}_e^d$.

In [57], authors have provided an exhaustive comparison of the External hybrid scheme against Hybrid- and Impedance-based vision/force schemes showing how this controller converges without problems whether the others have failed. The hierarchical juxtaposition of the force control loop on the vision control loop provides

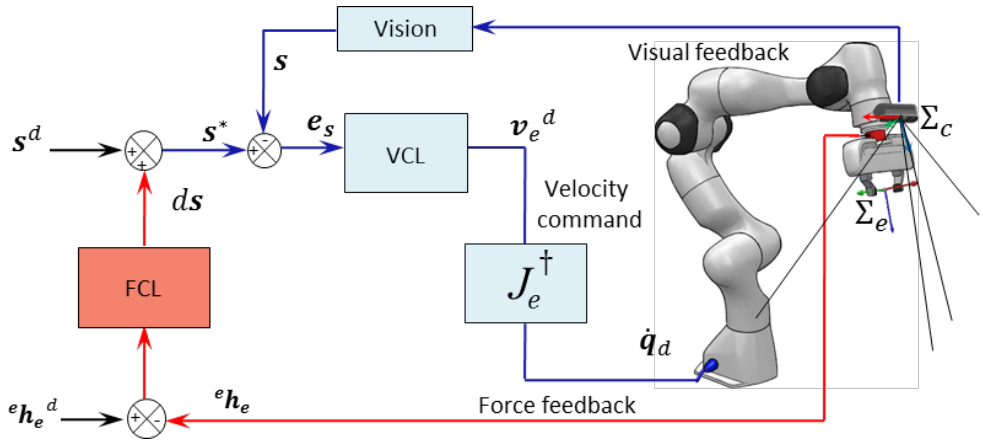


Figure 4.4 – External hybrid vision/force control scheme

several advantages according to the existing methods, *i.e.,* selection matrices and time-dependent geometric transformations are eliminated from the control loop, leading to a controller design independent of the arm configuration. Since the force control only acts on the reference trajectory, conflicts between force and vision controller are avoided. Furthermore, it has no effect on closed loop behavior, and therefore in its stability properties, since the wrench input can be viewed as a reference trajectory modifier [59].

As far as the external force loop is concerned, it should be stressed that it has no effect on the reference trajectory as long as the manipulator is moving in free space. The controller's behaviour and stability properties are exactly those of a classical visual servo scheme (inner loop), as shown in Chapter 3. Visual servoing schemes rely on the low-level manipulator's velocity controller to servo the robot. Such low-level controllers are usually high stiffness PID-like controllers implemented such as to guarantee that $\boldsymbol{q}(t) \approx \boldsymbol{q}_d(t)$ and the manipulator can be considered as an ideal positioning device. Once the end-effector comes into contact with the environment, the sensed wrench is transformed into a feature displacement $d\boldsymbol{s}$ through the inverse of the contact stiffness $\boldsymbol{K}$ (see equation (4.10)). If a soft contact behaviour is required for the manipulations task, the combination of the high stiffness of the joint controller and the consequent high gains involved in the external loop ($\boldsymbol{K}^{-1}$) could lead to instability as we will show in Section 6.6.3.

4.5 Constraint-based methods

Constraint-based methods are an alternative way to integrate vision and force sensing. There are different approaches in this category to perform the sensory integration, like the Stack-of-Task [60] or the Whole-body control framework [61].

The Stack-of-Task framework [60] is a generalized inverse kinematics abstraction layer that creates a hierarchical organization of different tasks, allowing them to be executed, for example, giving higher priority to critical tasks. The Whole-body control framework [61] formulation allows for secondary task objectives to be controlled without dynamically interfering with the operational task.

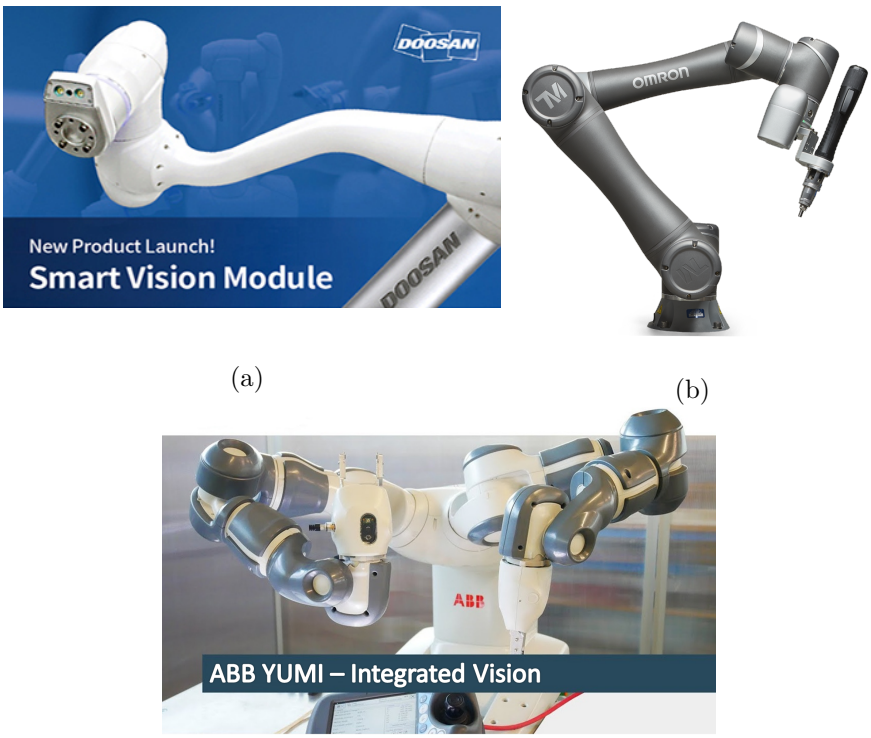
Despite the fact that these frameworks were not really designed for sensory integration, but rather for synthesizing the control commands that must comply with different types of constraints imposed on the system, they have been successfully used by [62] for combining vision and haptic (force) cues by coupling a visual servoing controller with an impedance controller within the Stack-of-Task framework [60]. In [63] it was shown that visual servoing tasks can be integrated into a prioritized constraint-based torque control framework formulating it as just another task/constraint for the whole-body controller [61] that can be combined with force control tasks.

Another family of more general constraint-based methods meant for specifying complex tasks of general sensor-based robot systems are represented by [64, 65]. The iTaSC framework [64], emerged as a specification formalism to generalise and extend existing frameworks like the Operational Space Approach [11], Task Function Approach [66], Task Frame Formalism [67], geometric Cartesian Space control, and Joint Space control. In this framework, the use of feature coordinates, defined with respect to object and feature frames, facilitates the task specification and the introduction of uncertainty coordinates to model geometric uncertainty. The etasl/etc [65] framework, is based on feature variables and the concept of expression graphs. It avoids some of the common pitfalls in previous frameworks, and provides a flexible and composable way to define robot control tasks. Those frameworks allow over-specifying constraints along a given direction and resolve the priority conflicts by adding weights to such constraints. The underlying optimization solver provides the joint commands. Using the [64] framework formalism, [68] reformulated the image-based visual servoing as a constraint-based robot task, allowing for potential integration with force controlled tasks using such framework.

We do not take into account any of these constraint-based strategies because the focus of our research regards the pure sensory coupling aspects. Furthermore, these frameworks have a high computational burden, limiting their applicability to a robot like the one used in this study, which requires a control rate of 1kHz to operate, if no further measures are taken.

4.6 Technological aspects

In the last 20 years the technological landscape of this field has changed considerably. While Baeten [3] stated at the beginning of the century that vision and force systems were rather confined to research areas with a few exceptions in the industrial field, today, we can see that some manufacturers already provide their own solutions deploying these systems directly integrated into their robots, as we can see in Figure 4.5. Other companies on the other side, provide solutions with standardised interfaces that can be adapted to different robots, as the *Robotiq*[1] solution for *Universal Robots*[2] shown in Figure 4.6, thus underlining the interest in adopting this technology in industry due to their potential.



(a) (b)

(c)

Figure 4.5 – (a) Doosan Cobot M-Series: it is equipped with an optional "in house" smart vision module. (b) Cobot Omron TM: is equipped with a screwing tool and a vision system with lighting function for image enhancement. (c) ABB YuMi presents an integrated camera on its gripper. All this robots are equipped with joint torque sensors.

[1] https://robotiq.com/
[2] https://www.universal-robots.com/

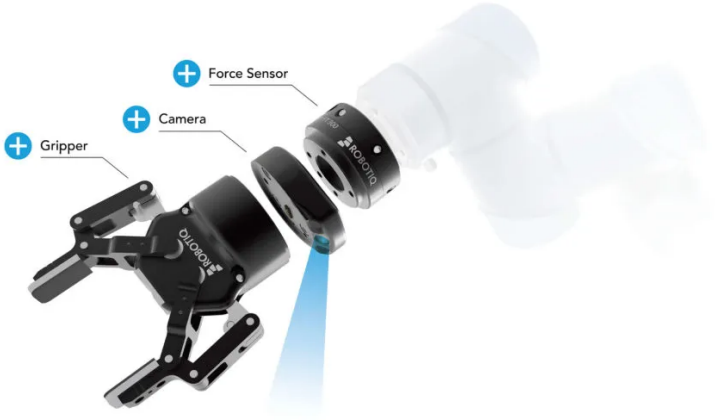


Figure 4.6 – Robotiq integrated solution for cascading wrist-mounted Force/Torque sensor, vision system and two finger gripper designed for the Universal Robots.

The adoption of built-in torque sensors into the joints and the significant advances in collision detection techniques [69] have allowed manipulators to break out of the safety cages in which they were confined to work, allowing them to be used alongside human operators with a high degree of safety. Joint torque sensors also allows to estimate the external wrench at the end-effector, avoiding the need of an external Force/Torque sensor, although they are less accurate. Despite the increasingly massive use of cameras in robotics applications, the use of combined techniques of vision and force sensing does not present today the same level of maturity as that demonstrated by both the collision detection and the use of force sensing for interaction tasks. Cameras appear to be used primarily for objects detection in the workspace and afterwards the interaction is managed using traditional force control approaches (e.g., impedance control) not fully exploiting the capabilities and complementarity of both sensors during all the execution phases of the task at hand.

The objective of this thesis work is to provide new integrated vision/force control strategies to meet market demands for this type of integrated solution by closing the gap between existing technology and available control techniques.

In the next part, we deal with the problem of *shared* sensor integration, proposing both dynamic (Chapter 5) and kinematic (Chapter 6) controllers that "fuse" the sensing modalities in sensor space while we tackle several aspects related to the implementation of the dynamic controller.

Part II Feature Space Compliance

Compliant motion occurs whenever a manipulator performs a task while maintaining the contact between the end-effector and the environment. When imposing a PD-like controller to the manipulated system, its behaviour is similar to a *mass-spring-damper* under the action of an external force. This part of the thesis is devoted to illustrate our contributions on the derivation of controllers that can deal with compliant motion along/about the visual features defining the visual task, see Figure Part II. In the case of PBVS, a 3D pose reconstruction phase is performed to estimate the position and orientation of the observed object in *operational space*. The visual feature in this case is the pose error vector between the actual and desired camera frames. In the IBVS case instead, the visual features (points, lines, moments, etc.) are directly extracted on the image plane. In this work we refer to the more general space of all possible kinds of visual features as the *Visual Feature Space*.

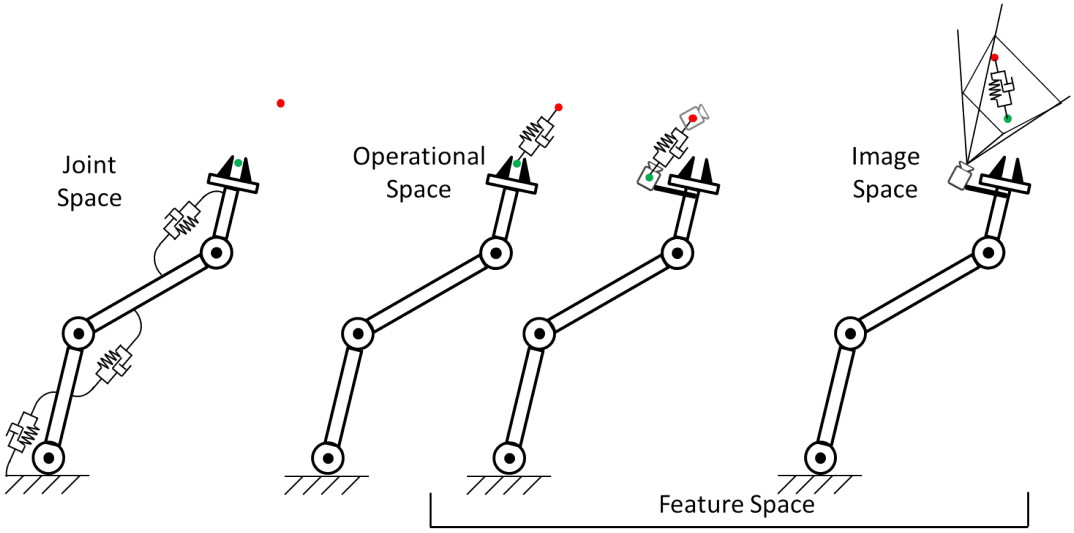


Figure Part II – Compliance achieved in different spaces: a PD-like controller brings the actual configuration (green dot) into the desired configuration (red dot), from left to right, in the joint space, in the operational space, in the operational space in the case PBVS and in the image space. We refer to the *Feature Space* as the more general space containing any kind of visual features, being them the parameters of a 3D pose, of image points, image lines, image moments, etc.

Chapter 5

Feature Space Impedance

Contents

5.1 Introduction

As discussed in Chapter 3, Visual Servoing is a consolidated control technique that allows to precisely position a vision sensor with respect to an observed object in the scene. In the past decades, most research efforts focused on the modeling of the existing relationships between descriptive features in the image and the motion of the sensor at kinematic level [40,41], and in the development of control strategies to guide the chosen visual features towards the desired ones as summarized in [44,45]. Although much research has been conducted in this sense, little attention has been devoted to the use of second order models linking the features accelerations to the robot torques via the robot dynamical model, with some early attempts dating back more than two decades [40,70] up to some more recent contributions [46,47]. As well-known, explicitly taking into account the robot dynamics allows to design controllers with superior performance, especially for what concerns the regulation of forces and interaction with the environment. For instance, in many industrial applications the interaction tasks are executed in static or quasi-static conditions and can be solved at kinematic level. However if one wishes to reduce the execution time, faster motions are required making the control task essentially dynamic [71].

Most of the above cited works mainly focused on the motion of the robot in free space without taking into account possible physical interactions with the environment. Indeed while [70] only compensates for the air drag forces acting on the blimp, in [46], second-order visual-servoing is used to control solely the unconstrained directions of an engraving task within a hybrid control scheme [14] with a pure force controller acting along/about the constrained directions.

The goal of this chapter is to study the possibility of using second-order visual servoing for interaction control, since, as we will show in Section 5.3, a second-order visual-servoing controller under interaction with the environment boils down to an impedance controller in the visual feature space, which is a force control scheme belonging to the category of indirect force control methods [13].

In [6] which is further developed in the next chapter, we proposed an alternative to the impedance control in feature space - which results to be complex to implement given the limitations discussed and addressed in this chapter - by defining an admittance in feature space.

A contribution of this work is, therefore, to overcome these difficulties (*i.e.,* low camera data rate, feature extraction delays, ill-conditioning of the task Jacobian, etc) and propose a true second-order visual servoing in feature space that can also take into account the possible, and generally unknown, motion of the observed target object to interact with the environment. This is achieved by modeling the

motion of the target in a second-order visual-servoing setting and by estimating its unknown motion via a suitable Extended Kalman Filter (EKF). The EKF is also instrumental for virtually increasing the data rate of the employed camera by providing high-rate estimations of the quantities needed by the derived torque-level controller, without resorting to dedicated high frequency, and therefore expensive vision systems like those in [5, 72]. The derivations of the filters' states are specialized for the cases of Image-Based Visual Servoing (IBVS) with *image points* as visual features and Pose-Based Visual Servoing (PBVS) while their respective similarities and differences are pointed out.

The filters derived in this chapter can also be easily integrated into the control scheme presented in the next chapter, both in case of static or moving target, although the discussion and respective experiments were carried out considering static targets only. The use of such filters, and the consequent increased data rate, would make the system even more responsive, particularly on rigid contacts.

5.2 Manipulator model in feature space

In this section, the manipulator's model in feature space under interaction with the environment is derived thanks to a suitable coordinates transformation exploiting the second order visual-servoing model derived in Section 3.4

The *Lagrangian* equation of motion of the manipulator (2.27) is a set of n nonlinear and coupled second-order differential equations for which the *inverse dynamics control* is a well-known control strategy for trajectory tracking aiming at linearizing and decoupling the manipulator's dynamics via feedback linearization, assuming that its model is perfectly known. We can rewrite equation (2.27) such as to explicit the joint accelerations:

$$\ddot{\boldsymbol{q}} = \boldsymbol{B}(\boldsymbol{q})^{-1}(\boldsymbol{\tau} - \boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - {}^{e}\boldsymbol{J}_{e}^{\top}\,{}^{e}\boldsymbol{h}_{e}) \tag{5.1}$$

having compacted in $\boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = (\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}))$ the terms relatives to the Coriolis, centrifugal and gravitational force effects. After replacing equation (5.1) into the second order visual servoing equation (3.26) and considering the observed target to be motionless (${}^{c}\boldsymbol{v}_{\boldsymbol{o}} = 0$), we find the law that governs the features motion:

$$\boldsymbol{J}_{\boldsymbol{s}}\boldsymbol{B}(\boldsymbol{q})^{-1}\boldsymbol{\tau} = \ddot{\boldsymbol{s}} - \boldsymbol{h}_{\boldsymbol{q}} + \boldsymbol{J}_{\boldsymbol{s}}\boldsymbol{B}(\boldsymbol{q})^{-1}\boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{J}_{\boldsymbol{s}}\boldsymbol{B}(\boldsymbol{q})^{-1}\,{}^{e}\boldsymbol{J}_{e}^{\top}\,{}^{e}\boldsymbol{h}_{e} \tag{5.2}$$

Taking into account the expression of the *feature Jacobian* $\boldsymbol{J}_{\boldsymbol{s}} = \boldsymbol{L}_{\boldsymbol{s}}\,{}^{c}\mathbb{T}_{e}\,{}^{e}\boldsymbol{J}_{e}$, the wrench-transformation matrix that projects the external *wrench* acting on the camera into the end-effector ${}^{e}\boldsymbol{h}_{e} = {}^{e}\mathbb{F}_{c}\,{}^{c}\boldsymbol{h}_{c}$, and recalling the relationship between

the *twist* and *wrench* transformation matrices ${}^{c}\mathbb{T}_{e}^{\top} = {}^{e}\mathbb{F}_{c}$ [13], we can rearrange the last member of (5.2) as:

$$\boldsymbol{L_s}\,{}^{c}\mathbb{T}_{e}\,{}^{e}\boldsymbol{J}_{e}\boldsymbol{B}(\boldsymbol{q})^{-1}\,{}^{e}\boldsymbol{J}_{e}^{\top}\,{}^{c}\mathbb{T}_{e}^{\top}\,{}^{c}\boldsymbol{h}_{c} = \boldsymbol{L_s}\boldsymbol{B}_{c}(\boldsymbol{q})^{-1}\,{}^{c}\boldsymbol{h}_{c} \tag{5.3}$$

where $\boldsymbol{B}_{c}(\boldsymbol{q})^{-1} = {}^{c}\mathbb{T}_{e}\,{}^{e}\boldsymbol{J}_{e}\,\boldsymbol{B}(\boldsymbol{q})^{-1}\,{}^{e}\boldsymbol{J}_{e}^{\top}\,{}^{c}\mathbb{T}_{e}^{\top}$ is the inverse of the manipulator inertia matrix projected in the camera frame. Equation (5.3) highlights how the external *wrench* acting on the camera frame and expressed in the same frame ${}^{c}\boldsymbol{h}_{c}$, projects in feature space as a virtual per unit of mass/inertia forces/torques (p.u.m.i.) (accelerations) acting on the image features. By renaming some terms in (5.2) as:

$$\boldsymbol{f_s} = \boldsymbol{J_s}\boldsymbol{B}(\boldsymbol{q})^{-1}\boldsymbol{\tau} \tag{5.4a}$$

$$\boldsymbol{b_s} = \boldsymbol{J_s}\boldsymbol{B}(\boldsymbol{q})^{-1}\boldsymbol{N}(\boldsymbol{q},\dot{\boldsymbol{q}}) \tag{5.4b}$$

$$\boldsymbol{f}_{s_{ext}} = \boldsymbol{L_s}\boldsymbol{B}_{c}(\boldsymbol{q})^{-1}{}^{c}\boldsymbol{h}_{c} \tag{5.4c}$$

one obtains the manipulator model in feature space:

$$\ddot{\boldsymbol{s}} - \boldsymbol{h_q} + \boldsymbol{b_s} + \boldsymbol{f}_{s_{ext}} = \boldsymbol{f_s} \tag{5.5}$$

The obtained model presents an analogy with the joint space model (2.27) but the acceleration term $\ddot{\boldsymbol{s}}$ do not present any inertia matrix, this means that the model is inertia-normalized, and the quantities in equation (5.5) are not forces and/or torques but they are relative forces and/or torques per unit of mass/inertia; indeed they have the physical dimension of an acceleration (m/s^2).

5.3 Impedance control in feature space for a static target

As we have seen in Chapter 2, impedance control aims at achieving a dynamic behaviour of the robot end-effector in contact with the environment equivalent to a *mass-spring-damper* system subject to an external force. Our purpose here is to replicate this behaviour between the current and desired visual features lying on the image plane. Defining the image feature error vector $\boldsymbol{e_s} = (\boldsymbol{s}^{d} - \boldsymbol{s}) \in \mathbb{R}^{k}$ as the difference between the desired $\boldsymbol{s}^{d}$ and current $\boldsymbol{s}$ visual features, or, in the case of 3D pose, $\boldsymbol{e_s} \in \mathbb{R}^{6}$ as the pose error vector between the current Σ_c and desired Σ_d camera frames as depicted in Figure 5.1, we formally want to obtain the following behaviour in feature space:

$$\boldsymbol{M_s}\ddot{\boldsymbol{e}}_{\boldsymbol{s}} + \boldsymbol{D_s}\dot{\boldsymbol{e}}_{\boldsymbol{s}} + \boldsymbol{K_s}\boldsymbol{e_s} = \boldsymbol{f} \tag{5.6}$$

Figure 5.1 – Desired *mass-spring-damper* behavior in the case of 3D pose (left) and four image point visual features (right). The current Σ_c and desired Σ_d camera frames are sketched as well as the object frame Σ_o. The current $\boldsymbol{s}$ (green) and the desired $\boldsymbol{s}^d$ (red) image point features in the image plane are depicted. The target is arranged in a coplanar square shape.

where $\boldsymbol{M_s}, \boldsymbol{D_s}$ and $\boldsymbol{K_s}$ are positive definite $k \times k$ matrices representing the relative per unit mass/inertia virtual mass, viscous and stiffness parameters of the impedance equation while $\boldsymbol{f} \in \mathbb{R}^k$ is a vector of per unit mass/inertia virtual forces (accelerations) acting on the features.

Let us suppose at this stage not to have any force measurement. The new control input for the manipulator model in feature space (5.5) is the vector of per unit of mass/inertia virtual forces $\boldsymbol{u_s} = \boldsymbol{f_s}$ which is chosen to compensate for any dynamic term in the model (5.5). One then has:

$$\boldsymbol{u_s} = \boldsymbol{\alpha} - \boldsymbol{h_q} + \boldsymbol{b_s} \tag{5.7}$$

and by injecting it back into (5.5) yields:

$$\boldsymbol{\alpha} = \ddot{\boldsymbol{s}} + \boldsymbol{f}_{s_{ext}} \tag{5.8}$$

in which $\boldsymbol{\alpha} \in \mathbb{R}^k$ represents a resolved acceleration in feature space and constitutes the new control input that has to be opportunely designed. A natural choice for $\boldsymbol{\alpha}$ is a PD controller with acceleration *feed-forward*:

$$\boldsymbol{\alpha} = \ddot{\boldsymbol{s}}^d + \boldsymbol{D_s}\dot{\boldsymbol{e}}_{\boldsymbol{s}} + \boldsymbol{K_s}\boldsymbol{e_s} \tag{5.9}$$

that replaced in (5.8) leads to the *closed-loop dynamics* of the system:

$$\ddot{\boldsymbol{e}}_{\boldsymbol{s}} + \boldsymbol{D_s}\dot{\boldsymbol{e}}_{\boldsymbol{s}} + \boldsymbol{K_s}\boldsymbol{e_s} = \boldsymbol{f}_{s_{ext}} \tag{5.10}$$

The behavior of the closed-loop system is the one sought in (5.6) in which $\boldsymbol{M_s}$ coincides with the identity matrix. In static conditions $\ddot{\boldsymbol{e}}_{\boldsymbol{s}} = \dot{\boldsymbol{e}}_{\boldsymbol{s}} = 0$, when the manipulator is in free space ($\boldsymbol{f}_{s_{ext}} = 0$), it is easy to verify that the current position of the visual features reaches the desired one ($\boldsymbol{s} = \boldsymbol{s}^d$) while when it is

in contact with the environment, the static error is inversely proportional to the relative per unit of mass/inertia stiffness matrix $\boldsymbol{K_s}$. Indeed, when $\boldsymbol{f}_{s_{ext}} \neq 0$ in equation (5.10), the position error of the visual features is given by:

$$\boldsymbol{e_s} = \boldsymbol{K_s}^{-1} \boldsymbol{f}_{s_{ext}} = \boldsymbol{K_s}^{-1} \boldsymbol{L_s} \boldsymbol{B}_c(\boldsymbol{q})^{-1c} \boldsymbol{h}_c \tag{5.11}$$

As we can see from the previous equation, the error is configuration dependent due to the presence of the inertia matrix of the robot projected in camera frame.

By replacing (5.9) into (5.7) in which the feature space per unit of mass/inertia force/torque vector $\boldsymbol{u_s}$ has the form (5.4a), leads to the resulting joint space controller:

$$\boldsymbol{u_\tau} = (\boldsymbol{J_s B}(\boldsymbol{q})^{-1})^{\dagger} (\ddot{\boldsymbol{s}}^d + \boldsymbol{D_s} \dot{\boldsymbol{e}}_s + \boldsymbol{K_s e_s} + \boldsymbol{J_s B}(\boldsymbol{q})^{-1} \boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{h_q}) \tag{5.12}$$

Even though we have considered the manipulator interacting with the environment in our derivation, if we do not have any force measurement, we arrive at the same control law derived in [46, 47]. In fact, this is analogous to the impedance control in the operational space case which is an inverse dynamic controller for trajectory tracking in free space whose properties when in contact with the environment have been taken into account.

Note that, in analogy to what was said in Subsection 3.3.3, in the IBVS case with $k > 6$ features, there is the theoretical risk, although very unlikely in practice, to specify unfeasible behaviours when the commanded feature space acceleration lies in $\ker((\boldsymbol{J_s B}(\boldsymbol{q})^{-1})^{\dagger})$ (see (5.12)). If for instance, one starts from a static position and servo the robot to reach a desired constant one with respect to a motionless target, from (5.12) one has $\boldsymbol{u_\tau} = (\boldsymbol{J_s B}(\boldsymbol{q})^{-1})^{\dagger} (\boldsymbol{K_s e_s})$; if $\boldsymbol{K_s e_s} \in \ker((\boldsymbol{J_s B}(\boldsymbol{q})^{-1})^{\dagger})$, the generated torque command $\boldsymbol{u_\tau} = 0$ even if $\boldsymbol{e_s} \neq 0$.

The drawback of this controller (5.12) is that the closed-loop system (5.10) exhibits a configuration-dependent compliant behavior due to the presence of the inertia matrix of the manipulator in the external *wrench* projection into the feature space (see (5.11)). An easy way to overcome this issue and render the manipulator behaviour isotropic with respect to the external *wrench*, *i.e.*, letting a given external *wrench* in Cartesian space be mapped into the image space generating the same per unit of mass/inertia force/torque vector regardless of the manipulator's configuration, is to measure the external forces being applied on the robot and fully compensate for them. This will make the manipulator infinitely rigid with respect to the measured forces but, thanks both to the fact that we have this measure and that we know how to project the wrenches into the feature space, we can impose the desired system's behavior during the interaction.

Let us suppose to have a wrist mounted force/torque sensor that provides the measured external *wrench* at the manipulator's end-effector ${}^{e}\boldsymbol{h}_{e}$. We can now add it in the feature space controller (5.7) to fully compensate for those forces as:

$$\boldsymbol{u_s} = \boldsymbol{\alpha} - \boldsymbol{h_q} + \boldsymbol{b_s} + \underbrace{\boldsymbol{L_s}\boldsymbol{B}_c(\boldsymbol{q})^{-1}\overbrace{{}^{c}\mathbb{F}_{e}\,{}^{e}\boldsymbol{h}_{e}}^{{}^{c}\boldsymbol{h}_{c}}}_{\boldsymbol{f_{s_{ext}}}} \tag{5.13}$$

Plugging this back again into (5.5), one obtains:

$$\ddot{\boldsymbol{s}} = \boldsymbol{\alpha} \tag{5.14}$$

For control purposes it is convenient to impose a constant apparent inertia matrix of the camera in order to have an homogeneous behavior along the Cartesian directions of the camera frame and manage the compliance along the visual features by opportunely tuning the relative stiffness $\boldsymbol{K_s}$ and damping $\boldsymbol{D_s}$ matrices of the impedance controller.

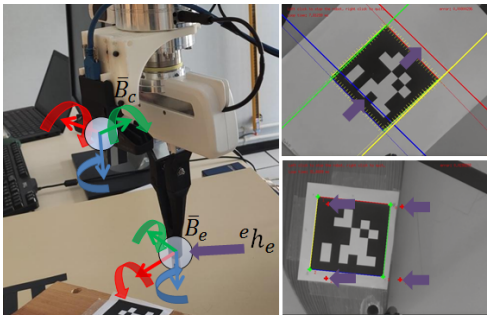


Figure 5.2 – By assigning a constant apparent inertia to the end-effector or camera frame, we impose the desired interaction behavior along/about the Cartesian directions. We choose an homogeneous inertial behavior so that the end-effector is equivalent to a sphere of mass. Applied forces on the end-effector (purple arrow on the left image) will be homogeneously distributed among the visual features (small arrows acting on the image lines (top-right) and image points (bottom-right)).

This can be done by choosing as resolved accelerations in feature space the controller:

$$\boldsymbol{\alpha} = \ddot{\boldsymbol{s}}^{*} + \boldsymbol{D_s}\dot{\boldsymbol{e}}_{\boldsymbol{s}} + \boldsymbol{K_s}\boldsymbol{e_s} - \boldsymbol{L_s}\bar{\boldsymbol{B}}_{c}^{\,-1}\,{}^{c}\mathbb{F}_{e}\,{}^{e}\boldsymbol{h}_{e} \tag{5.15}$$

being $\bar{\boldsymbol{B}}_c = diag(m_{c_d}, m_{c_d}, m_{c_d}, J_{c_d}, J_{c_d}, J_{c_d})$ in which m_{c_d} $[kg]$ and J_{c_d} $[kg.m^2]$ are respectively the desired apparent mass and inertia that the camera should exhibit. Industrial robots typically interact with the environment using a tool (the tool tip), it can be useful then to exhibit isotropy on the end-effector frame (tool tip frame) by choosing a constant diagonal inertia matrix $\bar{\boldsymbol{B}}_e$, as done before for $\bar{\boldsymbol{B}}_c$,

and then project it in the camera frame as $\bar{\boldsymbol{B}}_c^{-1} = {}^c\mathbb{T}_e \bar{\boldsymbol{B}}_c^{-1} {}^c\mathbb{T}_e^\top$. The resulting joint space controller is therefore:

$$\boldsymbol{u_\tau} = (\boldsymbol{J_s B}(\boldsymbol{q})^{-1})^\dagger \Big(\ddot{\boldsymbol{s}}^* + \boldsymbol{D_s}\dot{\boldsymbol{e}}_{\boldsymbol{s}} + \boldsymbol{K_s e_s} + \boldsymbol{J_s B}(\boldsymbol{q})^{-1}\boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{h_q} + \boldsymbol{f}_{s_{ext}} - \boldsymbol{L_s}\bar{\boldsymbol{B}}_c^{-1} {}^c\mathbb{F}_e {}^e\boldsymbol{h}_e \Big) \tag{5.16}$$

The closed-loop dynamics of the system is obtained by replacing (5.15) into (5.14) resulting in:

$$\ddot{\boldsymbol{e}}_{\boldsymbol{s}} + \boldsymbol{D_s}\dot{\boldsymbol{e}}_{\boldsymbol{s}} + \boldsymbol{K_s e_s} = \boldsymbol{L_s}\bar{\boldsymbol{B}}_c^{-1} {}^c\mathbb{F}_e {}^e\boldsymbol{h}_e \tag{5.17}$$

Now, the projected external forces in feature space do not depend on the manipulator's configuration anymore (see (5.11)). This translates in a more intuitive tuning of the impedance parameters since this controller would make the manipulator under interaction with the environment behave in the same manner no matters its configuration while preserving the same stability properties of the previous controller (5.12).

5.4 Impedance control in feature space for a moving target

In this section we will consider the more general case of tracking a moving target and, for simplicity of treatment, we will suppose not to have any force/torque sensor mounted on the robot but that at least an estimate of the features' velocity and acceleration due to the target motion $(\dot{\boldsymbol{s}}_o, \ddot{\boldsymbol{s}}_o)$ is available. In the next section we will detail how this estimation is obtained. Notice that the case in which a Force/torque sensor is available is formally analogous.

As done in the previous section, replacing (5.1) into the full equation of the second-order visual-servoing (3.26) results into:

$$\boldsymbol{J_s B}(\boldsymbol{q})^{-1}\boldsymbol{\tau} = \ddot{\boldsymbol{s}} + \ddot{\boldsymbol{s}}_o - \boldsymbol{h_q} + \boldsymbol{J_s B}(\boldsymbol{q})^{-1}\boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{J_s B}(\boldsymbol{q})^{-1} {}^e\boldsymbol{J}_e^\top {}^e\boldsymbol{h}_e \tag{5.18}$$

for which we rename the terms as in (5.4a)-(5.4c) obtaining:

$$\ddot{\boldsymbol{s}} + \ddot{\boldsymbol{s}}_o - \boldsymbol{h_q} + \boldsymbol{b_s} + \boldsymbol{f}_{s_{ext}} = \boldsymbol{f_s}. \tag{5.19}$$

We choose then a feature space control input $\boldsymbol{u_s}$ such as to compensate for any dynamic term, including the (usually unknown) acceleration term due to the target's motion

$$\boldsymbol{u_s} = \boldsymbol{\alpha} - \boldsymbol{h_q} + \boldsymbol{b_s} + \ddot{\boldsymbol{s}}_o \tag{5.20}$$

We then use the same acceleration resolved controller $\boldsymbol{\alpha}$ as in equation (5.9) and recalling the feature error definition $\boldsymbol{e_s} = \boldsymbol{s}^d - \boldsymbol{s}$, its time derivative $\dot{\boldsymbol{e}}_{\boldsymbol{s}} = \dot{\boldsymbol{s}}^d - \dot{\boldsymbol{s}}$ and equation (3.19), leading to the joint space torque controller

$$\boldsymbol{u_\tau} = (\boldsymbol{J_s B}(\boldsymbol{q})^{-1})^{\dagger} \Big(\ddot{\boldsymbol{s}}^* + \ddot{\boldsymbol{s}}_o + \boldsymbol{D_s}(\dot{\boldsymbol{s}}^d - \boldsymbol{J_s}\dot{\boldsymbol{q}} + \dot{\boldsymbol{s}}_o) + \boldsymbol{K_s}(\boldsymbol{s}^d - \boldsymbol{s}) + \boldsymbol{J_s B}(\boldsymbol{q})^{-1} \boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{h_q} \Big) \tag{5.21}$$

where the new terms $\ddot{\boldsymbol{s}}_o$ and $\boldsymbol{D_s}\dot{\boldsymbol{s}}_o$ compensate for the target motion.

All the considerations we have done in the previous section concerning the dependency on the manipulator's configuration on the wrench mapping into the feature space holds true. Following the same reasoning done before we can obtain an isotropic and configuration independent interaction behaviour, also in the case of a moving target, by measuring the contact forces/torques.

5.5 Increasing the visual data rate and estimating the target's motion

In the previous section, we derived a controller that linearizes and decouples the dynamics of the system via feedback linearization but, even though most of the terms in the controller can be easily computed, those related to the target own motion are usually unknown and must thus be estimated. Since the target motion is generally unknown, the simplest assumption one can do is to consider a constant acceleration model corrupted by Gaussian noise. The target motion can be treated as a disturbance [73] of the nominal motion of the features due to the motion of the camera as in (3.3), in which part of the feature velocity is given by the (unknown) target velocity $\dot{\boldsymbol{s}}_o = \boldsymbol{L_s}\, {}^{\boldsymbol{c}}\boldsymbol{v_o}$.

To estimate the target motion we now describe two instances of an EKF for the cases of image points and 3D pose error as visual features respectively. Note that both filters could be used in a classical VS control scheme to compensate for the target motion while here we exploit the entire state of the filters to implement the control law (5.21) for (i) virtually increasing the visual measurements from the camera (by running the filter at much higher frequency than the camera itself), and for (ii) compensating for the target own motion and, thus, significantly reducing the tracking error.

5.5.1 EKF for IBVS with image points features

The goal of the filter is to both increase the information rate of the camera and to estimate the velocity of the moving target. The feature vector $\boldsymbol{s}$ will then be part of the filter state whose dynamics following the visual-servoing kinematic model (3.19). For each image point feature $f_i = (f_{x_i}, f_{y_i}), \quad \forall i = 1, \ldots, 4$, the associated interaction matrix (3.7) (see Section 3.3.1) requires the knowledge of the depth of the point Z which is also estimated by the filter as long as the trajectories are "exciting" enough. The depth dynamic is given by [74]

$$\dot{Z} = \boldsymbol{L}_Z(f, Z)\, {}^c\boldsymbol{v} = \begin{bmatrix} 0 & 0 & -1 & -f_y Z & f_x Z & 0 \end{bmatrix} {}^c\boldsymbol{v}. \tag{5.22}$$

Furthermore, to implement controller (5.21), the feature acceleration contribution due to the motion of the target ($\ddot{\boldsymbol{s}}_o$) is also needed. The continuous-time state of the filter, stacking here four image points ($\boldsymbol{s} \in \mathbb{R}^8$), is then $\boldsymbol{x}(t) = [\boldsymbol{s}(t)\ \boldsymbol{z}(t)\ \dot{\boldsymbol{s}}_o(t)\ \ddot{\boldsymbol{s}}_o(t)]^\top = [\boldsymbol{x_1}(t)\ \boldsymbol{x_2}(t)\ \boldsymbol{x_3}(t)\ \boldsymbol{x_4}(t)]^\top \in \mathbb{R}^{28}$, and exploiting the filter dynamics $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) + \boldsymbol{w_t}$ one has:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{L_s}(\boldsymbol{x_1}, \boldsymbol{x_2}) {}^c\mathbb{T}_e\, {}^e\boldsymbol{J}_e \boldsymbol{u} - \boldsymbol{x_3} \\ \boldsymbol{L_Z}(\boldsymbol{x_1}, \boldsymbol{x_2})({}^c\mathbb{T}_e\, {}^e\boldsymbol{J}_e \boldsymbol{u} - \boldsymbol{L}_s^\dagger(\boldsymbol{x_1}, \boldsymbol{x_2})\boldsymbol{x_3}) \\ \boldsymbol{x_4} \\ \boldsymbol{0} \end{bmatrix} + \boldsymbol{w_t} \tag{5.23}$$

in which $\boldsymbol{u}(t) = \dot{\boldsymbol{q}}(t)$ is the plant input and $\boldsymbol{w_t} = [\boldsymbol{w}_{\boldsymbol{1_t}}^\top\ \boldsymbol{w}_{\boldsymbol{2_t}}^\top\ \boldsymbol{w}_{\boldsymbol{3_t}}^\top\ \boldsymbol{w}_{\boldsymbol{4_t}}^\top]^\top \in \mathbb{R}^{28}$ is a vector of additive Gaussian noise $\boldsymbol{w_t} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k)$. In the general case one does not have a model of the target motion, therefore we assume the dynamics of the acceleration of the features due to the target motion $\ddot{\boldsymbol{s}}_o(t)$ to be approximately constant (corrupted by some added noise), i.e., $\dddot{\boldsymbol{s}}_o(t) = \boldsymbol{w_{4_t}}$. Clearly if an accurate motion model was available, it could be used, adjusting opportunely the filter dynamics.

To update the process state estimate with the dynamics and its uncertainty, represented by the covariance matrix $\boldsymbol{P}_k$, we firstly need to linearize and discretize it at the control period ΔT:

$$\bar{\boldsymbol{A}} = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_k, \boldsymbol{u}=\boldsymbol{u}_k} \tag{5.24}$$

$$\boldsymbol{A}_k = e^{\bar{\boldsymbol{A}}\Delta T} \approx \mathbb{I} + \bar{\boldsymbol{A}}\Delta T \tag{5.25}$$

$$\boldsymbol{x}_{k+1|k} = \boldsymbol{x}_{k|k} + \boldsymbol{f}(\boldsymbol{x}_{k|k}, \boldsymbol{u}_k)\Delta T \tag{5.26}$$

$$\boldsymbol{P}_{k+1|k} = \boldsymbol{A}_k \boldsymbol{P}_{k|k} \boldsymbol{A}_k^\top + \boldsymbol{Q}_k \tag{5.27}$$

with $\boldsymbol{u}_k = \boldsymbol{u}(\tau), \forall \tau \in [k\Delta T, (k+1)\Delta T)$.

The state estimation improves during the measurement update phase based on the measures $\boldsymbol{y}_k$. In our case, the measurement is the set of image point features in the image plane, leading then to a linear measurement equation in the filter state that is already in discrete time:

$$\boldsymbol{\eta}_{k+1} = \boldsymbol{s}_{k+1} + \boldsymbol{W}_{s_{k+1}} = \boldsymbol{C}_{k+1}\boldsymbol{x}_{k+1|k} + \boldsymbol{W}_{s_{k+1}} \tag{5.28}$$

with $\boldsymbol{W}_{s_{k+1}}$ a vector of discrete-time white noise with auto-correlation matrix $\boldsymbol{R}_{k+1} \in \mathbb{R}^{8\times 8}$, while $\boldsymbol{C}_{k+1} = [\mathbb{I}_8 \ \boldsymbol{O}_{8\times 20}]$ is the measurement sensitivity matrix. It then follows the measurement update phase:

$$\boldsymbol{S}_{k+1} = (\boldsymbol{C}_{k+1}\boldsymbol{P}_{k+1|k}\boldsymbol{C}_{k+1}^\top + \boldsymbol{R}_{k+1}) \tag{5.29}$$

$$\boldsymbol{K}_{k+1} = \boldsymbol{P}_{k+1|k}\boldsymbol{C}_{k+1}^\top \boldsymbol{S}_{k+1}^{-1} \tag{5.30}$$

$$\boldsymbol{x}_{k+1|k+1} = \boldsymbol{x}_{k+1|k} + \boldsymbol{K}_{k+1}(\boldsymbol{y}_k - \boldsymbol{C}_{k+1}\boldsymbol{x}_{k+1|k}) \tag{5.31}$$

$$\boldsymbol{P}_{k+1|k+1} = (\mathbb{I} - \boldsymbol{K}_{k+1}\boldsymbol{C}_{k+1})\boldsymbol{P}_{k+1|k} \tag{5.32}$$

5.5.2 EKF for PBVS

The filter state for Pose-Based Visual Servoing (PBVS) is very similar to the previous case but without the depth dynamics. The feature $\boldsymbol{s}$ is a pose error vector whose rotational part is represented with the axis/angle representation and the filter state is now $\boldsymbol{x}(t) = [\boldsymbol{s}(t) \ \dot{\boldsymbol{s}}_o(t) \ \ddot{\boldsymbol{s}}_o(t)]^\top = [\boldsymbol{x_1}(t) \ \boldsymbol{x_2}(t) \ \boldsymbol{x_3}(t) \ ]^\top \in \mathbb{R}^{18}$ with dynamics

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{L_s}(\boldsymbol{x_1})^c\mathbb{T}_e \, {}^e\boldsymbol{J}_e \, \boldsymbol{u} - \boldsymbol{x_2} \\ \boldsymbol{x_3} \\ \boldsymbol{0} \end{bmatrix} + \boldsymbol{w_t}. \tag{5.33}$$

In order to update the state estimate with the process equations, one can follow the same steps as in (5.24)–(5.27).

The "measurements" provided by the camera in the PBVS case is the relative pose of the observed object w.r.t. the camera frame, expressed as the homogeneous transformation matrix ${}^c\boldsymbol{T}_o(\boldsymbol{y}_k)$. This leads to a discrete-time, non-linear and implicit measurement equation w.r.t. both the measure and filter state:

$$\boldsymbol{\eta}_{k+1} = \boldsymbol{h}(\boldsymbol{x}_{k+1|k}, \boldsymbol{y}_{k+1}). \tag{5.34}$$

The measurement equation (5.34) is a constraint equation ($\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) = 0$) and to compute it, we first define the "closure" equation: ${}^{cd}\boldsymbol{T}_c(\boldsymbol{x}_{k+1|k}) \, {}^c\boldsymbol{T}_o(\boldsymbol{y}_{k+1}) \, {}^{cd}\boldsymbol{T}_o^{-1} = \mathbb{I}$, being ${}^{cd}\boldsymbol{T}_o$ the desired pose of the camera with respect to the object (which is of course known by design. See Figure 5.1). The pose vector associated to the left-hand side of the *closure* equation must be zero, and this relationship represents

the measurement equation of the form (5.34). The measurement update equations differ from those in (5.29)–(5.32) as follows:

$$\boldsymbol{H}_{k+1} = \left.\frac{\partial \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_{k+1|k},\boldsymbol{y}=\boldsymbol{y}_{k+1}} \tag{5.35}$$

$$\boldsymbol{D}_{k+1} = \left.\frac{\partial \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{y}}\right|_{\boldsymbol{x}=\boldsymbol{x}_{k+1|k},\boldsymbol{y}=\boldsymbol{y}_{k+1}} \tag{5.36}$$

$$\boldsymbol{S}_{k+1} = (\boldsymbol{H}_{k+1}\boldsymbol{P}_{k+1|k}\boldsymbol{H}_{k+1}^{\top} + \boldsymbol{D}_{k+1}\boldsymbol{R}_{k+1}\boldsymbol{D}_{k+1}^{\top}) \tag{5.37}$$

$$\boldsymbol{K}_{k+1} = -\boldsymbol{P}_{k+1|k}\boldsymbol{H}_{k+1}^{\top}\boldsymbol{S}_{k+1}^{-1} \tag{5.38}$$

$$\boldsymbol{x}_{k+1|k+1} = \boldsymbol{x}_{k+1|k} + \boldsymbol{\eta}_{k+1}\boldsymbol{h}(\boldsymbol{x}_k,\boldsymbol{y}_k) \tag{5.39}$$

$$\boldsymbol{P}_{k+1|k+1} = \boldsymbol{P}_{k+1|k} + \boldsymbol{K}_k\boldsymbol{H}_k\boldsymbol{P}_{k+1|k}. \tag{5.40}$$

Once the filter converges, one can obtain the target velocity from the estimated disturbance state as ${}^{\boldsymbol{c}}\boldsymbol{v}_{\boldsymbol{o}}(t_k) = \boldsymbol{L}_{\boldsymbol{s}}^{-1}\boldsymbol{x}_{\boldsymbol{3}}(t_k)$, as we similarly did using $\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}$ in the depth dynamics of the EKF for IBVS (see second component of (5.23)).

5.6 Simulations

In order to validate the tracking performance of the proposed EKFs and study the feasibility of using second-order visual servoing in a task involving contacts with the environment, we simulate a Peg-in-Hole task with a moving target using FrankaSim [9], a co-simulation environment implemented in ViSP [10] and CoppeliaSim [75]. With FrankaSim it is possible to carry out dynamic simulations with a Panda robot and process different sensor information. This simulator will be further detailed in Chapter 8. The simulation was performed in synchronous mode with a time step of 1 ms using Vortex physics engine, since it is capable to deal with non-convex shapes. A vision sensor was mounted on the robot wrist and we programmatically drop and delay the grabbed frames in order to simulate a 30 fps (33 ms) camera rate with 10 ms of feature extraction delay. The visual features are extracted from an AprilTag of the family 36h11 using ViSP's detector (AprilTag pose or its corners). A 5×5 Gaussian mask is applied on the grabbed images to smooth the AprilTag's borders making the detection process more realistic. This is a convenient way to simulate noise because, despite being easy to add Gaussian noise around the detected pixel of points features, the uncertainties on the estimated pose depend on the distance of the object from the camera [76]. White noise of the magnitude of the real robot was also added to the measured joint velocities $\dot{\boldsymbol{q}}$ with standard deviation $\sigma = 0.021$ rad/s. The robot control loop runs at 1 kHz as for the real one.

5.6.1 Task description and controllers implementation

The simulated task consists in inserting the peg held by the manipulator's gripper into the hole of a work-piece both of cylindrical shape with diameters of 9 and 10 mm respectively. The piece is placed on a moving platform whose lowest joints turn at 0.5 rad/s in opposite directions allowing the upper part to undergo a purely translational motion, while the last joint of the platform performs a sinusoidal velocity motion of amplitude 0.2 rad/s and frequency $1/\pi$ Hz (see Fig. 5.3 and video).

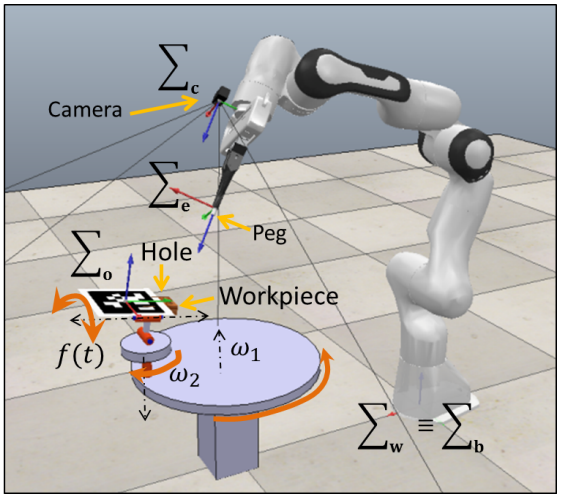


Figure 5.3 – Simulation setup: The Panda robot is in the initial configuration of the experiment. World and base $\Sigma_w \equiv \Sigma_b$, camera Σ_c, end-effector Σ_e and target Σ_o frames are drawn. An AprilTag is attached on the workpiece and tracked by the camera. The lower plate of the platform rotates at $\omega_1 = 0.5$ rad/s, the second plate at $\omega_2 = -\omega_1$ while the last joint motion if $f(t) = 0.2sin(2t)$.

Although the task constrains all the 6 degrees of freedom (DoF) of the operational space, our robot is a redundant manipulator and to prevent the robot joints to move along directions lying in the kernel of the task Jacobian $\boldsymbol{J_s}$, the null-space directions must be damped or, in general, a suitable secondary task must be considered. Controller (5.21) is then modified into a more computationally efficient form with the addition of a damping term in the null space of the task:

$$\begin{aligned} \boldsymbol{u_\tau} = (\boldsymbol{J_s}\boldsymbol{B}(\boldsymbol{q})^{-1})^{\dagger}\Big(\ddot{\boldsymbol{s}}^d + \boldsymbol{D_s}(\dot{\boldsymbol{s}}^d - \boldsymbol{L_s}\,{}^c\mathbb{T}_e\,{}^e\boldsymbol{J}_e\dot{\boldsymbol{q}} - \dot{\boldsymbol{s}}_o) + \\ \boldsymbol{K_s}(\boldsymbol{s}^d - \boldsymbol{s}) + \ddot{\boldsymbol{s}}_o - \boldsymbol{h_q}\Big) + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{P}^{\perp}\boldsymbol{\tau}_N \end{aligned} \tag{5.41}$$

in which the gravity term does not appear anymore since it is already compensated by the robot (in both the real and simulated cases), and $\boldsymbol{P}^{\perp}$ is a Null-space

projector that applies the secondary damping torques $\boldsymbol{\tau}_N = -k_d\dot{\boldsymbol{q}}$ on $\ker(\boldsymbol{J_s}\boldsymbol{B}^{-1})$. Assuming the manipulator is in a non-singular configuration, in the PBVS case the task Jacobian has dimension 6×7 with $rank(\boldsymbol{J_s}\boldsymbol{B}^{-1}) = 6$ and a Null-space projector based on the dynamically consistent Inertia weighted pseudo-inverse $\boldsymbol{P}^{\perp} = (\mathbb{I} - \boldsymbol{J_s}^{\top}\bar{\boldsymbol{J}}_{\boldsymbol{s}}{}^{\top})$ can be computed [11], where $\bar{\boldsymbol{J}}_{\boldsymbol{s}} = \boldsymbol{B}^{-1}\boldsymbol{J_s}^{\top}(\boldsymbol{J_s}\boldsymbol{B}^{-1}\boldsymbol{J_s}^{\top})^{-1}$. On the other hand, in the IBVS case the task Jacobian is 8×7 with $rank(\boldsymbol{J_s}\boldsymbol{B}^{-1}) = 6$ and more general methods must be used to find an orthogonal basis of $\ker(\boldsymbol{J_s}\boldsymbol{B}^{-1})$, *e.g.*, resorting to a Singular Value Decomposition (SVD) [77].

5.6.2 Simulation Results

We ran several simulations for both proposed filters in which the estimates were used in the controller (5.41) to accomplish the task described in Section 5.6.1.

During the first trials we noticed that the target's velocity estimation already showed a good performance but it was not perfect due to the low camera rate and the lack of a motion model of the target. A small tracking error remains, so we added an integral term which is activated when the feature error is sufficiently small in order to compensate for any final error and correctly realize the insertion task. The new controller is then

$$\boldsymbol{e}_{\boldsymbol{s}_{int}} = \begin{cases} \boldsymbol{K_I}\int(\boldsymbol{s}^d - \boldsymbol{s})dt & \text{if} \quad \|\boldsymbol{s}^d - \boldsymbol{s}\| \leq \epsilon \\ \boldsymbol{0} & \text{elsewhere} \end{cases}$$

$$\boldsymbol{u} = (\boldsymbol{J_s}\boldsymbol{B}(\boldsymbol{q})^{-1})^{\dagger}\Big(\ddot{\boldsymbol{s}}^d + \ddot{\boldsymbol{s}}_o - \boldsymbol{h_q} + \boldsymbol{D_s}(\dot{\boldsymbol{s}}^d - \boldsymbol{L_s}\,{}^c\mathbb{T}_e\,{}^e\boldsymbol{J}_e\dot{\boldsymbol{q}} + \dot{\boldsymbol{s}}_o) + \boldsymbol{K_s}(\boldsymbol{s}^d - \boldsymbol{s}) + \boldsymbol{e}_{\boldsymbol{s}_{int}}\Big) + \boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{P}^{\perp}\boldsymbol{\tau}_N + \boldsymbol{u}_{\boldsymbol{\tau}}(0)e^{-\mu t} \tag{5.42}$$

Since for the real robot it is recommended to generate the torque commands starting from zero (recall that gravity is internally compensated) we added a rapidly decaying exponential term, whose initial value coincides with the torque command at time $t = 0$, to guarantee a smooth start of the robot with a torque command that starts from zero ($\boldsymbol{u}(0) = 0$).

The following experiment for the IBVS case was carried out using controller (5.42) with controller gains: $\boldsymbol{K_s} = diag(250)\,\mathrm{s}^{-2}$, $\boldsymbol{D_s} = diag(50)\,\mathrm{s}^{-1}$, $\boldsymbol{K_I} = diag(300)\,\mathrm{s}^{-2}$ and $k_d = 20\,\mathrm{kg.m.s}^{-2}$ and filter parameters $\boldsymbol{R}_{k+1} = diag(6.8\mathrm{e}^{-06})$, $\boldsymbol{Q}_{k+1} = diag(1\mathrm{e}^{-06}\mathbb{I}_8, 1\mathrm{e}^{-06}\mathbb{I}_4, 5\mathrm{e}^{-04}\mathbb{I}_8, 1\mathrm{e}^{-06}\mathbb{I}_8)$. The filter state was then initialized with the position values extracted at the first grabbed image, while

the depths were initialized at 0.2 m and considering the target starts at rest $\boldsymbol{x}(0) = [\boldsymbol{s}(0)\ \boldsymbol{z}(0)\ \dot{\boldsymbol{s}}_o(0)\ \ddot{\boldsymbol{s}}_o(0)]^\top = [s(0)\ 0.2\ 0\ 0]^\top$.

The filter shows an excellent feature tracking and a good reconstruction of the estimated target's velocity (see Figure 5.4) once the depth estimation stabilizes. As we can see from Figure 5.5, at the beginning of the experiment, in which the camera is far and approaches the target, the motion is sufficiently exciting to reconstruct the depth's dynamics leading it to converge towards its true value but then (after 1 s of experiment) when the camera is on top of the target, only the small motion of the last joint of the turntable excites this dynamics. The depth estimates remain almost constant for the rest of the experiment without further converging towards the true values.

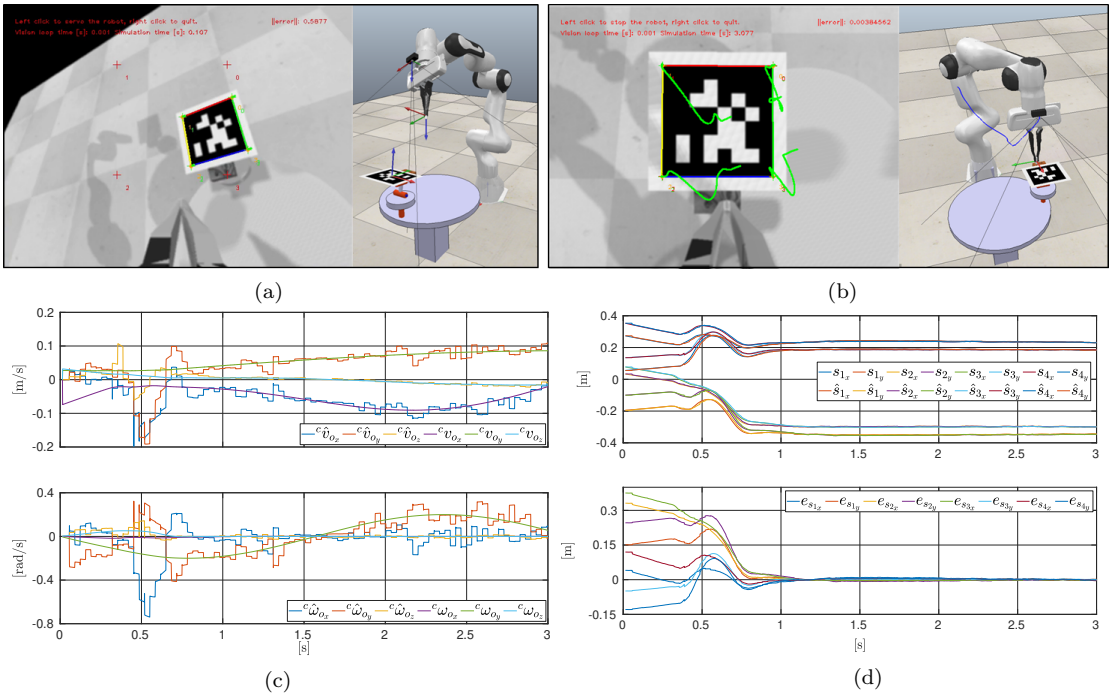


Figure 5.4 – Peg-in-Hole simulation by IBVS: (a): Initial configuration and camera view. (b): Final configuration and camera view; the features paths in the image plane are visible in green while the path followed by the camera is sketched in blue. (c): Target's linear (top) and angular (bottom) velocity estimated vs Ground-Truth. (d): Estimated vs Ground-Truth features' positions (top) and features' error ($\boldsymbol{e_s} = \boldsymbol{s}^d - \boldsymbol{s}$).

While the depth estimation is converging (between 0 and 1 s) the reconstruction of the target velocity is affected causing the controller to badly compensate for this velocity making the feature error to increase. Despite the sensibility of the controller in regards of the feature's depth, it was capable of successfully completing the task - thanks to its ability in reconstructing the feature's depth under the hypothesis of a sufficiently exciting motion - but this is not always the case,

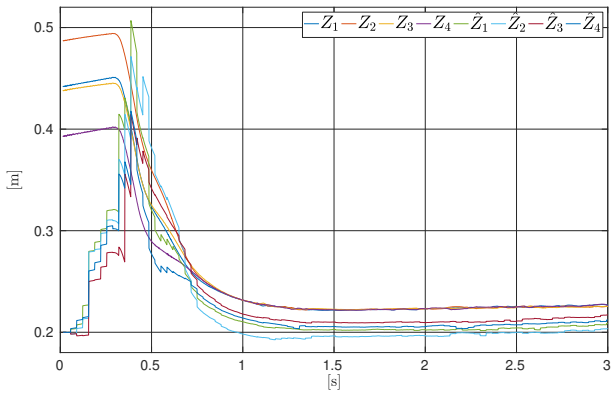


Figure 5.5 – Peg-in-Hole simulation: Feature's depth estimation Vs Ground-Truth. Under sufficiently exciting trajectories, the filter is capable of reconstruct the depth's dynamics. Once the camera lies on top of the target (around 1 s), the motion is not exciting enough to allow the estimates to further converge towards the true values.

and initializing the filter with another depth could lead to failure execution.

To circumvent the persistency of excitation issue, we assume that the forthcoming IBVS experiments will include a "measured" of the features' depth (extracted from the Apriltag's pose). The measurement equations (5.28) are then augmented by the availability of the depth, being now

$$\boldsymbol{R}_{k+1} = \begin{bmatrix} \boldsymbol{R}_s & \boldsymbol{O}_{8\times4} \\ \boldsymbol{O}_{4\times8} & \boldsymbol{R}_z \end{bmatrix}$$

$$\boldsymbol{C}_{k+1} = \begin{bmatrix} \mathbb{I}_8 & \boldsymbol{O}_{8\times4} & \boldsymbol{O}_{8\times16} \\ \boldsymbol{O}_{4\times8} & \mathbb{I}_4 & \boldsymbol{O}_{4\times16} \end{bmatrix}$$

in which $\boldsymbol{R}_z$ is the measurement noise covariance of the features depth.

We then repeated the same experiment for the PBVS and IBVS (with depth measurements) cases using the same controller (5.42), gains, and filter parameters of the previous experiment and their results are reported in Figures 5.6 and 5.7.

The choice of a constant acceleration model for the target motion is well evident from the plotted curves in Figure 5.8; note how the angular velocity about the y axis (${}^{c}w_{o_y}$) follows the target motion about the same direction ($f(t) = 0.2sin(2t)$). In Figure 5.7c (bottom) it is possible to appreciate what looks like a small horizontal shift between the Ground-Truth and the estimated target's pose. This is due to the crossing of the representation boundary at $\varphi_i = \pi$ and not because of the time delay, as one could think; the Kalman filter helps mitigate the delay effects.

With the proposed process models, the filters were able to properly reconstruct the disturbance (target velocity), thus succeeding in completing the task. Apart from the very beginning of the simulation, which is affected by the transitory

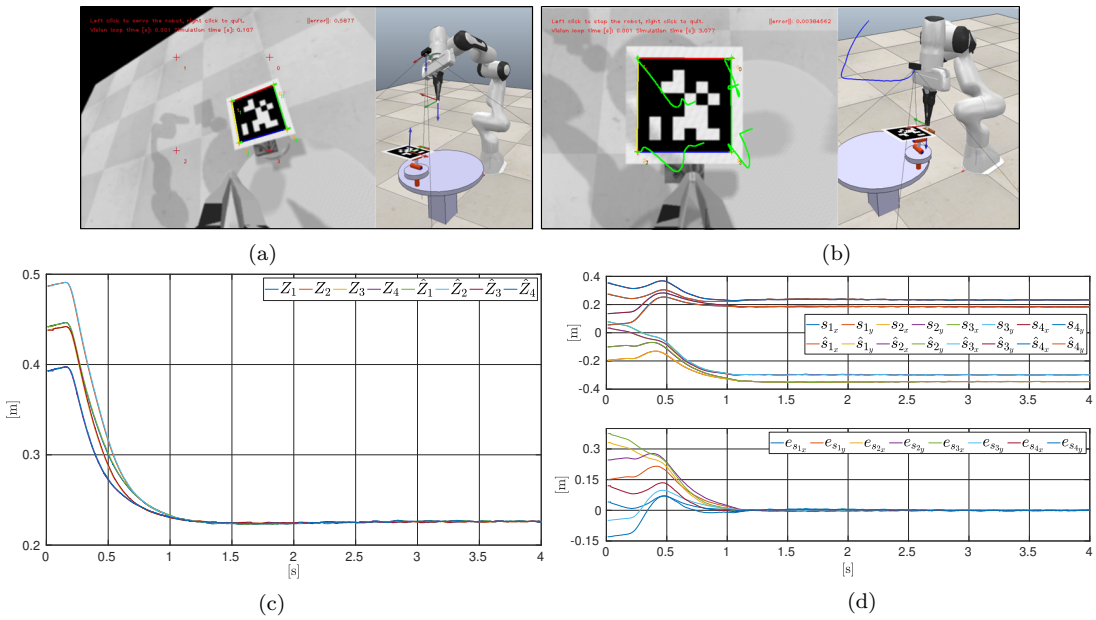


Figure 5.6 – IBVS Peg-in-Hole simulation: Initial (a) and final (b) camera views and robot configurations. It is possible to appreciate, in green, the features paths in the image plane and camera path in blue. (c) Estimated Vs Ground-Truth feature's depth. (d) Estimated Vs Ground-Truth feature's positions (top) and feature's error (bottom).

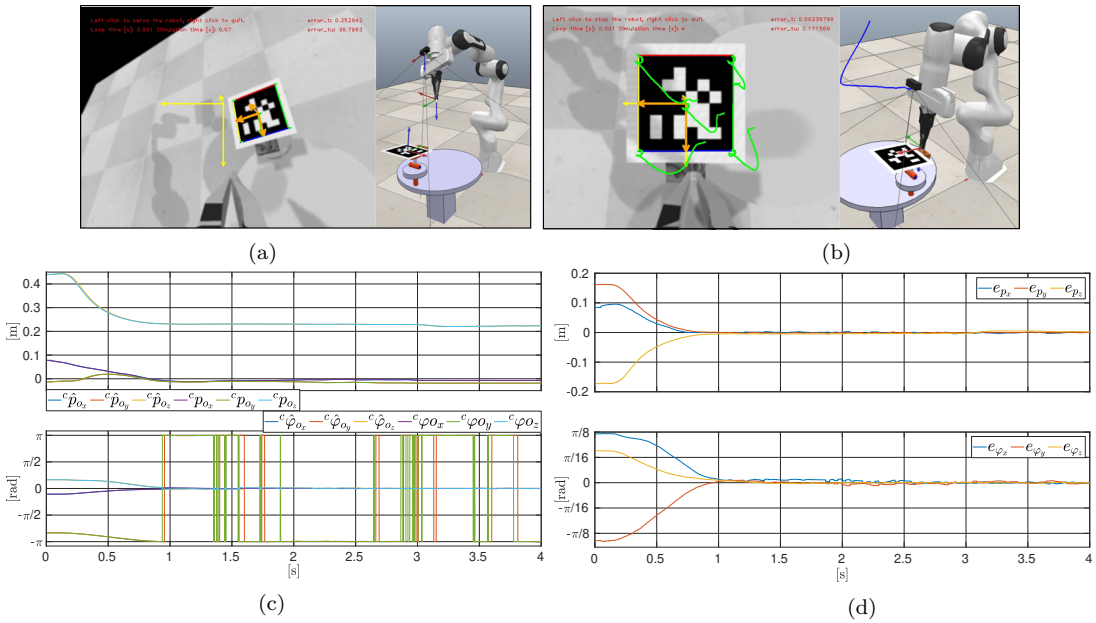


Figure 5.7 – PBVS Peg-in-Hole simulation: Initial (left) and final (right) camera views and robot configurations. Features position (top) and orientation (bottom) errors. On the right image it is possible to appreciate, in green, the trajectory of the target corners in the image plane. Convergence is successfully attained.

of the filter convergence and the effects the "smoothing" exponential ($\mu = 8$), the features trajectories shows a nice exponential decrease. Simulations suggest

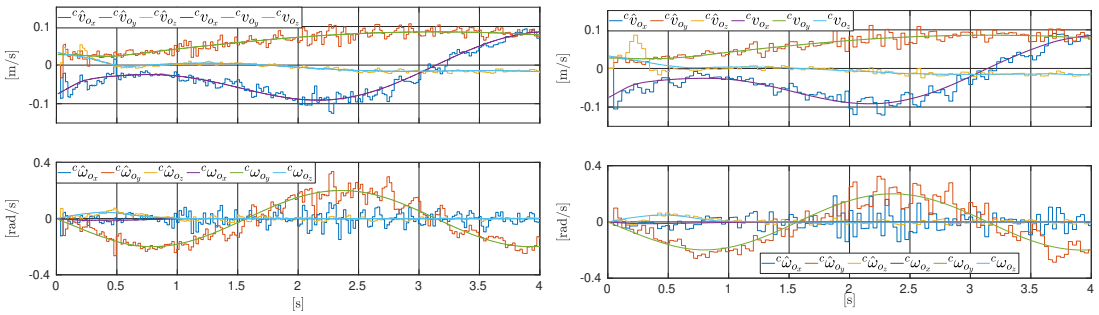


Figure 5.8 – Peg-in-Hole simulations: Estimated target's Linear (top) and angular (bottom) velocity in camera frame Vs Ground-Truth for both the IBVS (a) and PBVS (b) cases.

that second-order visual-servoing controllers can be suitably used to deal with interactions tasks. In the next section we deal with the real implementation of such controllers.

5.7 Real experiments

An experiment for tracking a moving target on a turntable was performed on a Franka Emika Panda robot equipped with a wrist mounted RealSense D435 camera operating at 60 fps. Note that, no Force/Torque sensor is required to implement controller (5.42) . In the following we report the results of the second-order visual servoing for the IBVS case, which is more complex than its PBVS counterpart given both the sensitivity with respect to the feature depths, as we shown before, and the difficulties related to the rank deficiency of the task Jacobian (more details below). The simulations and experiments for both the IBVS and PBVS cases presented in subsection 5.6.2 as well as some interaction experiments using the filters and controllers reported in this chapter can be found in the video reachable through this link https://www.youtube.com/watch?v=fOSRnxd1dlE.

5.7.1 Implementation issues

Most works have presented their results on second-order visual-servoing in simulation and tackled just few of the real world issues like measurement noise [46,47], or instead of generating the torque command from the features acceleration, a velocity signal command is obtained from numerical integration and sent to the robot, as in [78] for a MPC controller. In this work we aimed at pushing forward the state of the art by tackling several implementation aspects of a real implementation not covered in previous works.

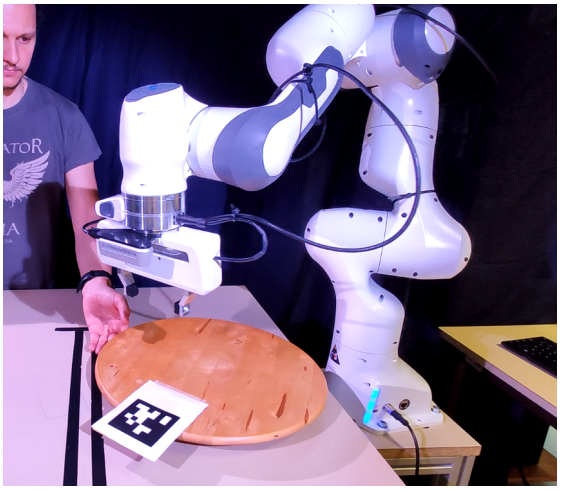


Figure 5.9 – Setup: Panda robot equipped with a RealSense D435 operating at 60 fps. The center of the target is located at 22 cm from the center of the turntable, which is manually operated.

5.7.1.1 Data rate

The main bottleneck of a second-order visual-servoing controller implementation is certainly the low data rate coming from the vision sensor and the latency of the computer vision algorithms for features extraction. It is important to evaluate the performance of the vision system as the whole system controlling the robot. When considering the lowest level servo rate of the robot controller, *i.e.,* the joint controller of the manipulator, it is generally accepted that the servo rate T_s should staisfy the condition:

$$T_s < \frac{T_m}{10} \tag{5.43}$$

where $1/T_m = f_m$ is the mechanical resonance frequency of the manipulator's structure [5] (see [4] for more details). It is therefore necessary that the low level robot's controller operates with a feedback rate of $200 \sim 1000$ Hz, which means a cycle time of $1 \sim 5$ ms since the mechanical resonance frequency of a manipulator usually ranges from $20 \sim 50$ Hz. It is obvious that higher control frequencies are desirable at any case and that conventional visual systems alone are too slow for controlling a robotic system at torque level.

One of the main contributions of this chapter are the proposed EKFs which virtually increase the data rate coming from the camera and helps alleviate the negative effects of the computer vision latency, allowing us to effectively deal with the issue of the servo rate discussed above. If this problem is not addressed, it is practically impossible to implement such a control scheme without resorting to expensive high-speed cameras.

5.7.1.2 Ill-condition of the task Jacobian

The computation of the control command (5.42) requires to pseudo-invert the task Jacobian $\boldsymbol{J} = \boldsymbol{J_s}\boldsymbol{B}^{-1} \in \mathbb{R}^{k\times n}$ which in practice turns out to be strongly ill-conditioned. This problem is due to the double effect of the ill-condition of the inertia matrix [79] which is then combined with the image Jacobian, resulting in a magnified singular values ratio. This issue is particularly pronounced when the disparity between the masses and inertia tensors of the individual links is large. Featherstone [80] has empirically found that the the condition number of the inertia matrix can asymptotically grow with $\mathcal{O}(n^4)$ of the number of bodies, which applies to our case since we are servoing a 7-joints manipulator.

In the simulations, we computed the control command by direct pseudo-inversion of the task Jacobian, through Singular Value Decomposition without observing any issue on the demanded torques, as one can see from the simulations on the video https://www.youtube.com/watch?v=fOSRnxd1dlE (min [0:06 - 0:55]). The task Jacobian is then

$$\boldsymbol{J} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \tag{5.44}$$

with $\boldsymbol{U} \in \mathbb{R}^{k\times k}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{k\times n}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times n}$. $\boldsymbol{\Sigma}$ is a rectangular diagonal matrix whose diagonal elements are the singular values. $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices. For the real robot, using the same control gains as in simulation, demanded torques were unrealistic and incompatible with the joint torque limits, leading the robot to abruptly stop due to the high command values requested. Lowering down the control gains to obtain command values compatible with the joint torque limits, we were unable to properly servo the robot. We then realized that the task Jacobian was highly ill-conditioned.

A common approach in robotics to deal with ill-conditioned problems is to perform a regularization, which allows for a numerically stable solution. Damped Least Squares (DLS), also known as Levenberg-Marquardt method, is one of the most common regularization techniques for solving the inverse kinematic problem in the form

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$
$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|^2 + \boldsymbol{R}\|\boldsymbol{x}\|^2$$

where $\boldsymbol{R} = diag(\lambda, \ldots, \lambda)$ is a regularization diagonal matrix. In this case, finding the solution to the problem is a compromise between preserving the control requirements and maintaining the solution limited [77]. With this solution, the singular values of the pseudo-inverse change from $\frac{1}{\epsilon_i}$ to $\frac{\epsilon_i}{\epsilon_i+\lambda}$ (with ϵ_i the i-th singular value of the non-inverted matrix). However, we found that with a DLS-based

regularization, rotations about x- and y-axis were far less "stiff" than the other directions, leading to poor orientation tracking performance.

In order to adjust separately for each singular value the given damping factor, we used a SVD-based regularization with a Gaussian function of the singular value as regularization term, $g_i(\epsilon_i) = m \exp(-\frac{\epsilon_i^2}{2\sigma^2})$, where m is the height of the curve's peak and σ controls the width of the "bell". We set then $\boldsymbol{P} = diag(g_1(\epsilon_i), \ldots, g_m(\epsilon_m))$, and compute the regularized solution as

$$\boldsymbol{x} = \boldsymbol{V}(\boldsymbol{\Sigma}^\top \boldsymbol{\Sigma} + \boldsymbol{P})^\dagger \boldsymbol{\Sigma}^\top \boldsymbol{U}^\top \boldsymbol{b}.$$

5.7.1.3 Joints friction

In [8], the dynamic friction for our robot has been identified and we used the released library to compensate for it adding this torque contribution in controller (5.42). Unfortunately we do not have a good model for the dry/static friction, and as a result small torques are actuating the robot, limiting the positioning performance close to the goal position; we have found this threshold to be ~ 0.5 Nm for our robot. This phenomenon particularly affects the last joint since the controlled torques are often lower than this value. To alleviate this issue as already mentioned in Section 5.6.2, an integral term on the features error was added, allowing to reduce the remaining steady state error. Further increasing the proportional gain could in fact lead to an unstable behaviour.

5.7.1.4 Illumination conditions

Despite evident when working with vision sensors, experiments are often conducted under poor illumination conditions with consequent performance degradation. In our case, it is of paramount importance to guarantee and maintain a continuous an constant image stream to avoid that the data fusion of delayed visual measurements leads to catastrophic results. A good illuminated environment stabilizes the feature extraction with ViSP to ~ 8 ms while in regular conditions it can oscillate from 8 to 20 ms. Spotlights were then placed around the experimental platform also trying to avoid obscuring the target or blinding the camera with reflections.

5.7.2 Results

The proposed EKF proves to be effective both in increasing the data rate of the camera, providing an accurate estimate of the features (and depth) at higher

frequency, and in alleviating the effects of delays in the feature extraction process as well as in providing an accurate estimate of the target velocity. In Fig. 5.10 we can see how the feature's errors remains limited despite the fast motion (up to 40 [cm/s] for the linear and almost 2 [rad/s] for the angular velocity of the target) manually applied to the turntable. Once the target is kept at rest ($t > 40$ s), the feature errors slowly converge to zero. This is due to the high dry friction in the joints that the integral term has to overcome. Dry friction remains the main issue limiting the tracking accuracy of our implementation, while SVD-based regularization managed to cope better with the strong ill-conditioning of the task Jacobian than Damped Least Squares.

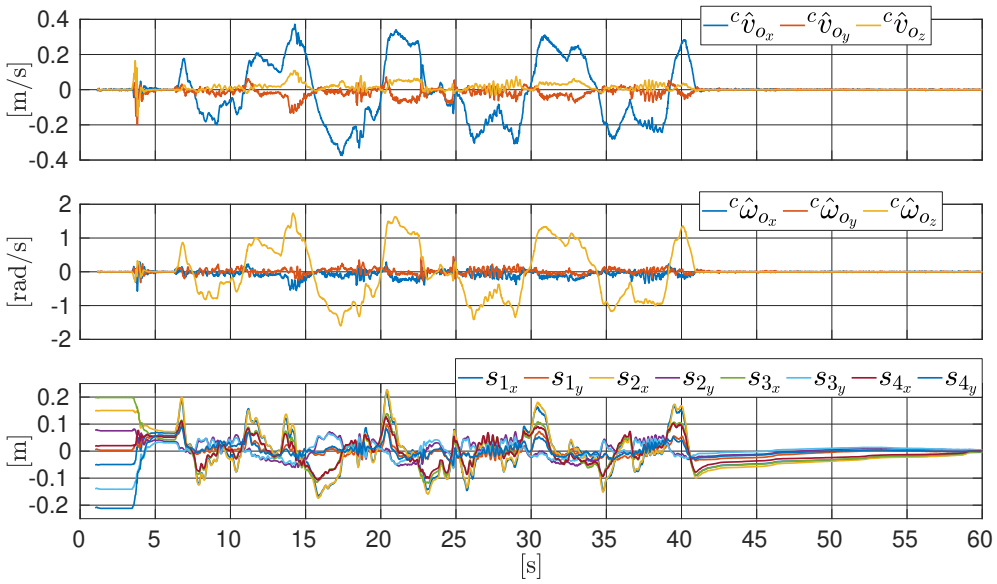


Figure 5.10 – Estimated target linear (top) and angular (middle) velocities in camera frame compared to the feature errors (bottom). The circular motion produces a linear velocity component tangential to the circle and from a frame on top of the target it produces a linear and an angular velocity in camera frame along the x-axis and about the z-axis respectively.

In the video https://www.youtube.com/watch?v=fOSRnxd1dlE, some further interaction experiments with different gains are carried out (min [2:12 - 2:23]). It is possible to appreciate (left bottom window) how the robot does some "jerky" motions. This happens when joint torque limits are violated. The ViSP wrapper of the *libfranka* that manages the control loop is implemented in such a way that the execution is recovered (a maximum of 10 times) after a violation occurs (*e.g.*, by crossing joint torque limits, communication error due to packet loss, and so on). A possible solution could be to saturate the commanded joint torques at the expense of the tracking performance when the saturation is crossed.

The last part of the same video (min [2:23 - 3:00]), shows the comparison of two experiments with a slightly different inertial configuration of the robot since a wrist-mounted Force/Torque sensor with its aluminium flanges ($\sim$ 0.6 kg) is present in one of the experiments (right window — note that the sensor is not used). The presence of this "payload" redistributes the inertias in the kinematic chain improving the condition number of the inertia matrix of the robot, causing a smoother motion. This is also confirmed by the simulator since for the same configuration of the robot, the one in figure 6.5, the condition number of the inertia matrix of the real robot is $\sim$ 800, while for the simulated robot it is $\sim$ 300. This is due to the identified parameters retrieval process, reported in Chapter 7, that tends to uniformly distribute the solution among the links of the manipulator.

5.8 Summary

In this chapter, we presented our results on Second-Order visual servoing for interaction control and moving targets. The proposed EKFs effectively tracks the target motion and provides high rate visual information allowing for the implementation on a real platform with very good performance for the motion tracking. Several real-world implementation issues have been discussed alongside the solutions adopted. We recognize in the dry friction the main cause limiting the tracking performance of our implementation, especially at low speeds.

The derivations and results shown in this chapter are the result of the works issued on the papers:

- **A. A. Oliva**, P. Robuffo Giordano, and F. Chaumette, "A general visual-impedance framework for effectively combining vision and force sensing in feature space," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp.4441–4448, 2021.
  Video: https://www.youtube.com/watch?v=Qw0Dnhq2uRQ

- **A. A. Oliva**, E. Aertbeliën, J. de Schutter, P. Robuffo Giordano, and F. Chaumette, "Towards dynamic visual servoing for interaction control and moving targets," ICRA 2022 - *IEEE International Conference on Robotics and Automation*, May 2022, Philadelphia, United States. (to appear)
  Video: https://www.youtube.com/watch?v=fOSRnxd1dlE

Chapter

6 Feature Space Admittance

Contents

6.1 Introduction

As it was shown in the previous chapter, the feature impedance control cannot be implemented as is with classical cameras due to their intrinsic technological limitations, *e.g.,* slow frame rate compared to the faster force/torque sensing, which makes it impossible to control a manipulator at torque level. The low control bandwidth resulting from the use of low-rate cameras is in contrast with the need of high gains in the impedance controller for ensuring good tracking performances and disturbance rejection. It follows that there exists a trade-off between the trajectory tracking accuracy and the compliance given to the robot. We have successfully overcome these limitations with the proposed Extended Kalman Filters and by tackling the further real-world difficulties that have arisen. On the other hand, when an interaction task can/must be performed in static or quasi-static conditions, the dynamics of the manipulator can be neglected and one can use the joint level manipulator's velocity controller; in this way, the manipulator can be seen as an ideal positioning device. The extra complexity needed to let the impedance controller properly operate can be avoided using, for example, an admittance law in feature space coupled with a velocity resolved controller, which is the subject of this chapter. In this case, thanks to the force/torque measurements provided by a 6D Force/Torque sensor, we will be able to deform the visual trajectory reference given to the robot according to the measured forces, being able to provide compliance.

6.2 Related works

The hybrid vision/force scheme presented in Section 4.2 aims at controlling force along constrained directions while vision controls the motion of the remaining ones. The task geometry needs to be known *a priori* in order to properly design the controller via a *selection* matrix for ensuring orthogonality between vision and force controlled directions. Despite the use of feedforward terms to share the controlled directions among vision and force, the intrinsic sensory separation enforced by the underlying hybrid scheme, does not fully exploit the complementarity of vision and force sensing.

As we have shown in Section 4.3, a position-based impedance controller was used to achieve compliance in Cartesian space while an external vision control loop is closed around the former [55]. The main disadvantage of this control scheme is that it can get stuck in a local minimum where the simultaneous convergence towards the force and visual reference will not be reached. Recently,

three image-based visual impedance control laws have been proposed [81]. Although the presented first- and second-order controllers rely on the regulation of the visual error, compliance of the end-effector carrying the object is achieved along/about its Cartesian directions.

Mezouar *et al.* [57] developed the External/Hybrid vision/force control scheme detailed in Section 4.4, to overcome the drawbacks of the Hybrid and Impedance-based vision-force schemes, for which they have provided an exhaustive comparative analysis. In their approach, the external *wrench* is transformed into a displacement of the image feature reference. This transformation is equivalent to an undamped spring which, as we show in Section 6.6, can start oscillating without ever reaching the convergence of the task or, in more serious cases, can damage the tool. We then seek for a higher order relation linking forces and features motion. One of the first works aiming at figuring out this relationship for vision driven robotic systems is [70], in which the visual servoing dynamics for a ball target is derived and, by exploiting a general definition and pose invariance of the Lagrangian function in the feature space, authors yields to an *ad hoc* simplified model dynamics for the features and the system considered. Vice versa, we are interested in the full Lagrangian model of the manipulator and in a generalized treatment that is independent of the type of visual feature.

Carelli *et al.* [82] proposed a Hybrid force- and vision-based impedance controller to perform a *peg-in-hole* task. Pose-based visual servoing was used to guide the end-effector towards the hole. Interaction forces were fed back to correct small errors of visual guidance by modifying the visual reference along the horizontal plane while pure force control was used along the vertical axis. The use of selection matrices does not allow to use vision along the vertical axis. The interaction wrench were fed back simply by changing the point of application through a coordinate transformation. The Authors do not provide a mapping to project physical forces into the feature space, so they could consider fictitious forces synthetically generated in the image plane or physical forces in operational space, limiting the range of applicability.

In the work presented in this chapter, we instead leverage on the derivation of second order visual servoing shown in Section 3.4 that does not depend on the particular choice of visual features: this allows us to derive a general framework that can effectively combine vision and force sensing directly in feature space. This differs from previous works on this topic since the derivation of second-order models has often been formulated *ad hoc* from the definition of the considered features. Furthermore, thanks to this formulation, the projection of the *wrench* applied on the camera into the feature space can also be generalized. This led us

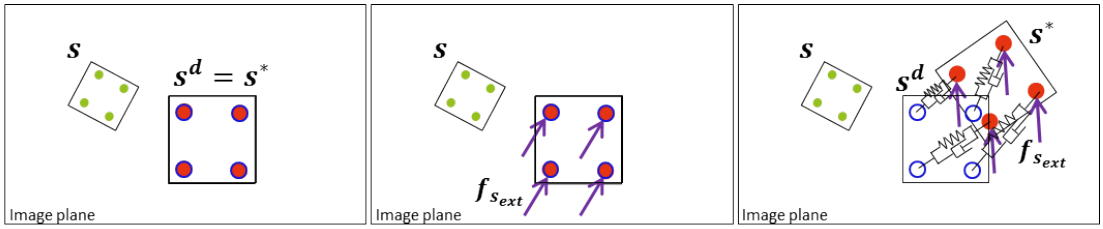
to use a feature admittance law coupled with a force control law to ensure precise force regulation.

6.3 Feature Space Admittance

In Chapter 5 it was shown how the external *wrench* projects into the feature space as virtual per unit of mass/inertia force/torque that acts on the visual features. Exploiting this relationship and the concept of compliant frame, shown in section 2.4.3, we can link the desired reference and the compliant one with a second-order relationship that will be used to feed a vision control loop.

For the PBVS case, the admittance is equivalent to the one defined in the operational space, as reported in [13], and depends on the chosen orientation representation (e.g., Euler angles, axis/angle, etc). In this work we have adopted the axis/angle representation for the 3D features. In our case, on the other side, the task frame is the reference frame attached to the camera rather than to the end-effector's frame (see Figure 6.5).

For the IBVS case, the concept of compliant frame can be extended to the image space taking into account the vector of the current features $\boldsymbol{s}$, the vector of the desired features $\boldsymbol{s}^d$ and defining a new vector of compliant features $\boldsymbol{s}^*$, as shown in Figure 6.1. As for the PBVS case, the compliant feature vector is



(a) (b) (c)

Figure 6.1 – The Compliant frame concept extended to the feature space: (a) Desired and Compliant features are coincident, the current features follow the compliant features. (b) The sensor undergoes the action of an external force. (c) The compliant features are linked to the desired ones with a *mass-spring-damper* relationship.

coincident with the desired one until it moves under the action of an external per unit of mass/inertia force/torque $\boldsymbol{f}_{s_{ext}}$. A controller will minimize the error between current and compliant feature vectors $\boldsymbol{e}_s = (\boldsymbol{s}^* - \boldsymbol{s})$ while the error $\bar{\boldsymbol{e}}_s = (\boldsymbol{s}^d - \boldsymbol{s}^*)$ between desired and compliant vectors will be linked through the second order relationship:

$$\boldsymbol{M}_s(\ddot{\boldsymbol{s}}^d - \ddot{\boldsymbol{s}}^*) + \boldsymbol{D}_s(\dot{\boldsymbol{s}}^d - \dot{\boldsymbol{s}}^*) + \boldsymbol{K}_s(\boldsymbol{s}^d - \boldsymbol{s}^*) = \bar{\boldsymbol{f}}_{s_{ext}} \tag{6.1}$$

with $\bar{\boldsymbol{f}}_{s_{ext}}$ the external *wrench* projected in feature space with the constant inertia as in (5.17). The difference between (5.17) and (6.1) are the features involved. In the former, the current $\boldsymbol{s}$ and the desired $\boldsymbol{s}^d$ features are considered while in the latter, are the desired $\boldsymbol{s}^d$ and the compliant $\boldsymbol{s}^*$ feature vectors to be related. When no external forces act on the end-effector, the compliant and desired reference frames are coincident ($\Sigma_* \equiv \Sigma_d$) or ($\boldsymbol{s}^* = \boldsymbol{s}^d$).

The feature reference position $\boldsymbol{s}^*$, as well as its velocity $\dot{\boldsymbol{s}}^*$ are obtained by time integration of the impedance equation (6.1) by isolating $\ddot{\boldsymbol{s}}^*$ and then integrating it twice:

$$\ddot{\boldsymbol{s}}^* = \ddot{\boldsymbol{s}}^d + \boldsymbol{M}_{\boldsymbol{s}}^{-1}(\boldsymbol{D}_{\boldsymbol{s}}(\dot{\boldsymbol{s}}^d - \dot{\boldsymbol{s}}^*) + \boldsymbol{K}_{\boldsymbol{s}}(\boldsymbol{s}^d - \boldsymbol{s}^*) - \bar{\boldsymbol{f}}_{s_{ext}}) \tag{6.2a}$$

$$\dot{\boldsymbol{s}}^* = \int \ddot{\boldsymbol{s}}^* \, dt \tag{6.2b}$$

$$\boldsymbol{s}^* = \int \dot{\boldsymbol{s}}^* \, dt \tag{6.2c}$$

It is interesting to notice that depending on both the type of visual feature chosen and the values assigned to the impedance parameters for each feature, we can have a different compliant behavior and this can be exploited to our advantage. If for example we have four feature lines expressed in the polar coordinates, as shown in Section 3.3.1 , one can opportunely tune the relative in per unit of mass/inertia stiffness of the impedance law and achieve the desired behaviour. Referring to Figure 6.2, the relative stiffness matrix was chosen in such a way we

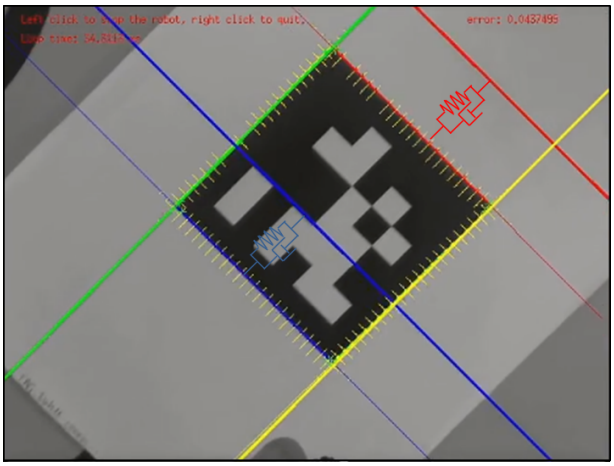


Figure 6.2 – The four sides of the AprilTag are used as image line features. The reference lines (thicker lines) experience a spring-like displacement according to the chosen impedance parameters and the sensed forces.

are stiff with respect to linear displacements of the yellow and green lines as well as for the rotations of all four lines while we are more compliant with respect to linear displacements of the blue and red lines, allowing us to have the yellow and

green lines as a rail on which we can slide. The same reasoning applies to any other visual feature defined in the image plane.

6.4 The Extended External Hybrid Vision-Force Control Scheme

From what explained in the previous section, we are modifying the visual reference according to the sensed force; therefore, we are not facing a regulation problem anymore but a trajectory tracking one. To this end, an appropriate visual control law is necessary in order to better track the moving visual reference [45, 48], as we show in Section 3.3.5. In this case we have to track the trajectory $\dot{\boldsymbol{s}}^d(t)$, while $\dot{\boldsymbol{s}}(t) = \boldsymbol{L_s}\,{}^c\boldsymbol{v_c}$ because we are considering the target to be static ${}^c\boldsymbol{v_o} = 0$. Imposing as usual an exponential decoupled decrease of the task error as control objective $\dot{\boldsymbol{e}}_{\boldsymbol{s}} = -\lambda \boldsymbol{e_s}$, being $\dot{\boldsymbol{e}}_{\boldsymbol{s}} = \dot{\boldsymbol{s}}^d(t) - \dot{\boldsymbol{s}}(t) = \dot{\boldsymbol{s}}^d(t) - \boldsymbol{L_s}\,{}^c\boldsymbol{v_c}$, we obtain

$$ {}^c\boldsymbol{v_c} = \lambda \boldsymbol{L}_{\boldsymbol{s}}^{\dagger} \boldsymbol{e_s} + \boldsymbol{L}_{\boldsymbol{s}}^{\dagger} \dot{\boldsymbol{s}}^* \tag{6.3} $$

In the more general case in which also the target moves, an approximation of the error rate of change can be computed as $\widehat{\dot{\boldsymbol{e}}_{\boldsymbol{s}}} = \frac{d\boldsymbol{e_s}}{dt}$ which contains ideally $\dot{\boldsymbol{e}}_{\boldsymbol{s}} = \dot{\boldsymbol{s}}^d(t) - \boldsymbol{L_s}\,{}^c\boldsymbol{v_c} + \dot{\boldsymbol{s}}_o$, i.e., the contribution to the error variation due to the changes in the trajectory reference and the camera and target motion. Subtracting then the feature velocity due to the camera motion, we can obtain an approximation of both the trajectory changes and the target motion

$$ {}^c\boldsymbol{v_c} = \lambda \boldsymbol{L}_{\boldsymbol{s}}^{\dagger} \boldsymbol{e_s} + \boldsymbol{L}_{\boldsymbol{s}}^{\dagger} (\widehat{\dot{\boldsymbol{e}}_{\boldsymbol{s}}} + \boldsymbol{J_s} \dot{\boldsymbol{q}}) \tag{6.4} $$

otherwise, we can of course use the results obtained in the previous chapter for the tracking of a moving target and directly compensating for its motion, finally obtaining

$$ {}^c\boldsymbol{v_c} = \lambda \boldsymbol{L}_{\boldsymbol{s}}^{\dagger} \boldsymbol{e_s} + \boldsymbol{L}_{\boldsymbol{s}}^{\dagger} \dot{\boldsymbol{s}}^* + \boldsymbol{L}_{\boldsymbol{s}}^{\dagger} \dot{\boldsymbol{s}}_o \tag{6.5} $$

Combining this classical visual servoing control law with the feature space admittance defined in the previous section we can build an interaction control scheme as follows:

It is worth noticing that, since we are moving the compliant features $\boldsymbol{s}^*$ around the image plane, we should use the interaction matrix $\boldsymbol{L_s}(\boldsymbol{s}^*)$ associated to those features to project the external wrench into the feature space while for the vision control law (VCL) (6.3) we use the interaction matrix associated to the current

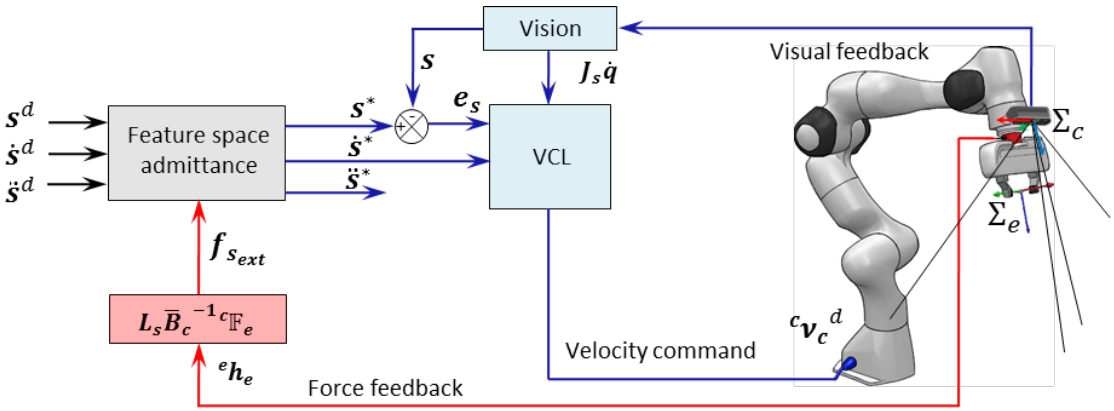


Figure 6.3 – Basic control scheme: An outer admittance control loop is closed around an inner vision control loop. The feature admittance enables the classical visual-servoing controller to accommodate to interaction forces with the environment. Although $\ddot{s}^*$ is available, it is not used in the VCL. The external loop is quite general and can be plugged in many vision control schemes.

features $\boldsymbol{L_s}(\boldsymbol{s})$. When the controller tracks the compliant feature vector closely, we could use the interaction matrix associated to the current features to approximate the projection into feature space of the external wrench.

6.4.1 Force regulation

For many robotic applications the regulation of the exchanged forces with the environment is of particular importance for the correct execution of the task itself. It should be stressed that, since our controller achieves force control by regulating the velocity setpoint, the amount of exerted force is a consequence of the positional error and the chosen impedance stiffness. In particular, for the visual-feature admittance law in Figure 6.3, at the equilibrium (*i.e.*, $\ddot{\bar{\boldsymbol{e}}}_{\boldsymbol{s}} = \dot{\bar{\boldsymbol{e}}}_{\boldsymbol{s}} = 0$) and referring directly to the external *wrench* in the end-effector frame ${}^e\boldsymbol{h}_e$, we have:

$$\boldsymbol{K_s}\bar{\boldsymbol{e}}_{\boldsymbol{s}} = \boldsymbol{L_s}(\boldsymbol{s}^*)\bar{\boldsymbol{B}}_c{}^{-1}\,{}^c\mathbb{F}_e\,{}^e\boldsymbol{h}_e \tag{6.6}$$

Our intention is to regulate the amount of exchanged force along/about a certain direction to a specific value. To this purpose we use, together with the visual-feature admittance, a PI controller based on the force error in operational space (where the *wrench* is naturally defined) projected into the feature space. We can replace then the second member of (6.1) with the following force control law (FCL):

$$\begin{aligned} \bar{\boldsymbol{f}}^*_{s_{ext}} &= \boldsymbol{L_s}(\boldsymbol{s}^*)\,\bar{\boldsymbol{B}}_c{}^{-1}\,{}^c\mathbb{F}_e\,{}^e\boldsymbol{h}^*_e \;\text{ with} \\ {}^e\boldsymbol{h}^*_e &= \boldsymbol{K}_{\boldsymbol{f_P}}({}^e\boldsymbol{h}^d_e - {}^e\boldsymbol{h}_e) + \boldsymbol{K}_{\boldsymbol{f_I}}\int({}^e\boldsymbol{h}^d_e - {}^e\boldsymbol{h}_e)\cdot dt \end{aligned} \tag{6.7}$$

being ${}^e\boldsymbol{h}_e^d$ the desired *wrench* in the end-effector frame, $\boldsymbol{K_{f_P}}$ and $\boldsymbol{K_{f_I}}$ two 6×6 positive gain matrices for the proportional and integral force error terms respectively. The presence of the integral term in the controller creates a dominance of the force loop over the internal positional loop. It is of course clear that such a desired force/torque to be regulated has to be specified according to the task geometry (along/about a constrained direction) otherwise velocities will arise along/about unconstrained directions.

Once all the parts of the framework have been explained, it is possible to put them all together. Figure 6.4 shows the block diagram of the proposed controller: it presents a hierarchical structure in which a FCL, together with a feature space admittance law, is closed around an inner vision control loop implementing the VCL in (6.3). The reference trajectory $\boldsymbol{s}^d(t)$ is opportunely modified by the

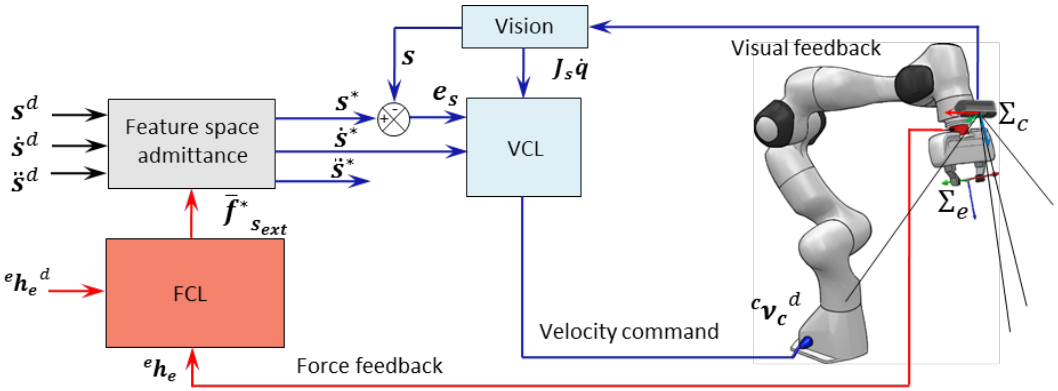


Figure 6.4 – Extended external hybrid vision/force control scheme. A feature space admittance is used as reference modifier to achieve compliance along visual features directions. Force control law ensures force regulation when the integral term is activated.

admittance (6.1) proportionally to the output of the FCL implementing (6.7). We named the resulting controller as the *Extended External Hybrid* vision-force control scheme since it presents the same structure of the External Hybrid (see Section 4.4). The main advantage over the latter is that our framework offers more flexibility in the choice of the compliant behavior we want to implement, thanks to a greater number of parameters in the admittance that can be tuned, besides giving the possibility to achieve force regulation. It will be shown in a comparative experiment in Section 6.6.3 how this framework outperforms the External Hybrid when a soft contact behaviour is required.

6.5 Stability analysis

As already pointed out, the proposed control scheme presents a hierarchical struc-

ture where an external force control loop is closed around an internal vision loop. The external force loop is used to modify the reference trajectory from its planned path, according to the sensed forces/torques and with a behaviour determined by the impedance parameters. In order to study its stability, let us start from the simplest case and gradually consider all possible cases.

If we assume that the manipulator moves in free space towards a motionless target regulated by a constant desired task position, we are in the conditions of a classical visual servoing control scheme. In this case, the stability conditions are those given in Subsection 3.3.3.

If we now consider the more general case of trajectory and target tracking, for which controller equation (6.5) is replaced in the time variation of the error $\dot{\boldsymbol{e}}_{\boldsymbol{s}} = \dot{\boldsymbol{s}}^d - \dot{\boldsymbol{s}} = \dot{\boldsymbol{s}}^d - \boldsymbol{L}_{\boldsymbol{s}}^{\boldsymbol{c}}\boldsymbol{v}_{\boldsymbol{c}} + \dot{\boldsymbol{s}}_o$, we obtain

$$\begin{aligned} \dot{\boldsymbol{e}}_{\boldsymbol{s}} &= \dot{\boldsymbol{s}}^d - \boldsymbol{L}_{\boldsymbol{s}}(\lambda\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}\boldsymbol{e}_{\boldsymbol{s}} + \widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}\dot{\boldsymbol{s}}^* + \widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}\widehat{\dot{\boldsymbol{s}}_o}) + \dot{\boldsymbol{s}}_o = \\ &= \dot{\boldsymbol{s}}^d - \lambda\boldsymbol{L}_{\boldsymbol{s}}\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}\boldsymbol{e}_{\boldsymbol{s}} - \boldsymbol{L}_{\boldsymbol{s}}\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}\dot{\boldsymbol{s}}^* - \boldsymbol{L}_{\boldsymbol{s}}\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}\widehat{\dot{\boldsymbol{s}}_o} + \dot{\boldsymbol{s}}_o \end{aligned} \tag{6.8}$$

Even if $\boldsymbol{L}_{\boldsymbol{s}}\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}} > 0$, the error will converge to zero only if the estimation of the features velocities due to the target motion $\widehat{\dot{\boldsymbol{s}}_o}$ is sufficiently accurate so that

$$\boldsymbol{L}_{\boldsymbol{s}}\widehat{\boldsymbol{L}_{\boldsymbol{s}}^{\dagger}}\widehat{\dot{\boldsymbol{s}}_o} = \dot{\boldsymbol{s}}_o \tag{6.9}$$

while for the trajectory tracking term $\dot{\boldsymbol{s}}^*$, the trajectory to follow is usually known ad can be fully compensated.

6.6 Experiments

In this section, we show the results of our proposed *Extended External Hybrid* controller executing a *Peg-in-Hole* task. We then compare it to the External Hybrid controller [57] in a critical situation. For a better understanding of this work we suggest to refer to the relative video https://www.youtube.com/watch?v=Qw0Dnhq2uRQ in which, in addition to the experiments here reported, we show a trial using lines as visual features and another trial that simultaneously copes with physical and fictitious forces to maintain the visual references in the field of view.

6.6.1 Experimental Setup

The setup consists of a Panda robot with its control software running on a Desktop PC with an Arch-Linux distribution patched with a *Preemptive* RT-Kernel 5.4.52-rt31. A wrist mounted six axis force/torque sensor Alberobotics FT45 with force

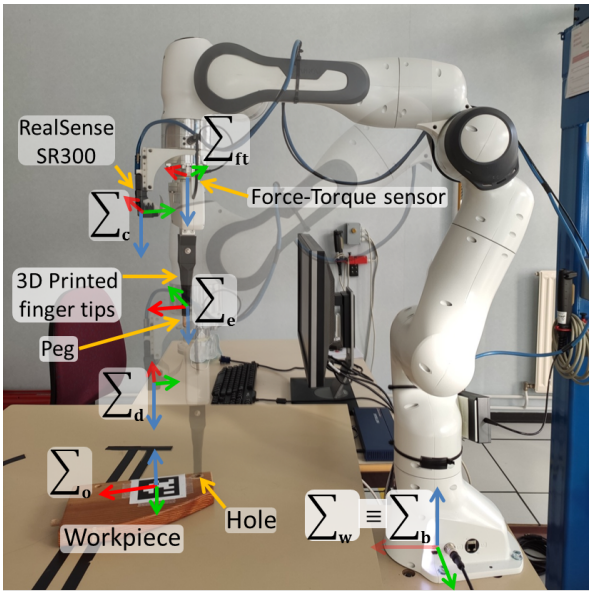


Figure 6.5 – System setup and reference frames.

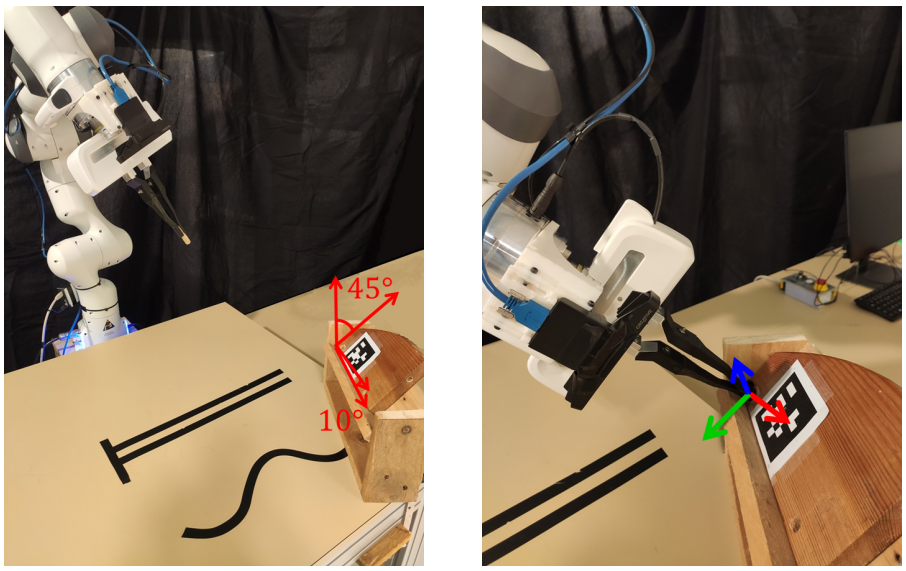
and torque ranges of $F_z \pm 1000N, F_x, F_y \pm 500N$ and $M_{x,y,y} \pm 20Nm$ respectively is present as well as a RealSense SR300 camera operating at 30 fps, mounted in a *eye-in-hand* configuration (see Figure 6.5).

Force/torque sensors are capable of measuring any kind of force, being it gravitational, inertial or contact forces. We are interested only in the contact forces. Since force measurements modify the visual reference, gravitational and inertial effects should be compensated. If the payload is not well estimated, spurious force/torque readings remain that cannot be compensated for, without resorting to online estimation strategies of the payload parameters. These spurious readings will be interpreted as forces/torques acting on the features reference and will move them according to the admittance stiffness.

To make the system more robust against misestimation of the payload parameters (*i.e.*, mass, inertia, and center of mass) or inertial effects, we can choose higher values of the virtual per unit of mass/inertia stiffness of the admittance at the expense of a more rigid system. In our setup, in addition to the standard gripper, we have the camera, the force/torque sensor and some 3D printed parts whose weight has to be estimated. An efficient way to estimate these parameters is reported in [83] that relies on the identified coefficients of the robot dynamics that can be found in [8] for our robot and will be the subject of Chapter 7. As we use a wrist-mounted force/torque sensor, we need to compensate only for the payload downstream the sensors flange. In all the reported experiments, a 3^{rd}-order Butterworth low-pass filters with *cut-off* frequency of $2\,Hz$ is used to smooth the sensor readings of the Force/Torque sensor.

6.6.2 Peg-in-Hole Experiment

The task consists in inserting a *peg* into a *hole* present in a wood workpiece, both with a diameter of 1 cm. The depth of the *hole* is 2 cm while the length of the *peg* is 4 cm. An AprilTag of the 36h11 family of size 6.45 cm is applied to the surface of the workpiece. The latter is supported by a wooden structure fixed to the work table, which holds an inclined groove of about 45° with respect to the horizontal plane on one side and about 10° on the other (see Figure 6.6(a)).



(a) (b)

Figure 6.6 – Peg-in-hole experiment setup: (a) The robot at the initial configuration. (b) Forces exerted by the end-effector on the environment during phase 2: the task requires to apply $[5, -5, -15]N$ along the end-effector's x-(blue), y-(green) and z-axis (red arrow) respectively.

The experiment consists of two phases. In the former the robot has to insert the *peg* in the *hole* without computing any trajectory while in the latter it pushes the *peg* towards the bottom of the *hole* and simultaneously applies lateral forces such that the workpiece slides into the groove until a desired force is reached along both directions. The task is executed using our proposed Extended Exteral Hybrid controller (see Figure 6.3) for which the VCL gain is set to $\lambda = 1.5\, s^{-1}$. The vision system guides the camera towards a desired final position of the features in the image plane. This final position corresponds to the position in which the peg, held by the robot gripper, is inserted in the hole at a depth of about $1.5\, cm$. Using Visp [10], the AprilTag on the workpiece is tracked and the coordinates of its corners are used as visual features $\boldsymbol{s} \in \mathbb{R}^k$, with $k = 8$. In Subection 3.3.1 we have shown the expression of the interaction matrix $\boldsymbol{L_s} \in \mathbb{R}^{8\times 6}$ associated to these features [48].

During the first phase of the task, the FCL only projects the external *wrench* into the feature space allowing the system to accommodate for unmodeled interaction forces. So, according to (6.7), we have: $\bar{\boldsymbol{f}}^{*}_{\boldsymbol{s_{ext}}} = -\boldsymbol{L_s}\bar{\boldsymbol{B}}_c^{-1}\,{}^c\boldsymbol{h}_c$, meaning that $\boldsymbol{K_{f_P}} = \mathbb{I}_8$, the identity matrix of dimension 8×8, $\boldsymbol{K_{f_I}} = 0\,s^{-1}$ and ${}^e\boldsymbol{h}_e^d = 0$. The admittance parameters have been chosen to have small interaction forces and sufficient damping to attenuate the oscillations that can be triggered while maintaining the overall reactiveness of the system ($\boldsymbol{M_s} = \mathbb{I}_8\,\frac{kg}{kg}$, $\boldsymbol{K_s} = diag(300)\frac{N/m}{kg}$, $\boldsymbol{D_s} = diag(200)\frac{N/m.s^{-1}}{kg}$). During the second phase, that is the force regulation phase, the FCL fully implements equation (6.7) with $\boldsymbol{K_{f_P}} = diag(0.2)$, $\boldsymbol{K_{f_I}} = diag(5)\,s^{-1}$ so as to guarantee fast force convergence and limited overshoot while keeping the system stable. Being now the peg's motion fully constrained inside the workpiece's hole, we make the controller way more stiffer ($\boldsymbol{K_s} = diag(20000)\frac{N/m}{kg}$, $\boldsymbol{D_s} = diag(400)\frac{N/m.s^{-1}}{kg}$). The desired wrench is set to ${}^e\boldsymbol{h}_e^d = [5\,N\ \ -5\,N\ \ -15\,N\ 0\ 0\ 0]^\top$ and is transformed to the camera frame as ${}^c\boldsymbol{h}_c^d = {}^c\mathbb{F}_e\,{}^e\boldsymbol{h}_e^d$. For both phases, the desired apparent constant inertia we want the camera to exhibit is $\bar{\boldsymbol{B}}_c = {}^c\mathbb{T}_e\bar{\boldsymbol{B}}_e^{-1}\,{}^c\mathbb{T}_e^\top$ with $\bar{\boldsymbol{B}}_e = diag(1\,\mathbb{I}_3\,kg, 0.1\,\mathbb{I}_3\,kg.m^2)$.

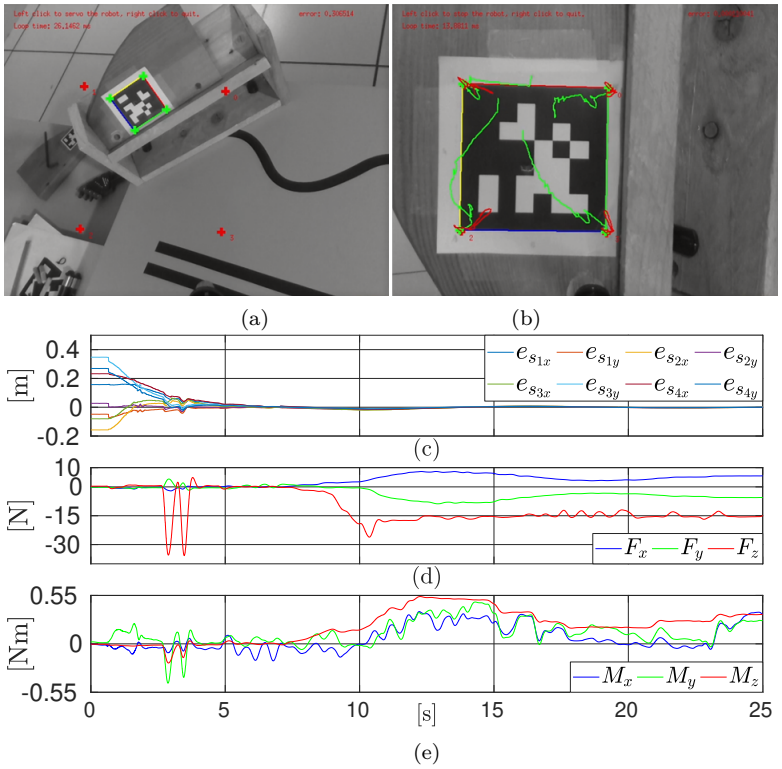


Figure 6.7 – Peg-in-hole experiment: (a): Initial camera view of the experiment. The green crosses are the current features while the red crosses are the references. (b): Final camera view. It is possible to appreciate (in green) the trajectory of the features in the image plane. (c): Feature errors ($\boldsymbol{s}^* - \boldsymbol{s}$), (d) End-effector forces and (e) moments.

The experimental setup and the framework's behaviour during the task execution are shown in Figures 6.6 and 6.7 respectively. As we can appreciate from the plots in Fig. 6.7c and the image points trajectory (almost straight) in Fig. 6.7b, the visual task begins to converge exponentially until it hits the structure, not being aware of it since there is neither a previous knowledge of the environment nor a pre-calculated trajectory. The impact force in the approach direction is approximately $35\,N$ and it projects into the feature space as virtual per unit of mass/inertia forces/torques that pull the four point features reference towards the target centroid causing the robot to slow down along this direction while it continues converging along the others. The robot then approaches the *hole* and the *peg* is successfully inserted. These two collisions are perfectly visible both in Figure 6.7b (small saw-teeth close to the desired features) and Figure 6.7d (two peaks along z-axis). Once the *peg* is within the *hole*, we observe that the visual error converges allowing the system to autonomously switch to the second phase in which the integral term in the FCL is activated. The lateral force pushes the workpiece into the groove and against the wall of the structure. The integral term makes the output of the FCL increase until the force error is nullified. The integral term creates a dominance of the force loop over the internal (positional) vision loop.

6.6.3 Extended External Hybrid vs External Hybrid

The external hybrid approach [57] has been shown to be successful where impedance- and hybrid-based vision/force schemes have failed. Then follows our interest in comparing this control scheme with the proposed one, as we inherit its structure and extend it, eliminating its drawbacks.

In Section 4.4 we have shown that, for the External Hybrid controller [57], a Cartesian displacement is firstly computed as $d\boldsymbol{X} = \boldsymbol{L}_{\boldsymbol{X}}^{-1}\boldsymbol{K}^{-1}({}^{e}\boldsymbol{h}_{e}^{d} - {}^{e}\boldsymbol{h}_{e})$ being $\boldsymbol{K} \in \mathbb{R}^{6\times6}$ the contact stiffness. This displacement is then projected in feature space using the interaction matrix $d\boldsymbol{s} = \boldsymbol{L}_{\boldsymbol{s}}d\boldsymbol{X}$. Finally, the compliant reference is obtained by adding the computed feature space displacement to the desired reference $\boldsymbol{s}^{*} = \boldsymbol{s}^{d} + d\boldsymbol{s}$.

To evaluate their relative performance over the task, we have executed different trials with both methods, each starting and ending from a different position. This time the workpiece is clamped on the work surface, preventing it from moving. The results of one of these experiments is shown in Figure 6.8.

To make the comparison as fair as possible, we choose the same gain $\lambda = 1.5\,s^{-1}$ for both controllers and the proportional gain of the external hybrid, *i.e.*, the

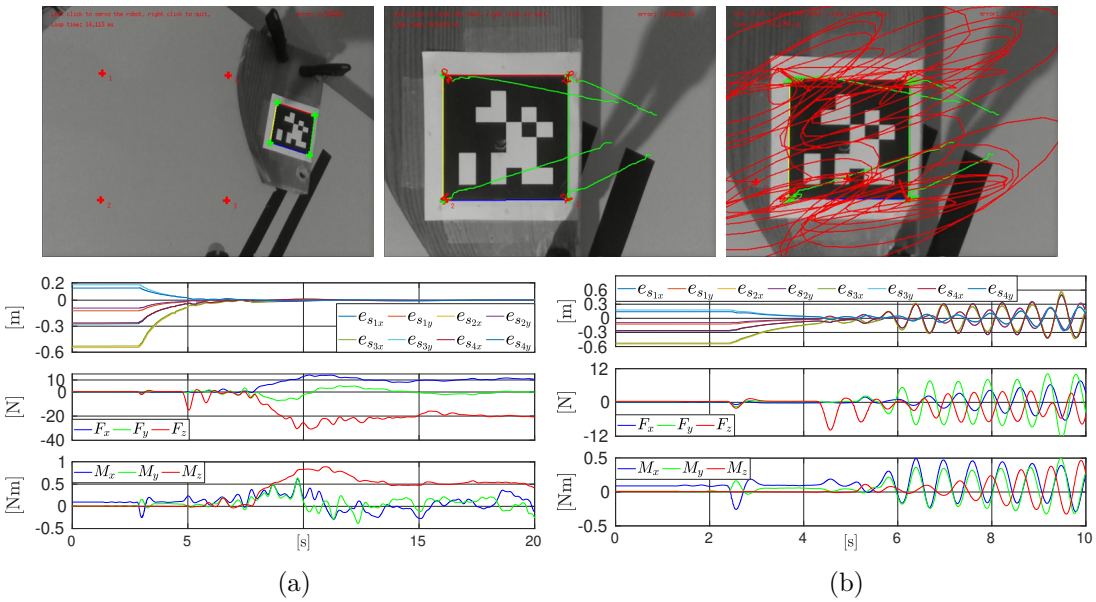


Figure 6.8 – Peg-in-hole comparison: (a) the results obtained with our method and, (b) those obtained with the external hybrid [57]. The images show the initial and final camera views with the current and reference feature trajectories in green and red respectively. The plots report, from top to bottom: the visual feature errors ($\boldsymbol{s}^* - \boldsymbol{s}$), measured end-effector forces and moments. ${}^e\boldsymbol{h}_e^d = [5N\,, 0\,, -20N\,, 0\,, 0\,, 0]^\top$

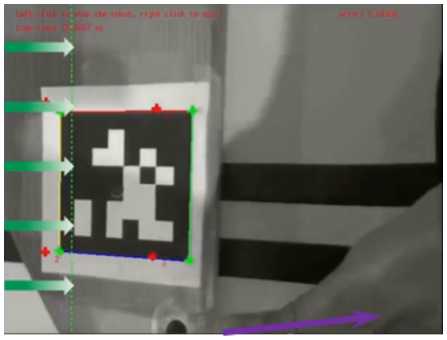
contact stiffness, in such a way that both methods achieve the same displacement of the visual reference when the same force is applied leading to choose $\boldsymbol{K} = (\boldsymbol{L_X L_s^\dagger K_s^{-1} L_s \bar{B}_c}^{-1} {}^c\mathbb{F}_e)^{-1}$. On the other hand, $\bar{\boldsymbol{B}}_c$, $\boldsymbol{K_{fP}}$, $\boldsymbol{K_{fI}}$ and $\boldsymbol{M_s}$ are as in the previous experiment, for both phases, while for this experiment we impose a less stiff behaviour by choosing $\boldsymbol{K_s} = diag(200)\frac{N/m}{kg}$ and $\boldsymbol{D_s} = diag(150)\frac{N/m.s^{-1}}{kg}$ for the first phase.

At the beginning, both methods starts converging towards the target with exactly the same behaviour. In fact, as long as there are no interaction forces, the reference is not modified and the task is a pure VS. When the *peg* approaches the *hole*, it hits the border before entering. As shown in Figure 6.8, for both the external hybrid and our method, the impact force along the approach direction stays quite limited even though for our method it is slightly higher. This is due to the presence of the damping term. Both methods succeed in the insertion of the *peg* tip but the increase of lateral forces once the *peg* is inside the workpiece, triggers the oscillation of the non-damped spring of the stiffness control for the external hybrid method. If these oscillations remain limited, they do not allow to reach the convergence of the visual reference while, in the event they explode as it is the case, it can lead to damage the manipulated object or even the robot

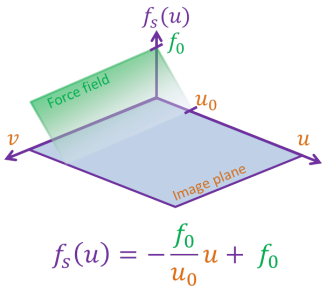
tool. This experimental result has shown that endowing the feature's motion with a dynamic capable of damping their velocity, improves the performance of the system, managing to dampen the oscillations that can be triggered.

6.7 Fictitious forces

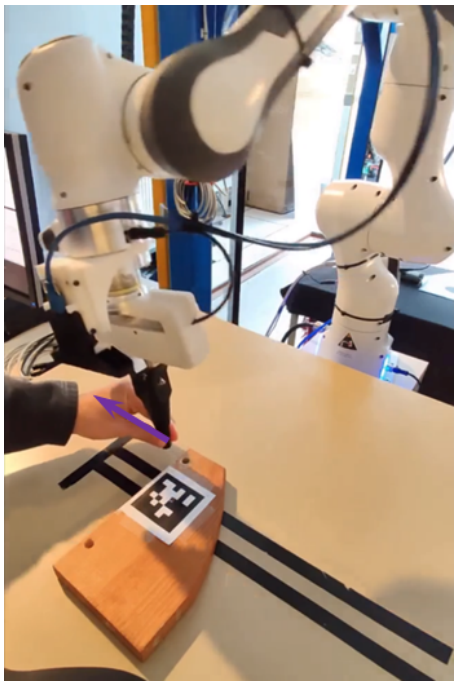
The vector of per unit of mass/inertia force/torques in the feature space is an artifact created to allow the physical forces to be mapped into the feature space. However, a vector of artificial forces in the feature space could be generated.



(a)



(b)



(c)

Figure 6.9 – Simultaneous interaction with both fictitious and physical forces. (a) Camera view: the reference features (red +) moves under the action of the physical force (purple arrow). A repulsive relative force field is generated on the left edge of the image plane (green arrows). The current features are the corners of the AprilTag (green +). (b) 3D visualization of the relative force intensity, the closer to the edge the higher. (c) Robotic system equipped with a camera in a *eye-in-hand* configuration. The operator physically pulls the end-effector (purple arrow) pushing the reference features towards the repulsive fictitious force field.

As can be seen from Figure 6.9, it is possible to generate, for example, a repulsive linear relative in per unit of mass/inertia force field on the left edge of the image plane, that prevents the reference to leave the *Field-of-View* when we physically interact with the manipulator's end-effector.

6.8 Summary

In this chapter we proposed a framework for dealing with the coupling of vision and force sensing directly in the feature space. The proposed controller presents the same hierarchical architecture of the External Hybrid vision/force control scheme [57] but with a substantial difference in the way we manage the compliant reference. Our controller, uses a second-order relationship between sensed forces and displacement quantities, *i.e.,* it implements an impedance law in feature space. This was possible thanks to the force map from Cartesian to feature space shown in Chapter 5. This framework allows to treat the combination of vision and force sensing in a unified way regardless the chosen visual features. Unlike the Hybrid vision-force scheme, we can control simultaneously all the system's degrees of freedom with both vision and force sensing. The hierarchical juxtaposition of the force control loop over the vision loop, avoids the rise of inconsistencies at actuation level and the convergence towards local minima, as for [57]. Furthermore, we have shown that simultaneous interaction with both physical and fictitious forces (i.e. generated in the image plane) can be handled in this framework to account for e.g., the physical interaction with the environment and/or a visually generated repulsive force field in a collision avoidance tasks, as well as the achievement of force regulation. Finally, we have shown that compliance is achieved in feature space, along the feature that defines the visual tasks. Due to the common architecture and properties shared with [57], we have named our controller the "Extended External Hybrid vision/force" control scheme, which features some improvements. The implementation results confirm the validity of this approach.

The derivations and experiments shown in this chapter are the result of the work published on the paper:

- **A. A. Oliva**, P. Robuffo Giordano, and F. Chaumette, "A general visual-impedance framework for effectively combining vision and force sensing in feature space," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp.4441–4448, 2021.
  Video: https://www.youtube.com/watch?v=Qw0Dnhq2uRQ

Part III Manipulator dynamic model and simulation

Chapter 7

Dynamic model of the Franka Emika's Panda robot

Contents

7.1 Introduction

The knowledge of accurate dynamic models is of fundamental importance for many robotic applications. It is necessary, in fact, for designing control laws with superior performance in free motion or when interacting with the environment [13], e.g., in strategies for the sensorless detection, isolation and reaction to unexpected collisions [84] or when regulating force or imposing a desired impedance control at the contact [85].

The controllers developed in Chapter 5 are indeed based on the manipulator model, which allows for inverse dynamic control. The cancellation of the manipulator dynamics is more effective the accurate the model is. On the other hand, as we will see in the next chapter, knowledge of the dynamic parameters is required for the dynamic simulation of the manipulator. Moreover, a first implementation step on the simulator was required to study the problem of second-order visual-servoing, followed by a second implementation step on the real platform. It would have been extremely difficult to determine the source of errors or control malfunctions, as well as their causes, without this intermediate step. The necessity of estimating these parameters is evident from the aforementioned considerations.

To obtain an estimation of the dynamic model, regression techniques are widely employed for industrial [86, 87] or humanoid robots [88, 89]. These techniques are hinged on a fundamental property: the linear dependence of the robot dynamic equations in terms of a set of ρ *dynamic coefficients* $\boldsymbol{\pi}_R \in \mathbb{R}^\rho$ [20], also known in the literature as *base parameters* [21], which are linear combinations of the *dynamic parameters* of each link composing the robot. In particular, each link has 10 parameters, specifying the mass, the position of the center of mass (CoM), and the 6 elements of the symmetric inertia tensor. Then, a robot with l links has a total $10\,l$ of such parameters, denoted as $\boldsymbol{p} \in \mathbb{R}^{10\,l}$. In addition, one may also include a number of parameters for modeling joint friction.

The regrouping of dynamic parameters, i.e., the dynamic coefficients, occurs because not all the parameters are excitable during motion (and therefore some of them have no influence on the robot dynamics), and others have an effect on the dynamics only in combination, that is, they are not identifiable separately. Hence, the dynamic parameters in $\boldsymbol{p}$ will appear as a combination and very rarely as singletons in $\boldsymbol{\pi}_R$.

The identification of the dynamic coefficients $\boldsymbol{\pi}_R$ is often sufficient for a number of robotic applications, such as dynamic motion robot control and motion planning, since knowledge of $\boldsymbol{\pi}_R$ allows for a numerical evaluation of the robot dynamic model in the Euler-Lagrange (E-L) form. However, the retrieval of a set of feasible

numerical values for the dynamic parameters $\boldsymbol{p}$ is also relevant. This is the case, for instance, when performing dynamic simulations via a CAD-based robotic simulator - like CoppeliaSim [75] - or when implementing torque-level control laws (such as the feedback linearization) under hard real-time constraints. In this case, a widely adopted solution is to use the recursive numerical Newton-Euler (N-E) algorithm, which is preferred to the evaluation of the symbolic computationally more expensive E-L approach (which relies on dynamic coefficients). However, usual N-E routines require the knowledge of the dynamic parameters $\boldsymbol{p}$ of each link in the kinematic chain, and not just of the dynamic coefficients $\boldsymbol{\pi}_R$. In [20], the problem of recovering a complete set of values for the original robot parameters starting from the identified dynamic coefficients was addressed. In general, this is a nonlinear problem admitting an infinite number of solutions. However, not all solutions are physically consistent (as example, negative masses may appear). To discard unfeasible solutions, we considered upper and lower bounds on each component of $\boldsymbol{p}$ by solving a constrained nonlinear optimization problem. Because of the ill-conditioned nature of the solution space, we used global optimization methods, such as simulated annealing.

The physical consistency of the identified dynamic parameters, as introduced in [90], is currently attracting more and more attention. Researchers have treated the problem of physical feasibility within the framework of linear matrix inequalities (LMIs), solving the problem of the identification of a physically consistent set of parameters by means of semi-definite programming (SDP) techniques [91]. Recently, this framework has been enriched by the addition of the triangle inequality of the inertia tensors [23, 24], a constraint which was originally mentioned in [22], providing the scientific community with a valuable tool for the dynamic identification problem. The approach presented in this chapter can be considered as an alternative to the LMI-SDP framework.

All these approaches act on the parameter identification phase, obtaining from this dynamic coefficients that are typically different from the classic ordinary least squares (OLS) solution [86, 87].

The approach presented in [24, 91] requires to express constraints as linear matrix inequalities, while the proposed optimization algorithm manages linear, nonlinear and even conditional constraints (e.g., *if-else*), without any mathematical manipulation. This additional flexibility allows to handle directly nonlinear constraints coming from the geometric shape of cylindrical or spherical links (like for Universal Robots manipulators or for the sixth link of the KUKA LWR IV+), or when the use of approximate box constraints may generate solutions which are even unfeasible (e.g., a center of mass outside a convex link of the robot).

On the other hand, a drawback of the proposed algorithm is that convergence to a global optimum in a finite number of steps cannot be guaranteed. It has to be noted that, whatever method is adopted, it is very unlikely that the original dynamic parameters are retrieved; indeed, the more constraints are supplied, the closer the solution will be to the actual one. Thanks to the algorithmic nature of the presented framework, a wide range of constraints can be easily added.

The proposed approach is general and can be applied to a large class of robot manipulators. In this work, as case study, we apply it to the Franka Emika Panda robot (see Figure 7.1), for which we have identified a complete set of feasible dynamic parameters.

7.2 The Panda robot

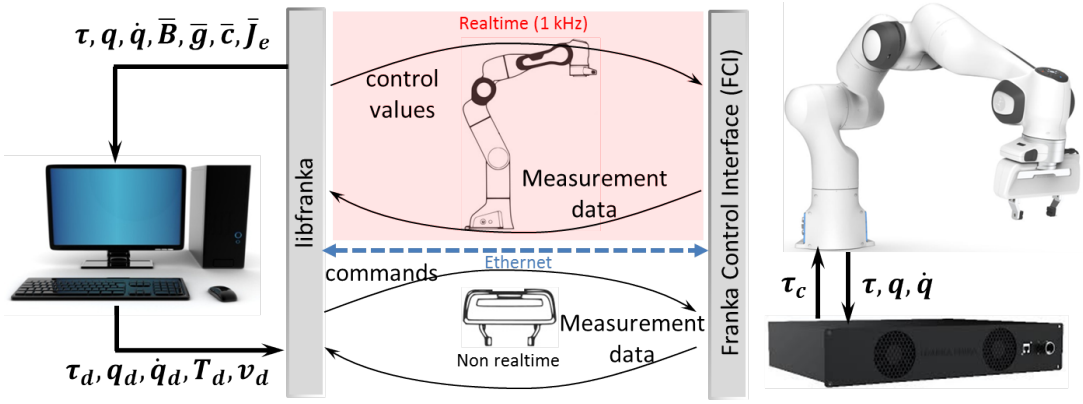


Figure 7.1 – The Panda arm equipped with its gripper and interfaced with its controller. The signal flows from and to the controller are depicted: The user sends a command to the *libfranka* API that communicates with the FCI controller through Ethernet. This input is then converted to a commanded torque $\boldsymbol{\tau}_c$ to the robot that returns the measured joint torque $\boldsymbol{\tau}$, as well as the joint positions $\boldsymbol{q}$ and velocities $\dot{\boldsymbol{q}}$. The FCI controller computes the numerical values for the inertia matrix $\bar{\boldsymbol{B}}(\boldsymbol{q})$, the gravity vector $\bar{\boldsymbol{g}}(\boldsymbol{q})$, the Jacobian $\bar{\boldsymbol{J}}_e(\boldsymbol{q})$, and the Coriolis term $\bar{\boldsymbol{c}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ as well. These data are sent back to the user through the *libfranka* API. More details can be found at: https://frankaemika.github.io/docs/index.html.

The robot considered in the experiments and simulations all along this thesis work, is the popular Panda robot manufactured by Franka Emika. Panda is a torque-controlled cobot that is attracting a large interest both on research and in small-sized business because of its safety, versatility, reliability and relatively low price (less than 20 k€). Its total weight is about 18 kg and it is capable of handling payloads up to 3 kg. It is a redundant manipulator with $n = 7$ revolute joints, each of which is equipped with a link-side torque sensor. This robot can

be controlled with five different interfaces: at joint level by sending torque $\boldsymbol{\tau}_d$, position $\boldsymbol{q}_d$ or velocity $\dot{\boldsymbol{q}}_d$ commands or by sending Cartesian pose $\boldsymbol{T_d}$ or velocity $\mathbf{v_d}$ commands to the Franka Control Interface (FCI) through the *libfranka*, its open-source C++ API. Through this library it is possible to send real-time control values (at 1 kHz) and have full access to the robot state, *e.g.*, the joint positions $\boldsymbol{q}$ and velocities $\dot{\boldsymbol{q}}$, as well as the link side torque vector $\boldsymbol{\tau}$. Moreover, it returns the *numerical* values of the inertia matrix $\overline{\boldsymbol{B}}(\boldsymbol{q})$, of the gravity vector $\bar{\boldsymbol{g}}(\boldsymbol{q})$, as well as the Jacobian $\bar{\boldsymbol{J}}_e(\boldsymbol{q})$ and the Coriolis vector $\bar{\boldsymbol{c}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ at a given joint configuration $\boldsymbol{q}$ and velocity $\dot{\boldsymbol{q}}$. These data will be of fundamental importance for the identification process of the dynamic model of the robot described in the next section. Robots with the Research Interface can also be controlled using the franka_ros metapackage [92], which integrates and exposes the complete *libfranka* API to the ROS ecosystem. This metapackage also provides hardware abstraction and model description in Universal Robotic Description Format (URDF) to simulate the robot in Gazebo [93].

The FCI controller is designed in such a way that the command inputs given by the user are appropriately manipulated so that the motors generate the proper torque $\boldsymbol{\tau}_c$ for the commanded task. Figure 7.1 depicts all the above-mentioned control signals.

7.3 Identification procedure

In order to derive the symbolic dynamic model of a robot with elastic joints, such as the Franka Emika Panda, one may follow the procedure presented in [94], separating the motor torques from the link-side torques. Nevertheless, the particular features offered by the robot controller allow us to simplify the modeling: in fact, since the FCI controller is able to return the estimations of the link-side torques (exploiting the motor position measures read from the encoders), we are able to adopt the classical model structure as for a rigid joints robot, neglecting the elasticity [95], and henceforth $\boldsymbol{\tau} \in \mathbb{R}^n$ is the vector of the *link-side* torques.

Exploiting the features offered by the controller of the Panda robot, as we mention in Section 7.2), it is possible to avoid the classical procedure involving exciting trajectories [96], and obtain the estimates of the gravitational and inertial coefficients by collecting a set of static positions only by means of a reverse engineering procedure (See Appendix A for a comparison of the dynamic coefficients estimated through the reverse engineering process with those obtained from the classical approach that exploits exciting trajectories). The same procedure had been used to retrieve the dynamic coefficients of the KUKA LWR robot [95, 97]:

in that case, the gravitational and the inertial coefficients have been estimated separately, while now a slightly different approach has been used, due to the fact that many coefficients can be retrieved both from the inertia matrix and from the gravity vector. As a first operation, one has to rearrange the symbolic inertia matrix $\boldsymbol{B}(\boldsymbol{q})$ in a vector form (the *inertia stack*), by exploiting its symmetry. Having for the Panda robot $n = 7$ joints, we obtain a vector $\tilde{\boldsymbol{b}}(\boldsymbol{q}) \in \mathbb{R}^m$, with $m = n(n+1)/2 = 28$ components, containing all the lower triangular elements of $\boldsymbol{B}(\boldsymbol{q})$. Now, it is possible to obtain – as described in Section 2.3.1 – the symbolic regressor $\boldsymbol{Y}_s(\boldsymbol{q})$ from the column vector $\boldsymbol{\varsigma}(\boldsymbol{q}) \in \mathbb{R}^{m+n}$ in such a way that

$$\boldsymbol{\varsigma}(\boldsymbol{q}) = \begin{bmatrix} \tilde{\boldsymbol{b}}(\boldsymbol{q}) \\ \boldsymbol{g}(\boldsymbol{q}) \end{bmatrix} = \boldsymbol{Y}_s(\boldsymbol{q})\boldsymbol{\pi}_s, \tag{7.1}$$

where $\boldsymbol{\pi}_s \in \mathbb{R}^{10n}$ is the vector containing both the gravitational and inertial dynamic parameters, which are the same – excluding joint friction and motor inertias – as in vector $\boldsymbol{\pi}$ of eq. (2.33).

In order to obtain a numerical estimation of the dynamic coefficients vector, a data acquisition procedure should be carried out: in the general case, performing exciting trajectories is required in order to span all the admissible joint positions, velocities and accelerations (see eq. (2.34)). Since $\boldsymbol{\varsigma}(\boldsymbol{q})$ depends only on the joint positions $\boldsymbol{q}$, it is just sufficient to retrieve data by imposing static positions.

The *libfranka* API exposes the numerical evaluation of the gravity vector and the inertia matrix of the Panda robot at the current link position. Therefore, it is possible to collect a fair amount of data even only in a static way, i.e., bringing the manipulator to a desired configuration and then retrieving and storing the numerical values of the gravity vector and the inertia matrix. This acquisition procedure can be performed during a motion as well. The main advantage of imposing static joint positions is that this procedure avoids any influence of friction and uncertainty (e.g., due to measure noise or to any unmodeled phenomenon).

The data is acquired and collected in a list of M different (special and/or random) configurations, under the weak condition

$$Mn \gg p. \tag{7.2}$$

For a generic configuration $\boldsymbol{q}_k$, with $k = 1 \dots M$, we have

$$\overline{\boldsymbol{\varsigma}}_k = \begin{bmatrix} \overline{\boldsymbol{b}}_k \\ \overline{\boldsymbol{g}}_k \end{bmatrix} = \overline{\boldsymbol{Y}}_{sk}\boldsymbol{\pi}_s, \tag{7.3}$$

where $\overline{\boldsymbol{b}}_k$ and $\overline{\boldsymbol{g}}_k$ represent, respectively, the numerical inertia stack vector and the numerical gravity vector, as they are retrieved from the *libfranka* interface at a given configuration $\boldsymbol{q}_k$, and $\overline{\boldsymbol{Y}}_{sk} = \boldsymbol{Y}_s(\boldsymbol{q}_k)$ is the evaluated regressor.

When all data are collected, they can be stacked into a vector $\overline{\boldsymbol{\varsigma}}$ such as:

$$\overline{\boldsymbol{\varsigma}} = \begin{bmatrix} \overline{\boldsymbol{\varsigma}}_1 \\ \overline{\boldsymbol{\varsigma}}_2 \\ \vdots \\ \overline{\boldsymbol{\varsigma}}_N \end{bmatrix} = \begin{bmatrix} \overline{\boldsymbol{Y}}_{s1} \\ \overline{\boldsymbol{Y}}_{s2} \\ \vdots \\ \overline{\boldsymbol{Y}}_{sN} \end{bmatrix} \boldsymbol{\pi}_s = \overline{\overline{\boldsymbol{Y}}}_s \boldsymbol{\pi}_s. \tag{7.4}$$

Nevertheless, the regressor $\overline{\overline{\boldsymbol{Y}}}_s$ is typically rank-deficient: this implies that the elements of the vector $\boldsymbol{\pi}_s$ are not fully identifiable. Therefore, it is necessary to drop linear dependent columns of the regressor in order to reach a full (column) rank condition (i.e., by means of the Gauss-Jordan elimination technique). As a consequence, some dynamic parameters will be grouped together accordingly, in the form of dynamic coefficients [21]. Exploiting condition (7.2) on the minimal number M of samples to retrieve, the ill-conditioning of the matrix is avoided. Denoting as $\hat{\boldsymbol{\pi}}_{s,R}$ the vector containing the regrouped parameters (a.k.a. dynamic coefficients), and as $\overline{\overline{\boldsymbol{Y}}}_{s,R}$ the full rank numerical regressor, eq. (7.4) is solved using a least squares technique as

$$\hat{\boldsymbol{\pi}}_{s,R} = \left(\overline{\overline{\boldsymbol{Y}}}_{s,R}^{\top} \overline{\overline{\boldsymbol{Y}}}_{s,R} \right)^{-1} \overline{\overline{\boldsymbol{Y}}}_{s,R}^{\top} \overline{\boldsymbol{\varsigma}} = \overline{\overline{\boldsymbol{Y}}}_{s,R}^{\dagger} \overline{\boldsymbol{\varsigma}}, \tag{7.5}$$

where †, as usual, denotes pseudoinversion.

Once one has the dynamic coefficients estimation $\hat{\boldsymbol{\pi}}_s$, from eq. (7.1), it is possible to obtain the estimates $\hat{\boldsymbol{g}}(\boldsymbol{q})$ and $\hat{\boldsymbol{B}}(\boldsymbol{q})$ as:

$$\hat{\boldsymbol{\varsigma}}(\boldsymbol{q}) = \begin{bmatrix} \hat{\hat{\boldsymbol{b}}}(\boldsymbol{q}) \\ \hat{\boldsymbol{g}}(\boldsymbol{q}) \end{bmatrix} = \boldsymbol{Y}_{s,R}(\boldsymbol{q}) \hat{\boldsymbol{\pi}}_{s,R}, \tag{7.6}$$

where $\boldsymbol{Y}_{s,R}(\boldsymbol{q})$ is the symbolic regressor pruned of the dependent columns (according to the full-rank matrix $\overline{\overline{\boldsymbol{Y}}}_{s,R}$) and $\hat{\boldsymbol{B}}(\boldsymbol{q})$ is built from the estimated inertia stack $\hat{\hat{\boldsymbol{b}}}(q)$. Finally, the estimation of the Coriolis and centrifugal forces vector $\hat{\boldsymbol{c}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is derived from $\hat{\boldsymbol{B}}(\boldsymbol{q})$ using the Christoffel's symbols according to [13]. The form of the inverse dynamics formula, providing an estimation of the joint torques $\hat{\boldsymbol{\tau}}$ needed to accomplish a given trajectory ($\boldsymbol{q}$,$\dot{\boldsymbol{q}}$,$\ddot{\boldsymbol{q}}$), is therefore:

$$\hat{\boldsymbol{\tau}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \hat{\boldsymbol{B}}(\boldsymbol{q}) \ddot{\boldsymbol{q}} + \hat{\boldsymbol{c}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \hat{\boldsymbol{g}}(\boldsymbol{q}). \tag{7.7}$$

7.3.1 Friction estimation

If a validation trajectory is executed and the measured torques $\boldsymbol{\tau}$ are compared (after a filtering procedure) with the estimated torques $\hat{\boldsymbol{\tau}}$ generated by eq. (7.7),

a difference in the two signals may be observed. This discrepancy is due to estimation errors (e.g., due to the noise affecting the measurements) or to unmodeled effects, such as joint friction. Typically, the latter effect, assumed to act separately on each joint, is expressed as an additional torque $\tau_{f,j},\ j \in \{1, \dots, n\}$:

$$\tau_{f,j}(\dot{q}_j) = f_{v,j}\dot{q}_j + f_{c,j}\text{sign}(\dot{q}_j) + f_{o,j}, \tag{7.8}$$

where $f_{v,j}$ and $f_{c,j}$ represent, respectively, viscous and Coulomb friction, while $f_{o,j}$ is the Coulomb friction offset. Model (7.8), although being simple and effective, has the main drawback of exhibiting sudden discontinuities in the neighborhood of $\dot{q}_j = 0$. In order to attenuate this chattering, a sigmoidal friction model can be used for avoiding discontinuities for low joint velocities. In case the viscous effects are negligible and a symmetric behavior for positive and negative joint velocities is observed (as for the Panda robot), $\tau_{f,j}(\dot{q}_j)$ can be expressed as the following function:

$$\tau_{f,j}(\dot{q}_j) = \frac{\varphi_{1,j}}{1 + e^{-\varphi_{2,j}(\dot{q}_j + \varphi_{3,j})}} - \frac{\varphi_{1,j}}{1 + e^{-\varphi_{2,j}\varphi_{3,j}}}, \tag{7.9}$$

which is characterized by 3 parameters for each joint ($\varphi_{1,j},\ \varphi_{2,j},\ \varphi_{3,j}$). In order to estimate the $3n = 21$ friction parameters for the Panda robot, trajectories spanning all possible joint velocities can be executed for each joint (possibly, keeping the others at rest). Computing the inverse dynamics for the previous trajectories according to (7.7), the difference $\Delta\tau_j = \tau_j - \hat{\tau}_j$ (where τ_j and $\hat{\tau}_j$ are, respectively, the measured and the estimated joint torques) can be interpreted as a measure of the friction effects. Solving a least squares problem, it is possible to find the parameters that make the curve (7.9) to fit data at best – e.g., by means of a Nelder-Mead routine.

The presented reverse engineering approach has the main advantage of allowing an estimation of the dynamic parameters by exploiting only static measures (which are affected by low noise) without the need of numerical derivations, which can dramatically affect the results of the identification process. Moreover, having a separate step for joint friction identification allows to repeat only this step (instead of the whole identification process) when lubrication changes or when moving to a new instance of the same robot. Conversely, we must rely on the numerical values of $\bar{\boldsymbol{\varsigma}}_k$ provided by the manufacturer.

7.4 Retrieval of feasible parameters

In [20], authors presented a framework for retrieving a feasible set of dynamic parameters from the previously identified dynamic coefficients, by solving a nonlinear global optimization problem.

For some purposes, in fact, the estimated dynamic coefficients (linear combination of dynamic parameters, such as masses, position of the centers of mass and inertia tensors) are not sufficient, and an estimation of the dynamic parameters themselves is needed. This occurs, for instance, if the dynamic parameters values are needed for simulating the robot behavior in a CAD software (like CoppeliaSim as we will see in the next chapter), or if a Newton-Euler routine is used to compute the joint torques estimation during the robot motion, with strict time constraints (in this case, a Lagrangian approach may not fit the hard real-time requirements), or in case one is interested in computing the wrenches acting on the robot joints.

The approach presented in [20] is able to return a possible and feasible set of the dynamic parameters: in general, infinite solutions exist (among which the real one) and the more constraints are given, the closer to the real solution is the returned one.

Starting from the identified vector of dynamic coefficients $\hat{\boldsymbol{\pi}}_R$, the following transformations may be applied to the corresponding symbolic vector $\boldsymbol{\pi}_R(\boldsymbol{p})$ (parallel axis theorem, see [13]):

$$\begin{aligned} J_{ixx} &\rightarrow I_{ixx} + m_i c_{iy}^2 + m_i c_{iz}^2 \; ; \; J_{ixy} \rightarrow I_{ixy} - c_{ix} c_{iy} m_i \\ J_{iyy} &\rightarrow I_{iyy} + m_i c_{ix}^2 + m_i c_{iz}^2 \; ; \; J_{ixz} \rightarrow I_{ixz} - c_{ix} c_{iz} m_i \\ J_{izz} &\rightarrow I_{izz} + m_i c_{ix}^2 + m_i c_{iy}^2 \; ; \; J_{iyz} \rightarrow I_{iyz} - c_{iy} c_{iz} m_i \end{aligned} \tag{7.10}$$

for each link ℓ_i, $i = 1 \dots n$. We can now rearrange the parameters vector $\boldsymbol{p} = \begin{bmatrix} \boldsymbol{p}_1^\top & \boldsymbol{p}_2^\top & \boldsymbol{p}_3^\top \end{bmatrix}^\top \in \mathbb{R}^{10n}$ as:

$$\begin{aligned} \boldsymbol{p}_1 &= \begin{bmatrix} m_1 & \dots & m_n \end{bmatrix}^\top \in \mathbb{R}^n, \\ \boldsymbol{p}_2 &= \begin{bmatrix} c_{1x} & c_{1y} & c_{1z} & \dots & c_{nx} & c_{ny} & c_{nz} \end{bmatrix}^\top \in \mathbb{R}^{3n}, \\ \boldsymbol{p}_3 &= \begin{bmatrix} \boldsymbol{\mathcal{I}}_1^\top & \dots & \boldsymbol{\mathcal{I}}_n^\top \end{bmatrix}^\top \in \mathbb{R}^{6n}, \end{aligned} \tag{7.11}$$

where

$$\boldsymbol{\mathcal{I}}_i = \begin{bmatrix} I_{ixx} & I_{ixy} & I_{ixz} & I_{iyy} & I_{iyz} & I_{izz} \end{bmatrix}^\top. \tag{7.12}$$

It is possible to provide lower bounds (LB) and upper bounds (UB) to $\boldsymbol{p}$ based on *a priori* information. In particular, condition (2.37) is managed by assigning a lower bound of zero to each link mass. Upper bounds for the masses are assigned exploiting, for instance, data retrieved from the datasheets of the robot. Moreover, for each link, one can easily infer that the center of mass is located inside the smallest parallel box which includes the link geometry, in the most general case. The lower and upper bounds are then set in order to guarantee a physical meaning to the obtained solution. In order to retrieve a possible set of dynamic parameter $\hat{\boldsymbol{p}}$, we propose to solve the nonlinear optimization problem depicted in Algorithm 1.

Algorithm 1: Parameters retrieval

1 $\boldsymbol{p}_0 \leftarrow LB + (UB - LB)\boldsymbol{u}, \quad \text{with} \quad \mathrm{u} \sim \mathcal{U}(0,1)$;

2 $\xi_1 \leftarrow 0$;

3 **for** $k = 1, \ldots, \kappa$ **do**

4 // Start the optimization from the previous step solution

5 $p_{k,\text{init}} \leftarrow p_{k-1}$;

6 // Solve the optimization problem using SA and Nelder-Mead as IP

$$\begin{cases} \min\limits_{\boldsymbol{p}_k} f(\boldsymbol{p}_k) &= \phi(\boldsymbol{p}_k) + \xi_k \gamma(\boldsymbol{p}_k) \\ &= \|\boldsymbol{\pi}(\boldsymbol{p}_k) - \hat{\boldsymbol{\pi}}\|^2 + \xi_k \textstyle\sum_\iota g(h_\iota(\boldsymbol{p}_k)) \\ \text{s.t.} \quad & LB \leq \boldsymbol{p}_k \leq UB \end{cases}$$

$$\xi_{k+1} \leftarrow 10k \tag{7.13}$$

7 **end**

The first two lines of Algorithm 1 are the initialization steps: the starting point is randomly selected between the lower and the upper bounds using a uniform distribution. Moreover, ξ_1 is set to zero. Lines $3-6$ of Algorithm 1 consist in solving the constrained nonlinear optimization problem κ times: at a given step $k = 1 \ldots \kappa$, the initial state is the optimal solution found at step $k-1$. The objective function presents also an exterior penalty function, in which ξ_k is the – progressively increasing – penalty coefficient: this function provides a penalty in case one of any additive constraint $h_\iota(\boldsymbol{p}_k)$ is violated; function $g(\cdot)$ is chosen to return a measure of the constraint violation. In particular, two kinds of constraints have been included:

- for each link ℓ_i, the triangle inequality (2.42) must be satisfied;
- the total sum of the link masses must be in a given range, that is:

$$m_{\text{rob,min}} \leq \sum_i m_i \leq m_{\text{rob,max}}. \tag{7.14}$$

Note that the presented framework is extremely flexible, and further external constraints can be easily included. The manifold generated by the cost function $f(\boldsymbol{p}_k)$ contains multiple local minima, and therefore a global optimization method, like genetic algorithms [98] or simulated annealing (SA) [99] is mandatory to address the problem (7.13). In the present case, we have used SA, applying a more sophisticated interior-point (IP) Nelder-Mead local optimization algorithm

at the end of each SA iteration k. Moreover, Q runs have been launched having a different random initial point $\boldsymbol{p}_0$, in order to span as much as possible the cost function manifold.

The improvement of the parameters retrieval framework (with respect to the one presented in [20]) has been proved necessary since the introduction of some constraints (*e.g.* instance the triangle inequality of the inertia tensors) eventually led the algorithm to get stuck in local minima. This was caused by recurring abrupt changes in the cost function given by the constant penalty function adopted before. Therefore, a violation-dependent penalty [100] has been implemented for the present algorithm, solving this problem. Moreover, the sequence of successive runs of the algorithm could in practice help in improving the solution.

The term $\phi(\boldsymbol{p}_k)$ of the parameters retrieval algorithm in eq. (7.13) requires to be computed a previous identification step returning the coefficients estimation $\hat{\boldsymbol{\pi}}$. Another possible $\phi(\boldsymbol{p}_k)$ function, yielding to a single-step procedure, is described in Appendix A.

7.5 Results

In this section, results from the dynamic coefficients and joint friction estimations for the Panda robot are reported, in addition to results from the parameters retrieval procedure.

In order to obtain a numerical estimation of the dynamic coefficients $\boldsymbol{\pi}_{s,R}$ (see eq. (7.5)), the numerical values of the inertia matrix and the gravity vector are retrieved from a set of $M = 1010$ static positions, spanning the whole joint space, according to the robot documentation[1]. The symbolic vector $\boldsymbol{\varsigma}(\boldsymbol{q})$ (see eq. (7.1)) has been computed according to [21], using the modified Denavit-Hartenberg convention.

Since our robot is mounted on a table parallel to the ground, the first element of the gravity vector (relative to joint 1) is not informative, and therefore it is discarded.

Stacking all the numerical quantities of the data acquisition phase and after evaluating the corresponding regressor (see eq. (7.4)), we obtained that $\text{rank}(\overline{\overline{\boldsymbol{Y}}}_s) = 43$: using the Matlab function `rref`, which implements Gauss-Jordan elimination technique, we finally obtained the dynamic coefficients estimations

[1]The joint position bounds of the Franka Emika Panda robot can be found at: https://frankaemika.github.io/docs/control_parameters.html

$\hat{\boldsymbol{\pi}}_{s,R}$ according to eq. (7.5), from a regressor $\overline{\overline{\boldsymbol{Y}}}_{s,R}$ whose condition number is 49. Moreover, the relative error percentage of predictions (defined in eq. (84) of [91]) for the identification set is 0.031%. Computing their standard deviations (see [87, 91]), we found that two coefficients exhibit a standard deviation greater than 30%, and therefore they were discarded (since their estimations are not reliable). All the estimated dynamic coefficients are reported with their standard deviations in the Appendix A, together with a comparison with the corresponding coefficients obtained from the classical identification procedure (see eq. (2.34)).

From the identified dynamic coefficients $\hat{\boldsymbol{\pi}}_{s,R}$, we were able to reconstruct the inertia matrix $\hat{\boldsymbol{B}}(\boldsymbol{q})$, the gravity vector $\hat{\boldsymbol{g}}(\boldsymbol{q})$ and the Coriolis and centrifugal force vector $\hat{\boldsymbol{c}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ following a Lagrangian approach (see Sec. 7.3).

After deriving the inverse dynamic model (7.7), we validate it by comparing the measured joint torques with the estimated ones during several motions. In particular, the robot was commanded in velocity-mode by means of sinusoidal trajectories in the joint space. In other words, each joint is commanded according to the following equation:

$$\dot{q}_{i,\text{des}}(t) = A_i \ \sin\left(\frac{2\,\pi}{T_i} t\right), \quad i \in [1, \ldots, n] \tag{7.15}$$

where A_i is the amplitude of the velocity profile and T_i is the period of the sinusoidal signal for the i-th joint. The numerical values for A_i and T_i, $i \in [1, \ldots, n]$ for a typical experiment are reported in Table 7.1, where 5458 samples were collected. The joint torque signals were recorded (and filtered through a 4-th order zero-phase digital Butterworth filter with a cutoff frequency of 1 Hz) during this motion, and compared with our Lagrangian inverse dynamics estimations $\hat{\boldsymbol{\tau}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ (from eq. (7.7)), feeding that model with the measured joint positions and velocities, and with the joint accelerations obtained by numerical differentiation of the filtered velocities. The joint torque comparison is reported in Figure 7.2: the torque estimations are almost perfectly superimposing the measured ones for joints $1 \ldots 4$, while the last joints show some discrepancies. One can notice, though, that the difference between the two signals strongly depends on joint velocities (it is more evident, e.g., for joint 7): this behavior is typical of joint friction.

Therefore, we performed the estimation of the joint friction according to the procedure reported in Section 7.3.1: we collected more than 10k samples of joint velocities and torques during sinusoidal motions in (7.15); eventually, we found that the friction model that best fits the data was given by a sigmoidal function (7.9), which is characterized by 3 parameters per joint. These were estimated by solving a nonlinear least squares problem by means of a Nelder–Mead routine

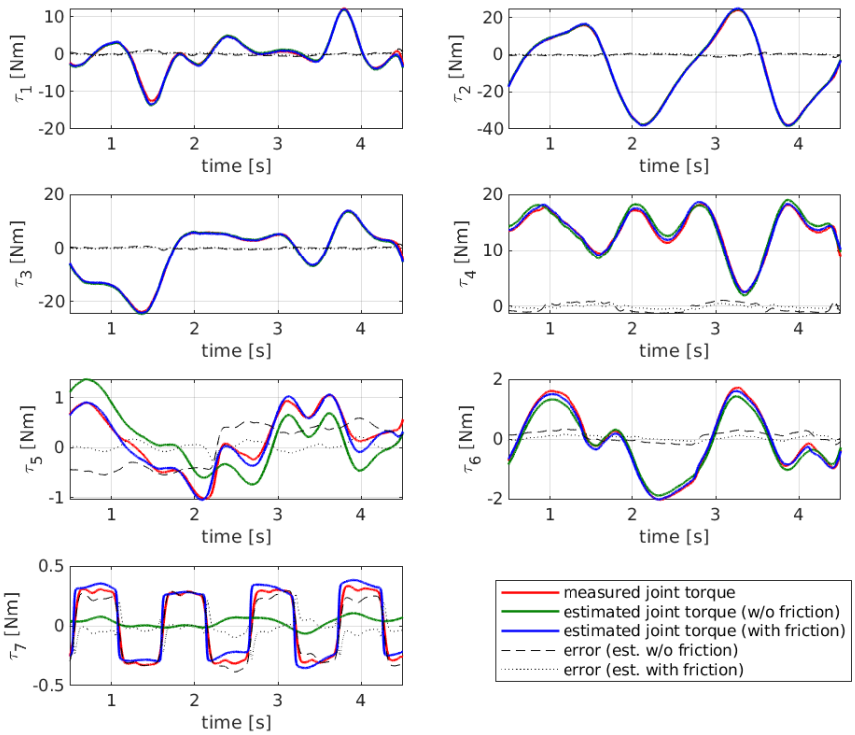


Figure 7.2 – Comparison between the torques of the derived E-L dynamic model. The red lines represent the measured joint torques during the validation experiment reported in Tab 7.1, while the green lines are the torque estimations $\hat{\boldsymbol{\tau}}$ computed according to eq. (7.7)). The blue lines represent the joint torque estimates comprehensive of the joint friction term (7.9). The dashed and the dotted lines are the errors between the torque sensors readings and, respectively, the torques estimates without and with the friction component.

(Matlab function `fminsearch`), using as fitting data the differences $\Delta\tau_i$ between the measured joint torques and the estimated torques $\hat{\tau}_i$ for each joint i separately. The results of the fitting procedures are reported in Fig. 7.3, while Table 7.2 provides the numerical values of the parameters obtained with the fitting procedure described in Section 7.3.1. These parameters refer to the equations (7.9) which models the friction acting at the robot joints.

Table 7.1 – Amplitudes A_i and periods T_i, $i \in [1, \ldots, n]$ of the sinusoidal trajectories in eq. (7.15) used for validating the friction acting on the robot joints.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**units**
A_i	2.21	-2.21	1.2	-2.1	-2.3	2.1	-2.5	rad/s
T_i	3.68	2.04	2.98	1.75	4.43	2.749	1.06	s

Table 7.2 – Estimated joint friction parameters.

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Joint 7	units
φ_1	5.4615e-01	0.87224	6.4068e-01	1.2794e+00	8.3904e-01	3.0301e-01	5.6489e-01	N·m
φ_2	5.1181e+00	9.0657e+00	1.0136e+01	5.5903e+00	8.3469e+00	1.7133e+01	1.0336e+01	s/rad
φ_3	3.9533e-02	2.5882e-02	-4.6070e-02	3.6194e-02	2.6226e-02	-2.1047e-02	3.5526e-03	rad/s

Finally, adding the newly estimated joint friction components $\hat{\boldsymbol{\tau}}_f(\dot{\boldsymbol{q}}) = \begin{bmatrix} \hat{\tau}_{f,1}(\dot{q}_1) & \hat{\tau}_{f,2}(\dot{q}_2) & \dots & \hat{\tau}_{f,7}(\dot{q}_7) \end{bmatrix}^\top$ to the previous estimations $\hat{\boldsymbol{\tau}}$, we obtained a satisfactory compensation, as shown with the blue solid lines of Fig. 7.2.

Despite the estimated inverse dynamic model in Lagrangian form described so far, is very cumbersome and computationally intensive for a 7 DoF robot, we manage to release a C++ library, with some naive code optimizations, that reliably works under hard real-time constraints [2] (the mean execution time of the model is ~ 0.05 ms on a Dell Latitude 7490 - Intel CORE i7-8650U - 32Gb RAM). For performance purposes, a Newton-Euler (N-E) approach is often more appropriate and effective to quickly return joint torque estimates, due to its recursive form. Nevertheless, a N-E routine requires the dynamic parameters (masses, inertia tensors and centers of mass of each link) of the robot. Exploiting the parameters retrieval algorithm (7.13), though, we are able to extract a *feasible* set of dynamic parameters, which provides the same dynamics (although there is no guarantee that the estimated robot parameters set is coincident with the real one).

In order to implement Algorithm 1, the (bounded) simulated annealing Matlab function `simulannealbnd` has been used, together with the IP hybrid function `fmincon`; the problem parameters were $Q = 100$ (total number of independent runs, providing Q solutions – possibly coincident, since there might exist multiple minima) and $\kappa = 10$ (number of successive runs). The penalty functions $g(h_\iota(\boldsymbol{p}_k))$ are chosen as the distance functions of the external constraints, that is:

$$
\begin{aligned}
g_{1,i} &= -\min\left\{0, \operatorname{tr}(\boldsymbol{I}_{\ell_i})/2 - \lambda_{\max}(\boldsymbol{I}_{\ell_i})\right\}, \quad i \in [1, \dots 7] \\
g_2 &= -\min\left\{0, m_{\text{rob,max}} - \sum_{i=1}^{7} m_i, \sum_{i=1}^{7} m_i - m_{\text{rob,min}}\right\}
\end{aligned} \tag{7.16}
$$

where $g_{1,i}$ regards the triangle inequality on each inertia tensor (eq. 2.42), while g_2 regards the constraint on the total mass of the robot (we chose $m_{\text{rob,min}} = 16$ kg and $m_{\text{rob,max}} = 20$ kg).

[2] The Lagrangian C++ library can be found in the following repository https://github.com/marcocognetti/FrankaEmikaPandaDynModel/tree/master/cpp.

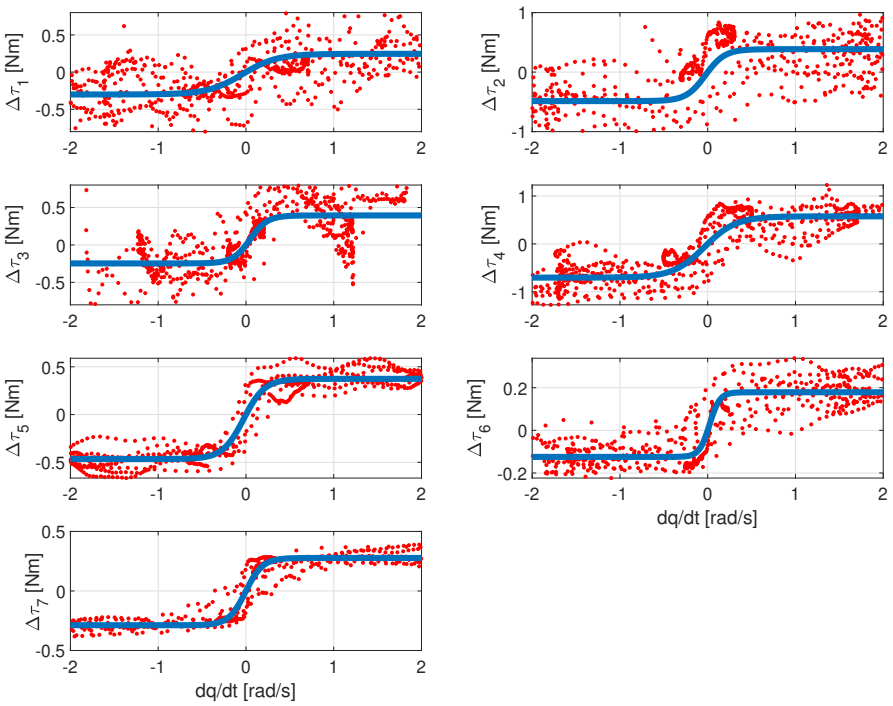


Figure 7.3 – Joint friction estimates. The red dots are the differences $\Delta\tau_j = \tau_j - \hat{\tau}_j$ for each joint j, while the blue lines are the sigmoidal fitting functions. The effect of friction is clear for joints $4 \dots 7$, while its contribution to the total joint torque is slight for the other joints.

A feasible set of dynamic parameters $\hat{\boldsymbol{p}}$ has been then retrieved and its numerical values are reported in Tabs. 7.4 and 7.5.

In order to validate this set, we inserted the retrieved dynamic parameters in a N-E routine in order to compute the joint torques $\hat{\boldsymbol{\tau}}_{ne}$ necessary to perform a given validation trajectory in the joint space. In particular, we commanded a sequence of sinusoidal trajectories (with different amplitudes and periods) to the joints according to eq. (7.15). We then compared the measured joint torques $\boldsymbol{\tau}$ with the estimations $\hat{\boldsymbol{\tau}}_{ne} + \hat{\boldsymbol{\tau}}_f$: the result of this comparison is reported in Fig. 7.4, showing good results.

7.5.1 Validation on a physics simulator

In order to validate the set of identified parameters reported in Tables 7.4 and 7.5, we built a robot model in CoppeliaSim [75], as shown in Figure 7.5, from which we subsequently derived the simulator presented in Chapter 8 . We adjusted the parameters for each link, in order to be compatible with the CoppeliaSim interface (e.g., for each link, the inertia tensor in CoppeliaSim has to be expressed w.r.t.

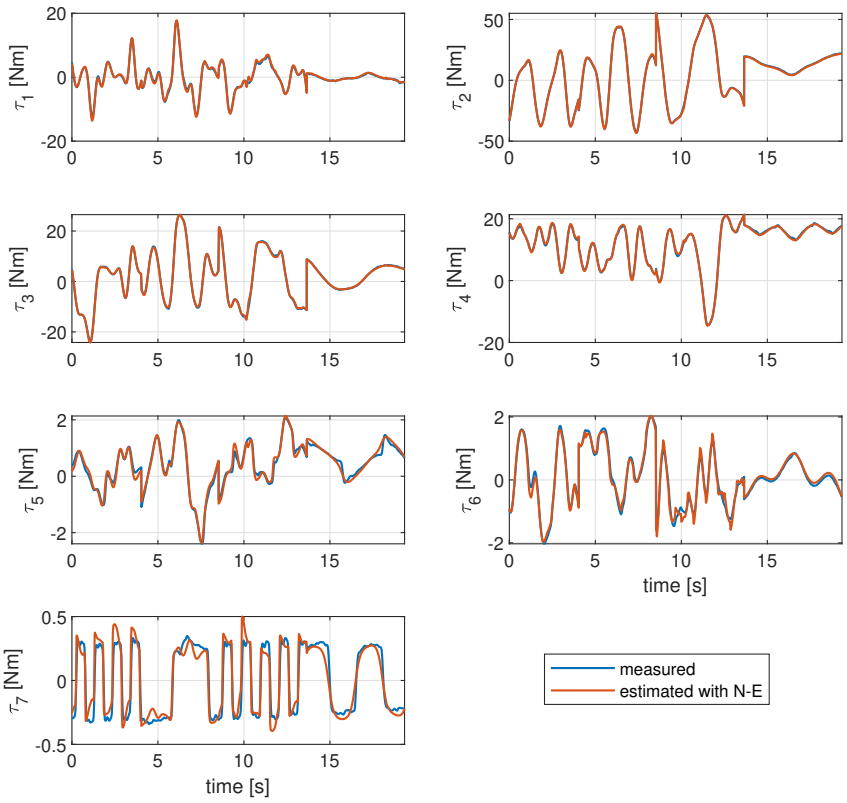


Figure 7.4 – Comparison between the torques $\hat{\boldsymbol{\tau}}_{ne}$ (including friction $\hat{\boldsymbol{\tau}}_f$) from the N-E dynamic model and the ones retrieved from the the joint torque sensors. The overlapping of two signals shows the quality of the obtained parameters estimation $\hat{\boldsymbol{p}}$.

the given CoM reference frame).

The validation process follows the idea that if the identification of the robot dynamic parameters is satisfactory, the measured torques on the real robot, apart from friction which is not simulated in CoppeliaSim, should match the ones provided by the physics engines of the simulator when performing the same trajectory.

In particular, CoppeliaSim provides four different free licence physics engines. A performance comparison of these engines is reported in Figs. 7.6 and 7.7. Both the real and the simulated robot are commanded through sinusoidal joint velocity inputs (spanning their maximum available range). Figure 7.6 shows, for each joint, a filtered version of the measured torques of the real robot (solid blue lines) vs. those provided by the different physics engines (light blue: Bullet 2.78, orange: Bullet 2.83, yellow: ODE, and purple: Newton). As it can be seen, the simulated torques for all engines have a good overlap with the measured ones. Figure 7.7 shows the mean and the standard deviation of the torque errors for each joint.

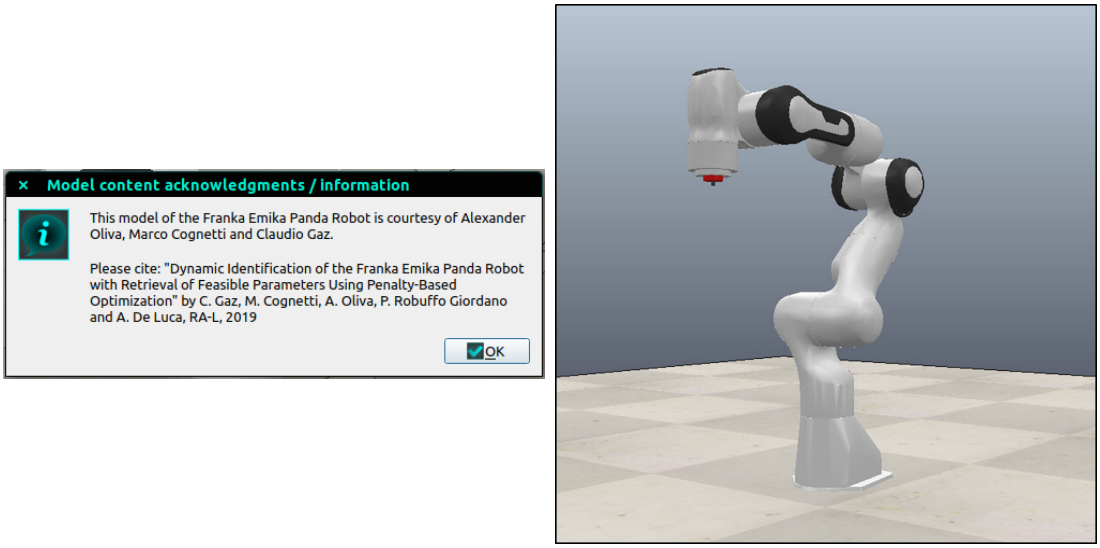


Figure 7.5 – The Franka Emika Panda robot in CoppeliaSim. The released model is available by default since the version V4.0.0.

This error is defined as the difference between the measured torque on the real robot and the one generated from the simulation. We can notice that the mean torque errors are close to zero while the peaks of the standard deviations are due to measurement noise on the real robot torque readings. In our experiments, all engines showed a good numerical performance, as confirmed by the values in Table 7.3, and looking at the results in joint space, ODE (although it is the most noisy among the engines) and Bullet 2.83 seem to be the best performing engines in CoppeliaSim.

Table 7.3 – Mean values of the torque errors at the robot joints for each physics engine available in CoppeliaSim.

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Joint 7	units
Bullet 2.78	0.0554	1.7616	-0.7363	-0.0111	0.0735	0.0970	-0.0372	N·m
Bullet 2.83	0.0508	0.5735	-0.2617	-0.0844	0.0698	0.0628	-0.0386	N·m
ODE	0.0474	0.1782	-0.0986	-0.1493	0.0795	0.0397	-0.0386	N·m
Newton	0.1091	-0.8096	-0.1440	-1.4133	0.0989	0.0027	-0.0255	N·m

To confirm the quality of the proposed solution, we analyzed the performance also in Cartesian space. To this aim, we compared the 3D pose (as obtained from CoppeliaSim) of the end-effector of the simulated robot with the corresponding pose obtained by reading the joint positions from the real robot and passing them to the direct kinematics module. We defined then a position and an orientation error (following the Roll-Pitch-Yaw convention), whose mean values are reported

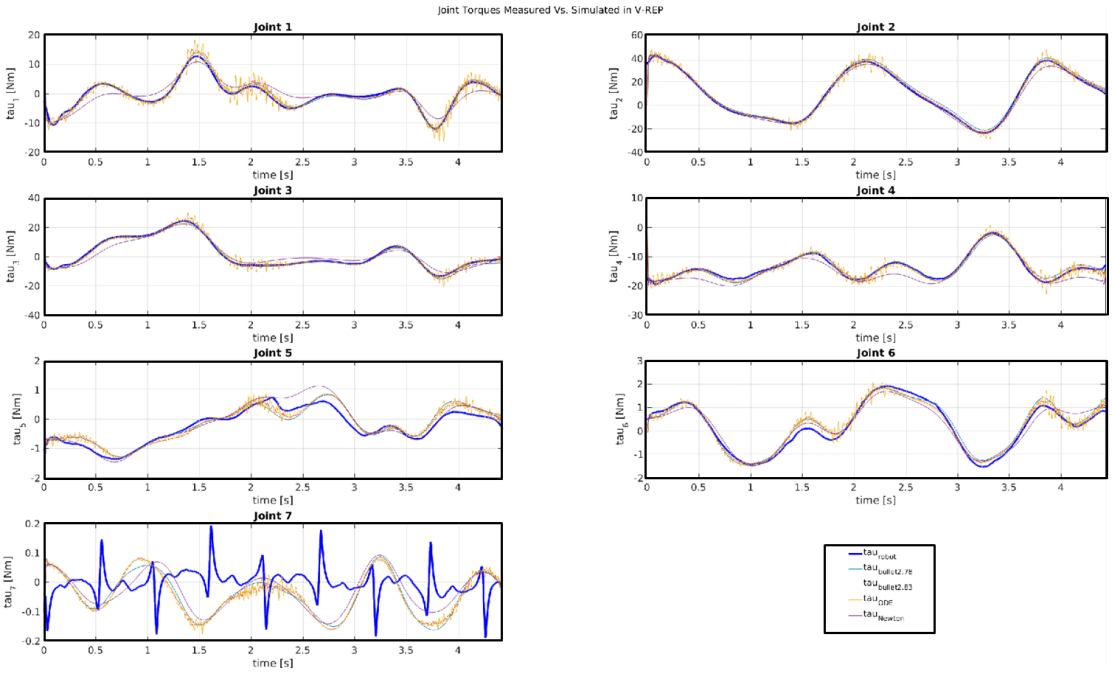


Figure 7.6 – Comparison between the torques generated by the real robot during an experiment (blue) and those obtained from the simulated Panda robot in CoppeliaSim using different physics engines (Bullet 2.78, Bullet 2.83, ODE, Newton).

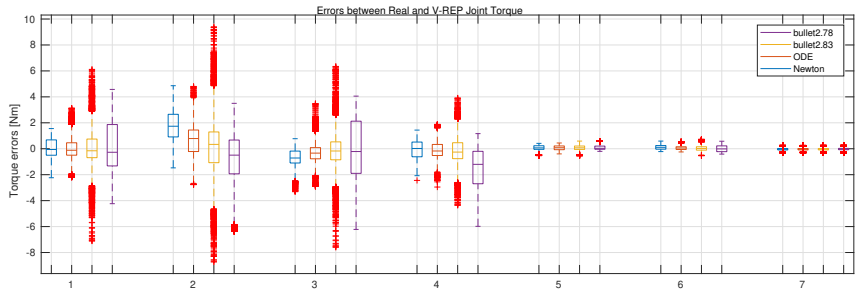


Figure 7.7 – Torque errors (w.r.t. the real robot) of the physics engines Bullet 2.78, Bullet 2.83, ODE, and Newton in CoppeliaSim during an experiment with the Panda robot for each of its 7 joints. The boxes indicate mean values of the errors while the bars denote their standard deviations.

in Fig, 7.8 for all physics engines available in CoppeliaSim. This comparison is shown also in the video accompanying [8] (https://ieeexplore.ieee.org/document/8772145/media#media).

The results support once again the quality of the estimated dynamic parameters, since the mean position error is below 1.5 cm and the mean orientation error is below 3 degrees for all the engines. Particularly good results are obtained with the ODE engine, with mean errors of less than 5 mm in position and below 1.2 deg in orientation. Moreover, the maximum errors for the worst performing

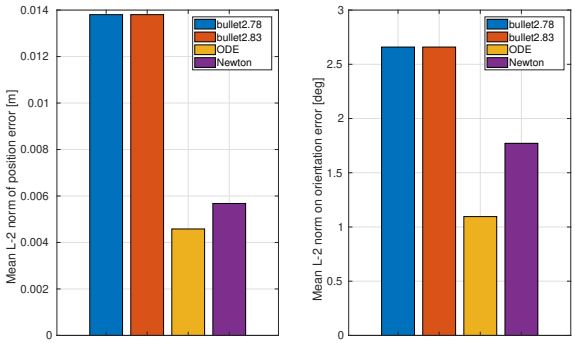


Figure 7.8 – The position (left) and orientation (right) mean error for the different physics engines available in CoppeliaSim. While all the physics engines show a good performance, ODE seems to outperform the others.

physics engine (Bullet 2.78) are 9.13 cm in position and 3.14 deg in orientation. Notice that the velocity command in CoppeliaSim was given in open-loop. Furthermore, the simulation step was set to 20 ms in order to run faster simulations albeit leading to greater integration errors in the physics engines solver. Better performance can be achieved with a smaller simulation step as we will see in the results of the next chapter.

7.6 Summary

In this chapter we addressed the problem of extracting a feasible set of parameters that characterizes the dynamics of the robot. We identified the dynamic coefficients through a standard least squares algorithm by means of a reverse engineering approach. Thanks to an improved version of the algorithm proposed in [20], we retrieved a set of feasible parameters by solving a nonlinear optimization problem, taking into account several constraints including the physical bounds on the dynamic parameters (such as the total mass of the robot) and the triangle inequalities of the link inertia tensors. The proposed framework, was validated by deriving the Lagrangian model of the Franka Emika Panda robot and testing it through experiments. We also validated the extracted feasible set of dynamic parameters using a Newton-Euler routine [3]. To the best of the authors' knowledge, this was the first work that retrieves the dynamic coefficients and a feasible parameter set for the Panda robot. A major feature of the proposed framework is that it can be easily modified to include further (possibly) nonlinear

[3]The dynamic identification procedure presented in this chapter has been performed accordingly to a reverse engineering approach. In the Appendix A, however, the reader can find also the results of the classical identification approach.

constraints (e.g., based on *a priori* information on the robot). We have released both the dynamic model and the parameters retrieval framework as open-source code, so that they are available to the robotics community at the following website: https://github.com/marcocognetti/FrankaEmikaPandaDynModel.

The results presented in this chapter are issued in the paper:

- C. Gaz, M. Cognetti, **A. Oliva**, P. Robuffo Giordano, and A. De Luca, "Dynamic identification of the Franka Emika Panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Robotics and Automation Letters*, 2019

The supplementary material and accompanying video is reachable at https://ieeexplore.ieee.org/document/8772145/media#media.

Table 7.4 – Lower and upper bounds for masses and centers of mass of the Panda robot and their corresponding estimated values.

parameter	LB	UB	$\hat{p}$	units
m_1	0	10	4.970684	kg
m_2	0	10	0.646926	kg
m_3	0	10	3.228604	kg
m_4	0	10	3.587895	kg
m_5	0	10	1.225946	kg
m_6	0	10	1.666555	kg
m_7	0	10	7.35522e-01	kg
c_{1x}	-0.05	0.05	3.875e-03	m
c_{1y}	-0.05	0.05	2.081e-03	m
c_{1z}	-0.4	0.05	*	m
c_{2x}	-0.05	0.05	-3.141e-03	m
c_{2y}	-0.15	0.05	-2.872e-02	m
c_{2z}	-0.05	0.05	3.495e-03	m
c_{3x}	-0.05	0.15	2.7518e-02	m
c_{3y}	-0.05	0.05	3.9252e-02	m
c_{3z}	-0.1	0.05	-6.6502e-02	m
c_{4x}	-0.15	0.05	-5.317e-02	m
c_{4y}	-0.05	0.15	1.04419e-01	m
c_{4z}	-0.05	0.05	2.7454e-02	m
c_{5x}	-0.05	0.05	-1.1953e-02	m
c_{5y}	-0.05	0.05	4.1065e-02	m
c_{5z}	-0.05	0.05	-3.8437e-02	m
c_{6x}	-0.05	0.15	6.0149e-02	m
c_{6y}	-0.05	0.05	-1.4117e-02	m
c_{6z}	-0.05	0.05	-1.0517e-02	m
c_{7x}	-0.05	0.05	1.0517e-02	m
c_{7y}	-0.05	0.05	-4.252e-03	m
c_{7z}	0.04	0.15	6.1597e-02	m

Table 7.5 – Lower and upper bounds for the inertia tensor elements of the Panda robot and their corresponding estimated values.

parameter	LB	UB	$\hat{p}$	units
I_{1xx}	0	1	7.0337e-01	kg·m^2
I_{1xy}	-1	1	-1.3900e-04	kg·m^2
I_{1xz}	-1	1	6.7720e-03	kg·m^2
I_{1yy}	0	1	7.0661e-01	kg·m^2
I_{1yz}	-1	1	1.9169e-02	kg·m^2
I_{1zz}	0	1	9.1170e-03	kg·m^2
I_{2xx}	0	1	7.9620e-03	kg·m^2
I_{2xy}	-1	1	-3.9250e-03	kg·m^2
I_{2xz}	-1	1	1.0254e-02	kg·m^2
I_{2yy}	0	1	2.8110e-02	kg·m^2
I_{2yz}	-1	1	7.0400e-04	kg·m^2
I_{2zz}	0	1	2.5995e-02	kg·m^2
I_{3xx}	0	1	3.7242e-02	kg·m^2
I_{3xy}	-1	1	-4.7610e-03	kg·m^2
I_{3xz}	-1	1	-1.1396e-02	kg·m^2
I_{3yy}	0	1	3.6155e-02	kg·m^2
I_{3yz}	-1	1	-1.2805e-02	kg·m^2
I_{3zz}	0	1	1.0830e-02	kg·m^2
I_{4xx}	0	1	2.5853e-02	kg·m^2
I_{4xy}	-1	1	7.7960e-03	kg·m^2
I_{4xz}	-1	1	-1.3320e-03	kg·m^2
I_{4yy}	0	1	1.9552e-02	kg·m^2
I_{4yz}	-1	1	8.6410e-03	kg·m^2
I_{4zz}	0	1	2.8323e-02	kg·m^2
I_{5xx}	0	1	3.5549e-02	kg·m^2
I_{5xy}	-1	1	-2.1170e-03	kg·m^2
I_{5xz}	-1	1	-4.0370e-03	kg·m^2
I_{5yy}	0	1	2.9474e-02	kg·m^2
I_{5yz}	-1	1	2.2900e-04	kg·m^2
I_{5zz}	0	1	8.6270e-03	kg·m^2
I_{6xx}	0	1	1.9640e-03	kg·m^2
I_{6xy}	-1	1	1.0900e-04	kg·m^2
I_{6xz}	-1	1	-1.1580e-03	kg·m^2
I_{6yy}	0	1	4.3540e-03	kg·m^2
I_{6yz}	-1	1	3.4100e-04	kg·m^2
I_{6zz}	0	1	5.4330e-03	kg·m^2
I_{7xx}	0	1	1.2516e-02	kg·m^2
I_{7xy}	-1	1	-4.2800e-04	kg·m^2
I_{7xz}	-1	1	-1.1960e-03	kg·m^2
I_{7yy}	0	1	1.0027e-02	kg·m^2
I_{7yz}	-1	1	-7.4100e-04	kg·m^2
I_{7zz}	0	1	4.8150e-03	kg·m^2

Chapter 8

FrankaSim

Contents

8.1 Introduction

Software simulators are valuable proof-of-concept tools for validating methods, theories and ideas in a simple and cost-effective way. They are useful for testing and validating ideas and algorithms that would otherwise be hard to implement due to limited resources or restrictions [101]. For instance, recently robot simulators are being exploited for generating large amounts of training data for training neural networks allowing to reach training levels that would be impossible with data collected on real hardware [102, 103].

Any robotic simulator must be able to simulate different kinds of robots, actuators and sensors with enough accuracy for closing the reality gap. The most common robotic system that is featured by virtually any robotic simulator is a manipulator arm for tasks involving motion control, pick-and-place, interaction with the environment, and so forth. A correct dynamic modeling of a manipulator is of paramount importance in relation to problems of motion simulation, analysis of manipulation structures and control algorithms. Simulating the motion of a manipulator allows indeed to safely test control strategies and trajectory planning techniques in a low-risk environment without the need to refer to a physically available platform. Modern robotics simulators are flexible and dynamic and can handle complex physics simulations as well as disparate sensors that a user may need to evaluate. In this respect, the goal of this work is to propose a *high-fidelity* and *openly available* simulator for a popular manipulator arm, the Panda robot manufactured by Franka Emika, with in addition, as case study, the integration and exploitation of the simulator for achieving visual-servoing tasks [44]. Indeed, visual-servoing is a very popular set of techniques for controlling the motion of a robot from visual input provided by one or more cameras, and in this chapter we show how an accurate dynamical simulation of the Panda robot can be exploited to perform realistic and non-trivial visual servoing schemes.

The robot considered in this work is, as the name of the simulator suggest, the Panda robot by Franka Emika. For more details on the robot refer to Section 7.2.

8.2 Related Works

The manufacturer has not released, to the best of our knowledge, the official dynamic parameters of the Panda robot and before the recent identification results reported in [8], no parameters were publicly available. To overcome this drawback, a first attempt to obtain an estimation of the dynamic parameters of the Panda was made in [104] by taking the original URDF description files and the robot

meshes from the official repository and adding the missing gazebo-specific tags and inertial parameters (mass, center of mass and inertia tensor) recovered from a CAD-based software. Once the identified dynamic parameters became available, some researchers started building their own simulators from the same repository and tweaking the URDFs [105, 106] to achieve more realistic simulations both in Gazebo and MuJoCo [107] respectively. In [108], the authors built a Panda simulation platform using Simscape Multibody MATLAB Toolbox based on the kinematic, dynamic and friction models identified in [8]. Simscape is neither open nor free, and the developed code for the simulation cannot be transferred to real implementations.

As reported in the surveyed literature in [101], the most important criterias for choosing a simulator are: **small reality gap** (32%), **to be open-source** (24%), **light & fast** (11%), **simulation-to-real code transfer** (9%), **customization** (6%), **others** (5%), **no inter-penetration between bodies** (3%). Despite Gazebo is the most used simulator in research, CoppeliaSim (formerly V-REP) [75] scores the best in the previous criteria [101] while [109] pointed out that it offers a number of useful features, such as multiple physics engines (Bullet, ODE, Vortex and Newton), a comprehensive model library, the ability of a user to interact with the world during simulation and, most importantly, mesh manipulation and optimisation. Moreover, it automatically spawns new threads on multiple CPU cores and therefore utilises the full amount of CPU power when necessary. It is therefore suitable for high-precision modelling although it is the most demanding in terms of resources among the compared simulators. In the reality gap analysis conducted in [110] on a grasping task, CoppeliaSim with Newton and Vortex performed 1st and 3rd respectively

Most of the reviewed works are based on Gazebo, which has some limitations compared to CoppeliaSim which, despite its higher demand on resources, is more accurate and faithful to reality. To the best of our knowledge, the simulator proposed in this work is the only dynamic simulator supporting CoppeliaSim. Besides offering the characteristics of the other Panda simulators in terms of kinematics, dynamic model parameters and ROS integration, this simulator further offers full integration within the ViSP library [10], enabling it with visual-servoing capabilities. There exist some other libraries allowing for rapid prototyping of visual-servoing applications [111, 112], but probably ViSP is the most complete open-source visual-servoing library.

The proposed simulator features a Lagrangian Dynamic model library that can provide the Coriolis matrix of the manipulator and the estimated joint dynamic friction, unlike dynamic models based on standard Newton-Euler algorithms. Fi-

nally, being the code of the simulator implemented as a replica of the ViSP class that wraps the *libfranka*, it greatly facilitates the possibility of transfer verified code to the real robot. This simulator lends itself well to simulate Dynamic Visual-Servoing [46] or Vision-Force control schemes [6, 55, 57], relieving the user from the burden of implementing his/her own simulation platform.

8.3 The simulator: FrankaSim

The simulator proposed in this work, is a co-simulator that includes several conveniently integrated components allowing for rapid prototyping and testing of manipulator controllers. It is mainly based on ViSP code that provides a bulk of utilities such as native support to linear algebra and transformation matrices, among others, greatly simplifying both the simulator implementation itself as well as the deployment of user defined controllers. ViSP further extends the simulator control skills by enabling it with visual servoing control capabilities. A physical simulation is performed in CoppeliaSim, in which one (or more) instance(s) of a Panda robot can be used, with its identified dynamic model parameters [8]. The communication between CoppeliaSim and the C++ instance of a simulated robot takes place through ROS. All the software presented in this work, the CoppeliaSim scenes and models as well as different experiments, for both the simulated and real robot, are available on `https://github.com/lagadic/visp_ros` while step-by-step tutorials can be found in the ROS wiki `https://wiki.ros.org/visp_ros`.

A high-level view of the architecture and operation of the simulator software will be provided below, after having detailed the various components.

8.3.1 Kinematics

As we have shown in Chapter 2, a manipulator can be mechanically described as a kinematic chain consisting of rigid bodies (links) connected by means of revolute or prismatic joints, which constitute the DoF of the structure. One end of the chain is bounded to the floor while at the other extreme a tool is usually attached (see Fig. 8.1). The kinematic description of the Panda according to its Denavit-Hartenberg parameters was shown in Section 2.2.

In this subsection, under the term *kinematics* we encompass either direct, inverse and differential kinematics (see Chapter 2 for further details).

In FrankaSim, we delegate the issue of dealing with kinematics to the well-known kinematics and dynamics library OROCOS_KDL [113]. The Orocos project aims at providing kinematic and dynamic code usable in real time; it

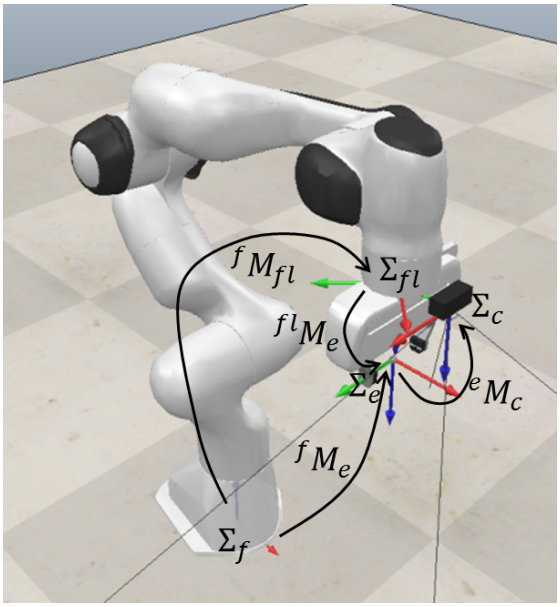


Figure 8.1 – An instance of the Panda model in CoppeliaSim with its gripper and a wrist mounted camera. The floor Σ_f, flange Σ_{fl}, end-effector Σ_e and camera Σ_c frames are depicted as well as some transformation matrices between them.

contains code for rigid body kinematic representation and calculations for structures and their direct and inverse kinematic solvers.

8.3.2 Dynamics

The dynamic model of a manipulator provides a description of the relationship between the actuation torques at the joints and the motion of the structure. The Lagrange formulation of the equation of motion has been presented in Section 2.3.

In the previous chapter, an accurate dynamic model of the robot has been identified and a set of feasible dynamic parameters retrieved. These parameters have been adopted in the MATLAB Robotics toolbox[1] [114] alongside many handmade URDFs of other simulators [105, 106] and, more recently, were also included in the official URDF model provided by the manufacturer[2], making our identified model, *de facto*, the standard model.

From the dynamic parameters identified in [8], the authors released both

[1] https://github.com/petercorke/robotics-toolbox-matlab/blob/master/models/mdl_panda.m

[2] https://github.com/frankaemika/franka_ros/blob/develop/franka_description/robots/panda_gazebo.xacro

a MATLAB and C++ libraries as open-source software[3] under GPLv3 license and a CoppeliaSim model of the Panda (arm only, see Figure 8.1). These libraries provide an estimate of the dynamic terms of the Lagrangian model (2.27) $(\hat{\boldsymbol{B}}(\boldsymbol{q}), \hat{\boldsymbol{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}}), \hat{\boldsymbol{g}}(\boldsymbol{q}), \hat{\boldsymbol{\tau}}_f(\dot{\boldsymbol{q}}))$ computed from the symbolic expressions of the equation of motion and identified parameters. The main limitation of this library is that the model was identified until the flange of the Panda arm, making it useless if one wants to attach a payload on it, *e.g.* the Panda Hand gripper, for which we have also provided a model for CoppeliaSim within the simulator package (see Fig 8.2).

In the work presented in this chapter, we have further extended this library, parameterizing it with respect to both the payload parameters, as reported in [83], and the gravitational acceleration vector. This way the library can provide the same features offered by Newton-Euler (N-E) algorithms, *i.e.* adding payloads to the chain or mounting the robot in a position different from vertical, while providing the full Coriolis matrix and the viscous friction terms. This constitutes an advantage w.r.t. standard N-E algorithms since they are not able to provide neither the Coriolis matrix $\hat{\boldsymbol{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ -or any of its columns- nor the mixed velocity term $\hat{\boldsymbol{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}_r$, with $\dot{\boldsymbol{q}}_r$ a reference velocity different from the one used to compute $\hat{\boldsymbol{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ [115], but only the resulting vector $\hat{\boldsymbol{c}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \hat{\boldsymbol{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$. These matrices are needed, for instance, to evaluate the residual vector for collision detection and safe reaction applications [116] or to implement passivity-based trajectory tracking control laws [117]. Although [115] has proposed a modified N-E algorithm to overcome the aforementioned limitation of the standard N-E method, it is not implemented in OROCOS_KDL nor it is provided by the *libfranka*. This motivates the use of our library both to avoid implementing this method in OROCOS_KDL from scratch and because this library can be easily integrated within the real robot controller code without having the dependency of the KDL library.

To add a tool to the real robot, one has to pass through the Panda's Desk web interface to specify both the new pose of the end-effector in flange frame (Σ_{fl}) and the dynamic parameters. Figure 8.2 reports those values for the real gripper. Due to possible simulation malfunctioning with some physics engines, such as Bullet and ODE mainly, when the difference between connected masses/inertias is too large it is convenient to split the total mass of the gripper among the fingers and the body uniformly, in order to avoid "strange" behaviours. By doing so, the total mass will stay the same but the inertia tensor and the location of the center of

[3] https://github.com/marcocognetti/FrankaEmikaPandaDynModel

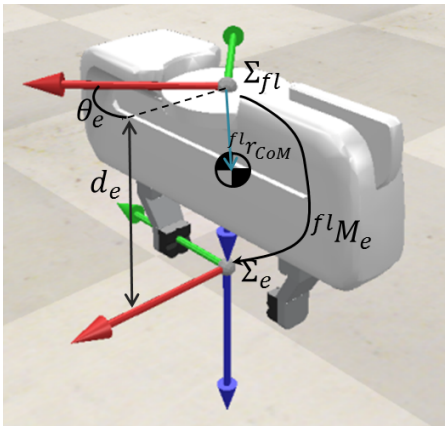


Figure 8.2 – CoppeliaSim model of the Franka Hand gripper. End-effector position ${}^{fl}\boldsymbol{r}_e = [0;0;d_e]^\top\,[m]$ and orientation ${}^{fl}\boldsymbol{R}_e = R_z(\theta_e)$ and CoM position ${}^{fl}\boldsymbol{r}_{CoM} = [-0.01;0;0.03]^\top\,[m]$ in flange frame. Mass $m_L = 0.73\,[kg]$ and inertia tensor $\boldsymbol{I_L} = diag(0.001;0.0025;0.0017)\,[kg.m^2]$. $\theta_e = -45°, d_e = 0.1034[m]$.

mass of the overall gripper will differ from the real one. This is not an issue since, as we will see in Section 8.3.5, we automatically gather this information from the simulator and send it to the robot through ROS.

8.3.3 ViSP

ViSP[4] or Visual Servoing Platform, is a modular C++ library developed and maintained by the Rainbow team (former Lagadic) at Inria/IRISA Rennes, France, that allows for fast development of visual servoing applications. This library was designed to be hardware independent, simple, portable and easily extendable. Furthermore, it features a large class of elementary tasks with various visual features ranging from image points, to image lines and moments, 3D pose estimation and so forth, as well as providing native support for linear algebra operations, visual trackers, plotters, and of course, visual-servoing controllers.

There is already a class in ViSP that wraps the *libfranka* robot API named *vpRobotFranka*. This is a multi-thread implementation that runs the robot control routine in a separated thread at the control rate (1 kHz) allowing the main program to continue its own execution; this means that from the same *main()*, one can read and process information coming from other sensors, like cameras and force/torque sensors, even at slower sampling rates. This implementation allows to simultaneously control more than one robot at a time.

[4] https://visp.inria.fr

8.3.4 Visp_ros

Visp_ros is a basket of generic ROS nodes that are based on the ViSP library and exposes it to the ROS ecosystem. It contains a C++ library of classes that allows using ROS with transparency and could be used like usual ViSP classes without the need to write ROS specific code. The nodes developed in Visp_ros exploit all the potential offered by both ViSP and ROS.

FrankaSim is developed as part of Visp_ros. The simulator *vpRobotFrankaSim* class replicates the class *vpRobotFranka* present in ViSP for the real robot, providing the same methods, and extending it with few others to keep the same interface and behavior on both the simulator and with the real robot. The ROS-specific code is implemented in the *vpROSRobotFrankaCoppeliasim* class which inherits from class *vpRobotFrankaSim*. It also contains CoppeliaSim-specific callbacks - passing through ROS- to manage e.g. the synchronization between CoppeliaSim and the C++ simulation code, the setting of the control mode. This is also a multi-threaded implementation featuring, other than the *main()* process in which the robot instance is declared, a "reading" thread, listening at the topic containing the robot state $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau})$, and a "writing" one, that continuously publishes the joint velocity $\dot{\boldsymbol{q}}_{cmd}$ or torque $\boldsymbol{\tau}_{cmd}$ commands depending on the selected control mode. Specific methods to get the simulation time from CoppeliaSim, for adding a tool, e.g. the Panda Hand, as well as internal flags to select a synchronous or asynchronous simulation further extend the set of methods of the *vpROSRobotFrankaCoppeliasim* class. Preserving the same interface for the simulator classes compared to the one that controls the real robot allows code reusability.

8.3.5 CoppeliaSim

CoppeliaSim is a versatile and scalable simulation framework offering a multitude of different programming techniques for the controllers, and allows to embed control functionalities in simulation models easing the programmers effort and reducing the deployment complexity. It integrates four free physics engines (Bullet 2.78 & 2.83, ODE, Newton) and a more precise closed source engine (Vortex), for which it is possible to obtain a free non-commercial license. It also provides a large variety of ready to use sensors and support scripting via Lua. Using a Lua script attached to the robot, we have automated some procedures such as finding the dynamic and kinematic parameters of the load (gripper), retrieving the absolute gravity vector in base frame or the choice of the control mode.

Thanks to the RosInterfaceHelper Lua script, it is possible to automatically start/stop the CoppeliaSim scene simulation as well as to synchronize each simu-

lation step between the C++ code and the physical simulation. In order to make the synchronization more transparent to the user, the triggers were embedded in the *vpROSRobotFrankaCoppeliasim* class, and a flag has to be selected to enable it.

Within the simulator we deployed the scenes of the experiments, the model of the Panda Hand gripper (see Figure 8.2) and some AprilTags.

8.3.6 Software Architecture

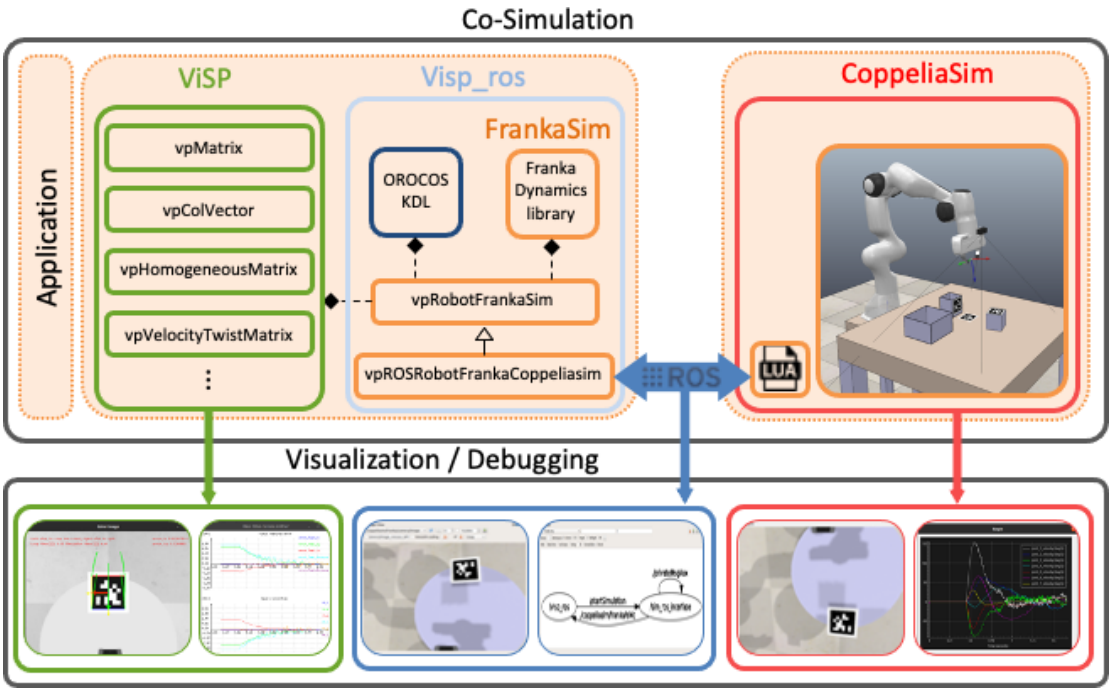


Figure 8.3 – Block diagram of the main components of the FrankaSim architecture. An UML-like diagram highlights the principal class relationships while a wide range of visualization and debugging possibilities at ViSP (green), ROS (blue) and CoppeliaSim (red) levels is depicted.

A block diagram of the co-simulation environment with some of the C++ classes and dependencies in a UML-like diagram is shown in Figure 8.3. The diagram is complemented with some data visualization utilities provided by the various software components that help the user in analyzing the system performance and/or debugging.

A simulation is built as a ROS node that benefits from both FrankaSim and ViSP capabilities to implement one or multiple Panda robots controllers. The scene rendering and the physics engine are supported by CoppeliaSim. Communication between FrankaSim and CoppeliaSim is performed using ROS communication level. The ROS node could be seen as a *main()* that firstly initializes

the simulation (e.g. establishing robot and camera connections, initialize position and so on) and then enters an infinite *while* loop. At each iteration of the loop, from the robot state measurements (position, velocity or torque) and, potentially any other sensor measurement like an image acquired by a virtual camera, a new joint/Cartesian velocity/torque command is computed and sent back to CoppeliaSim to update the scene rendering. There exists a mechanisms that allows to perform a synchronous execution between the simulation step in CoppeliaSim and the control program. Typically, for a visual servoing simulation, one iteration of the loop could be synchronized at 20 ms (like for a real camera), while for impedance control it is recommended to reduce this time step between 1 and 3 ms (the simulation will be more accurate with shorter time steps but will last longer). FrankaSim has been designed in such a way as to hide ROS from the user, making its use transparent. This approach makes it very easy to prototype an application in simulation first, and then to deploy it on a real robot by just changing few lines of code, as we will see in Section 8.4.1.

8.4 Experiments

In this section we present two experiments to evaluate on the one hand the reality gap between the simulation and the real hardware both from the implementation and the obtained results point of view and, on the other hand, to show the great simulation potential of the proposed package.

8.4.1 Single-Arm Experiment: Real vs Simulated

In this experiment we consider the following joint impedance controller

$$\boldsymbol{\tau}_t = \boldsymbol{B}(\boldsymbol{q})\left(\boldsymbol{K}\boldsymbol{e} + \boldsymbol{D}\dot{\boldsymbol{e}} + \boldsymbol{I}\int \boldsymbol{e}\,dt + \ddot{\boldsymbol{q}}_d\right) + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_f(\dot{\boldsymbol{q}}) - \boldsymbol{\tau}_0\, e^{-\mu t} \tag{8.1}$$

with $\boldsymbol{\tau}_t$ the joint torque command at time t, $\boldsymbol{q}_d(t)$ a desired joint trajectory, $\boldsymbol{e} = \boldsymbol{q}_d - \boldsymbol{q}$, and $\boldsymbol{K}$, $\boldsymbol{D}$, $\boldsymbol{I}$ are the proportional, derivative and integral diagonal gain matrices respectively. The exponential term allows for a smooth start of the commands making the computed joint torques start from zero having set $\boldsymbol{\tau}_0 = \boldsymbol{\tau}_t(0)$ equal to the computed torque at $t = 0$ while μ determines the decay rate. A Panda arm equipped with its gripper and a camera attached to the flange is controlled using (8.1). Figures 8.4 and 8.5 compare the joint positions and torques measured on a real Panda arm versus a simulated one using FrankaSim while applying a desired sinusoidal joint trajectory on joints 1, 3 and 4. As

expected, the obtained results on the real robot are noisier, but perfectly follow those obtained in simulation. In the video related to this work we show also a PBVS and an IBVS experiment with an AprilTag target where the visual-servo behaviors are very close between the real and simulated cases. More than 95% of the C++ code is common between real and simulated experiments. Only about ten lines of code are simulator-specific to initiate communication with the robot and the camera if the latter is used.

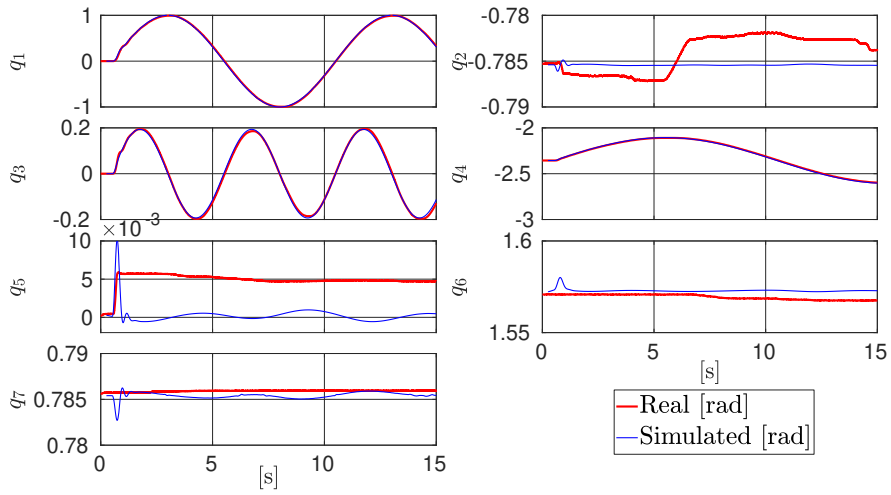


Figure 8.4 – Real vs Simulated measured joint positions while tracking a desired joint trajectory using controller (8.1). The total joint position Root Mean Square Error (RMSE) along the whole trajectory ($[0.5; 15]$ [s]) for all the joints is $RMSE_q = 0.006$ [rad].

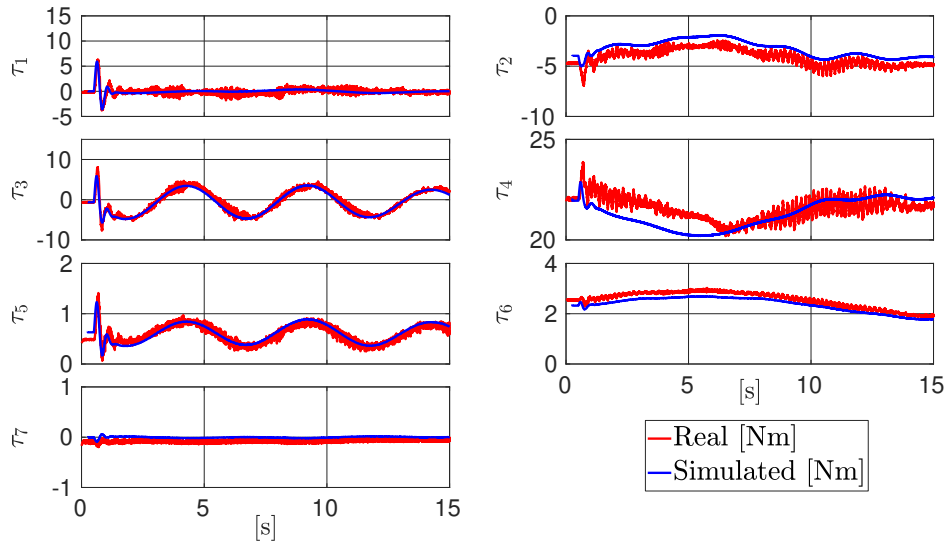


Figure 8.5 – Real vs Simulated measured joint torques while tracking a desired joint trajectory using controller (8.1). The total RMS torque error along the whole trajectory ($[0.5; 15]$ [s]) for all the joints is $RMSE_\tau = 0.5546$ [Nm].

8.4.2 The Dual-Arm Experiment

This example was designed to showcase together most of the salient features of the simulator, *i.e.*, the possibility to handle multiple robots in the same program, the ability to control the robot in velocity or in torque, the capacity of the presented Lagrangian dynamic model library to fully compensate for the gravity and the payload (gripper) even when not vertically mounted, the synchronous execution with the physics simulator, as well as simulating a lower camera rate and a Visual-admittance Pose-Based Visual-Servoing in interaction with the environment.

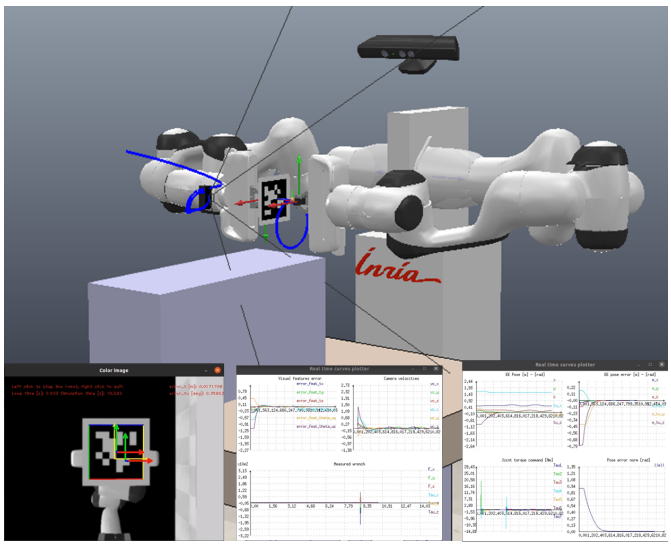


Figure 8.6 – Dual-Arm Experiment: two Panda robots constitute the arms of "Franko". The torque controlled left arm holds an AprilTag while following a circular trajectory. The right arm implements an Extended External Hybrid controller to follow the AprilTag while dealing with the collision with the environment. The circular and "D" shaped paths of both arms are visible (blue lines). The camera view and some plotted curbs are shown in the bottom of the figure.

The simulation platform consists of a dual arm manipulator named "Franko", equipped with two Panda arms with their grippers and a wrist mounted camera on the right arm. The left arm is torque controlled; it implements a computed torque Cartesian controller and holds an AprilTag of the family 36h11. The right arm implements an Extended External Hybrid controller [6] to visually follow the AprilTag. At start, both arms are not servoed and the left one is only compensating for the gravitational effects (both for the arm and the gripper), then it is commanded to reach a desired pose in the workspace while the right arm is commanded to make sure that the AprilTag is centered in the image at a distance of 20 cm. Once the visual-servoing converges, the left arm starts tracking a circular trajectory with the end-effector, causing the right arm to follow. There is an obstacle on the worktable to which the gripper of the right arm collides with while

chasing the AprilTag, resulting in a "D" shaped trajectory (please see Fig. 8.6 and related video for more in depth and clear understanding of this experiment).

8.5 Summary

In this chapter, we presented our simulator for the Franka Emika Panda robot. The simulator has been designed with most of the criteria of importance to users in mind *i.e., small reality gap, open-source, simulation-to-real code transfer, customization, etc.* It has a number of features that make it unique compared to other dynamic simulators for this robot; in fact, to date, it is the only one to support CoppeliaSim and to integrate a library providing the Coriolis matrix and the estimated dynamic friction as well as being fully integrated into the ViSP ecosystem that enables it with visual servoing capabilities. The simulator is currently being used by researchers in the Rainbow team for prototyping and testing novel sensor-based control strategies, to study second-order visual servoing, and the combined use of vision and force sensing for precision assembly tasks. We hope that providing the code to the entire community will allow increasing the number of end-users and applications addressed.

The work presented in this chapter was extracted from the paper:

- **A. A. Oliva**, F. Spindler, P. Robuffo Giordano, and F. Chaumette, "FrankaSim: A Franka Emika's Panda robot simulator with visual-servoing enabled capabilities," 2022 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. (submitted)

Video: https://www.youtube.com/watch?v=Kxn3pXsK9h4

The video related to this work shows some features of the simulator while an IBVS and PBVS experiments are performed. Then, a complex multi-robot simulation is showcased; the left arm is torque controlled while the right one is commanded in velocity, using our Extended External Hybrid controller, which allows the manipulator to deal with the impacts with the environment. Finally, an experimental comparison between simulated and real robot executing the same trajectory is shown followed by some further experiments carried out along the thesis work.

Part IV Conclusions and future directions

Chapter 9 Conclusion

In this concluding chapter, we want to review the results obtained in this thesis and point out what might be open points that need further attention.

9.1 Summary and contributions

The goal of this thesis was to understand how to effectively combine visual and force information within the same control algorithm. The system considered is a classical serial manipulator equipped with a wrist-mounted 6D force/torque sensor and a monocular camera, regardless of its mounting (i.e., Eye-in-hand or Eye-to-hand). What we have set out to accomplish was a *shared* integration of vision and force sensing along all the controlled directions as well as a general treatment, that was independent of the visual feature. This led us to explore second-order models that relate the motion of the visual features in the image plane to the forces acting on the end-effector of the manipulator. This research allowed us, as shown in Chapter 5, to express the model of the manipulator in the visual feature space and from there derive an impedance controller in the same space. This type of controllers had been little studied in the literature with few recent works which reported their advantages over the classical kinematic visual servoing controllers. Those studies had been conducted to control the motion of the manipulator in free space and their implementations were limited to simulated environments only. Instead, we investigated these controllers and their feasibility for interactions control both through simulations and by implementing them on a real platform. We have proposed an estimator based on the kinematic model of visual servoing to virtually increase the visual information and thus overcome one of the biggest limitations of visual control, namely the low data rate from the camera and latency associated with computer vision algorithms for extracting visual features. This estimator also allows to estimate the velocity of the tracked moving target. The results obtained are very promising; we have demonstrated the feasibility of the practical use of these controllers in tasks of interaction with the environment on the one hand, and on the other we have obtained, with classic low-

cost cameras, tracking performances comparable to those using dedicated high-frequency cameras and specialized hardware.

On the other hand, when one needs to perform manipulation tasks interacting with the environment under static or quasi-static conditions, the dynamics of the manipulator can be neglected, and it might be convenient to rely on a velocity resolved controller, which results much simpler than computed-torque ones. In this regard in Chapter 6 we have proposed what we have called the Extended External Hybrid Vision-Force Control scheme in which, thanks to an admittance law in feature space, the visual reference is modified accordingly with the measured external wrench allowing to achieve compliant motion, including soft behaviours. This controller, due to its hierarchical structure, avoids both inconsistencies at the actuation level and convergence towards local minima. Moreover, given the presence of a force controller on the external loop, it is possible to regulate the exchanged force/torque with the environment to a precise value. We have also shown how this controller can simultaneously deal with both physical and fictitious forces, the latter being generated in the image plane and that can be used, for example, as repulsive force field preventing the visual features to leave the field of view or in collision avoidance tasks to avoid the detected obstacles.

In both proposed controllers, vision and force sensory integration occurs in visual feature space with consequent compliance achieved along the visual features that define the visual task.

The impedance control presented in Chapter 5 imposes a system behavior equivalent to a mass-spring-damper system in the visual feature space due to the fact that, through feedback linearization, it cancels out the manipulator's dynamics and imposes the desired behavior. To do this, one needs to know the model of the manipulator. In this regard in Chapter 7 we proposed a framework for the retrieval of a feasible set of the identified dynamic model parameters of the manipulator and we provided for the first time a set of these parameters for our manipulator as well as an accurate model of the dynamic friction in its joints; this model has become *de facto* the standard model adopted by the scientific community for this manipulator.

From this identification procedure, a C++ library has been created which provides all the dynamic coefficients in the Lagrangian model. We have then further developed this library and a dynamic simulator was developed, reported in Chapter 8, which allowed us to obtain the results shown in Chapter 5. This simulator is very useful to rapidly prototype visual servoing applications and allows to easily switch, by modifying a few lines of code, to the implementation on the real robot.

This thesis work has covered several aspects both theoretical and practi-

cal/implementation on the problem of the coupling of vision and force control for manipulators by providing both the necessary tools for its study and implementation. It has also provided innovative control schemes capable of addressing the different interaction tasks that can be encountered in more or less structured environments, responding therefore to the needs raised by the smart production.

9.2 Open issues and future perspectives

Despite the proposed EKFs track the target's motion very well and provides high-rate visual information, the control bandwidth is still insufficient to impose higher control gains without becoming unstable. These filters allowed torque-level controllers to be implemented on a real platform with low-cost cameras, but the stability margins are limited if compared to those achieved with traditional impedance control. Time domain passivity could be used to monitor and passivize the extra energy that time delays provide to the system as in [118], but directly in the feature space.

Concerning the sensor integration problem, much research has been done in the last three decades, including this thesis, to couple the contact information from a force/torque sensor with the visual information provided by cameras within a control loop. All the approaches surveyed in this thesis, except for some of the constraint-based methods presented in Section 4.5, use a "task frame" concept in one way or another to specify the task, at least the force goals of the task. This is a legacy that carries over from the fact that integrated vision/force control schemes are in essence repurposed motion/force schemes for which the contact modelization have been done based on the assumption that there is a unique contact point between the end-effector and the environment. However, this is not what actually happens in real scenarios. The sensed contact wrench is the aggregate of all the forces/torques acting downstream the wrist-mounted force/torque sensor [119], and the resultant wrench of a contact situation can be indistinguishable from a completely different other that generates the same measured wrench at the sensor frame. A systematic and general contact model which is intuitive, flexible and is able to deal with multiple, complex, and time-varying *motion constraints* between non-polyhedral objects, was developed in [120]. They showed that it is nevertheless possible to reconstruct a complex, multi-point contact situation from the measurements of a force sensor alone, but full identification of 3D contact situations with general and unknown objects is difficult. Humans, nature's most versatile manipulators possess a distributed sense of touch, which makes them particularly adept at manipulating objects, reacting to, or even taking advan-

tage of the different perceived contacts. Furthermore, in a recent study, it was shown that human visual and tactile sensory systems share common features in object recognition processes [121]. It might be then worth investigating the use of distributed force sensors, as suggested in [119], because they can provide richer information about the contact than traditional wrist-mounted force/torque sensors. In this context, the main question posed in this thesis would hold, but with a slight difference: *how to effectively combine vision and distributed force sensing?*

There are many tactile sensors based on different physical principles that are already advanced at commercial level. As for the human, artificial vision and tactile sensing share common characteristics, that are more evident when using a visuo-tactile sensor like gelsight [122]. The *tactile images* can be exploited in a way similar to vision, unifying the processing of both visual and tactile data taking advantage of the use of existing methods in the fields of computer vision and visual servoing for the aspects of perception and control respectively

Not only distributed contact information, that is spatially localized in a small surface (like the fingers) is of interest, but also sparse multiple contacts that can occur on the mechanical structure of the manipulator may be; for instance in situations where one has to carry/manipulate a large and/or too heavy object to lift with only the terminal link; in these cases it could be advantageous to "embrace" the object by making contact on several points between the manipulator and the object (imagine someone carrying a large box under the arm). Further questions arise: *how to servo the robot in order to achieve a desired multi contact configuration?*, which is an active field of research if we think at the in-hand manipulation problem. On the other hand, *how to describe the combined vision/force manipulation task?* would perhaps require more attention.

In addition to multi-contact modelling, one also needs sensors capable of perceiving these contacts. Using only proprioceptive sensors [123, 124] monitor the residual and the robot generalized momentum for contact detection, isolation and identification (a survey [69]); using probabilistic methods, [125] were able to locate the point of contact on one of the manipulator's links with good precision using a particle filter, while with a similar approach [126] performs small exploratory motions once contact was established to locate the contact. In a different way, the use of artificial skins could be valuable in retrieving those contacts, but *how can the knowledge of this multiple contacts be coupled with visual cues in a meaningful manner?* is still an open question.

Appendix A

Identification procedure comparison

A.1 Comparison of dynamic coefficients

The manufacturer does not provide any information on the dynamic parameters of the Panda robot, and we have no ground truth values at our disposal for the dynamic coefficients (a.k.a., *base parameters*) that have to be estimated.

Nevertheless, we have compared the dynamic coefficients obtained with the reverse engineering approach in [8], that we discussed on Chapter 7, with those of a classical identification method [127] that uses measurements of the joint positions and joint torques only. In this case, static positioning is no longer sufficient and the robot must execute sufficiently exciting trajectories in order to allow a reliable estimation of the dynamic coefficients. In particular, following [91], we designed the reference trajectories for the joints $j \in \{1, 7\}$ as

$$q_j(t) = \sum_{l=1}^{L} \frac{a_{l,j}}{l\omega_f} \sin(l\omega_f t) - \frac{b_{l,j}}{l\omega_f} \cos(l\omega_f t) + q_{0,j},$$

where $L = 5$ is the number of harmonics, $\omega_f = 0.15\pi$, and the coefficients $a_{l,j}$, $b_{l,j}$ and $q_{0,j}$ are reported in Table A.1. These trajectories have been commanded to the robot joints during 21 s, collecting a total of 20544 samples of positions and torques per joint.

Since we noticed already that joint friction is not negligible for the robot under study, we added in the robot dynamic model suitable friction functions to each joint $j \in \{1, 7\}$, which are linear in the three parameters $f_{v,j}$, $f_{c,j}$, and $f_{o,j}$ (see eq. (2.34)):

$$\tau_{f,j}(\dot{q}_j) = f_{v,j}\dot{q}_j + f_{c,j}\text{sign}(\dot{q}_j) + f_{o,j}.$$

These functions may suffer from sudden discontinuities in the neighbourhood of $\dot{q}_j = 0$, $j = 1, \ldots, 7$, but are adopted here because of their linearity in the defining parameters.

The results of the identification process is summarized in Table A.2. The first column provides the symbolic expressions of the dynamic coefficients. The

Table A.1 – Trajectory parameters for model identification of the Franka Emika Panda robot.

Joint j	**1**	**2**	**3**	**4**	**5**	**6**	**7**
$a_{1,j}$	-0.2031	-0.0699	0.3076	0.1269	-0.2773	0.2100	-0.2273
$a_{2,j}$	0.1295	-0.5380	0.5864	-0.0253	-0.0857	-0.1194	-0.5636
$a_{3,j}$	-0.0090	-0.2015	-0.4813	-0.2405	0.4810	-0.0950	-0.2099
$a_{4,j}$	0.2319	-0.5535	-0.1228	0.2178	0.5311	-0.0964	0.5725
$a_{5,j}$	-0.7598	0.2352	-0.5273	0.6984	-0.1639	-0.0399	-0.3748
$b_{1,j}$	-0.1136	-0.2437	0.3898	0.0401	0.5491	0.0692	0.2023
$b_{2,j}$	0.6600	0.1820	-0.0268	-0.1503	0.1046	-0.6946	-0.2570
$b_{3,j}$	-0.1858	-0.2390	0.0179	0.2359	0.0407	-0.4442	-0.5010
$b_{4,j}$	0.1867	-0.2647	0.4767	-0.2022	-0.0432	-0.5749	0.0774
$b_{5,j}$	0.2357	0.4252	0.2726	0.3693	0.2448	-0.0698	-0.4833
$q_{0,j}$	-0.5850	-0.1744	-0.3373	-1.8767	-1.0631	1.7917	-0.7284

second and third columns show the values of the coefficients $\hat{\boldsymbol{\pi}}_{\mathrm{R-E}}$ identified using the reverse engineering approach and, respectively, their standard deviations $\%\sigma(\hat{\boldsymbol{\pi}}_{\mathrm{R-E}})$. The regressor condition number is 48.9, while the relative error percentage is 0.0315%. The fifth and sixth columns show the values of the coefficients $\hat{\boldsymbol{\pi}}_{\mathrm{CLS}}$ identified through the classical method with exciting trajectories and, respectively, their standard deviations $\%\sigma(\hat{\boldsymbol{\pi}}_{\mathrm{CLS}})$. In this case, the regressor condition number is 121.88, while the relative error percentage is 1.41%. Finally, the fourth column reports the numerical value of the dynamic coefficients $\boldsymbol{\pi}(\hat{\boldsymbol{p}})$ computed by substituting the values of the dynamic parameters retrieved with our framework [8] (see Chap. 7) in their symbolic expressions. It can be noted that these values are slightly different from the original ones, showing that the ordinary least squares solutions $\hat{\boldsymbol{\pi}}_{\mathrm{R-E}}$ and $\hat{\boldsymbol{\pi}}_{\mathrm{CLS}}$ are usually not physically consistent [91].

On the other hand, to estimate a set of dynamic coefficients which are physically consistent, the following *one-step* approach can be used. It is possible, indeed, to change the term $\phi(\boldsymbol{p}_k)$ of the cost function in the Parameters Retrieval **Algorithm 1** in chapter 7, as:

$$\phi(\boldsymbol{p}_k) = \left\| \overline{\boldsymbol{Y}}\boldsymbol{\pi}(\boldsymbol{p}_k) - \overline{\boldsymbol{\tau}} \right\|^2,$$

where $\overline{\boldsymbol{Y}}$ is the stacked regressor, $\overline{\boldsymbol{\tau}}$ is the stacked measurements vector and $\boldsymbol{\pi}(\boldsymbol{p}_k)$ is the coefficients vector computed from the current parameters vector $\boldsymbol{p}_k$. The drawback of the use of this function ϕ (with respect to the one we reported in chapter 7) is that it is computed from the cumbersome stacked regressor, thus affecting the computational time. The identified dynamic coefficients $\hat{\boldsymbol{\pi}}_{\mathrm{OSI}}$ obtained from the previously described one-step identification procedure are reported in the seventh column of Tab. A.2.

Note that the fourth column of Table A.2 has two missing entries for the eight and twenty-fourth dynamic coefficients: these small coefficients are in fact

Table A.2 – Dynamic coefficients identified using our reverse engineering approach ($\hat{\boldsymbol{\pi}}_{\mathrm{R-E}}$) and using a classical approach ($\hat{\boldsymbol{\pi}}_{\mathrm{CLS}}$), with their standard deviations (in percentage). The coefficients $\boldsymbol{\pi}(\hat{\boldsymbol{p}})$ are computed from the retrieved dynamic parameters $\hat{\boldsymbol{p}}$. Dynamic coefficients $\hat{\boldsymbol{\pi}}_{\mathrm{OSI}}$ are obtained from a one-step identification procedure.

Dynamic coefficients	$\hat{\boldsymbol{\pi}}_{\mathrm{R-E}}$	$\%\sigma(\hat{\boldsymbol{\pi}}_{\mathrm{R-E}})$	$\boldsymbol{\pi}(\hat{\boldsymbol{p}})$	$\hat{\boldsymbol{\pi}}_{\mathrm{CLS}}$	$\%\sigma(\hat{\boldsymbol{\pi}}_{\mathrm{CLS}})$	$\hat{\boldsymbol{\pi}}_{\mathrm{OSI}}$
$J_{2yy}+J_{1zz}$	0.0292166719319167	0.3734649992604760	0.0373379631298541	0.0204364103682226	2.36090168554992	0.0132
$J_{2xx}-J_{2yy}+J_{3yy}+0.0998m_3+0.632c_{3z}m_3+0.1067(m_3+m_4+m_5+m_6+m_7)$	0.9818183628371487	0.0176017697538843	0.9896236345516l5	0.918812119094733	0.120211264877l	0.9485
J_{2xy}	-0.00519124422423026	1.56250607091839	-0.00398364337529688	-0.00328924846223289	11.0391711409267	0.0014
J_{2xz}	0.0284199502364441	0.269753339175229	0.0102607714654092	0.00415827279030233	14.3569921920133	7.4640e-04
J_{2yz}	-0.00349341313716002	2.33296149141574	0.00076826805457974	-0.0154536414540448	2.3054111673656	-0.0113
$J_{3yy}+J_{2zz}+0.09985\,m_3+0.632\,c_{3z}\,m_3+0.1067\,(m_4+m_5+m_6+m_7)$	1.04277709862499	0.00913880051976703	1.03577961636038	0.934608341141512	0.0972895287336884	0.9491
$J_{3xx}-J_{3yy}+J_{4yy}-0.0068\,m_4$	0.0105900270669364	1.26639366187113	0.0115952028662261	0.0148617879406225	5.81492624535104	-0.0164
$J_{3xy}+0.0825\,c_{4z}\,m_4$	-0.000212952994256617	30.362154615825	----	0.0126539359335843	3.56686186344772	-0.002
J_{3xz}	-0.0104385481552488	0.487399874522834	-0.00548819376576177	-0.00608661113924325	5.21289431802778	0.0021
J_{3yz}	-0.00484578939822065	1.23787966442069	-0.00437747027079482	-0.0137886404379563	2.390652l6162938	-0.0055
$J_{4yy}+J_{3zz}+0.0068\,m_4+0.0136\,m_5+0.0136\,m_6+0.0136\,m_7$	0.116875696122888	0.0690642374025275	0.124454379641766	0.0825491861219277	0.539445669597377	0.0922
$J_{4xx}-J_{4yy}+J_{5yy}+0.768\,c_{5z}\,m_5+0.1406\,(m_5+m_6+m_7)$	0.532431222701l1	0.0202220115117817	0.540830009861762	0.546156450936052	0.095115324036964l	0.5381
$J_{4xy}+0.0317\,m_5+0.0317\,m_6+0.0317\,m_7+0.0825\,c_{5z}\,m_5$	0.150065576541699	0.0274046793405843	0.138763629896975	0.12021718586004	0.166544012462708	0.1255
J_{xx4}	0.00477223336242583	0.907940594558059	0.00390545637077114	0.0039942878536377	6.52849780592255	-0.0020
J_{4yz}	-0.00273179005775726	1.61771068964401	-0.00164409853610947	0.00784034630893363	3.47234203529251	1.8047e-04
$J_{5yy}+J_{4zz}+0.1543\,m_5+0.1543\,m_6+0.1543\,m_7+0.768\,c_{5z}\,m_5$	0.636935579516876	0.0102544668319368	0.632524072189963	0.619605971610654	0.0829425702599799	0.6089
$J_{5xx}-J_{5yy}+J_{6yy}+0.0077\,m_7$	0.032153688598929	0.273853880191597	-0.00749118756382638	0.0113908605848426	3.9130825847695	0.0137
J_{5xy}	-0.00373093470053514	0.957845342351179	-0.00151487508690249	0.00360033703880724	5.77183379287562	2.3992e-04
J_{5xz}	-0.00613311885722265	0.493163771229148	-0.00460057242062744	-0.0101595380396465	1.67641418026388	-0.0117
J_{5yz}	0.00771532410759932	0.434454843989011	0.00216441377866959	0.00799849602524438	2.71328707999367	0.0077
$J_{6yy}+J_{5zz}+0.0077\,m_7$	0.0180693692687778	0.316857102141596	0.0275646178962388	0.0216438275993359	1.63948495212235	0.0210
$J_{6xx}-J_{6yy}+J_{7yy}-0.0077\,m_7$	-0.00653174843772509	1.0591268329627	-0.000884713343309503	-0.0156911207208524	1.8290470363416	-0.0097
$J_{6xy}+0.088\,c_{7z}\,m_7$	0.00536526100475549	0.533117731705522	0.00551140006361939	0.00140972185984143	8.96017184074576	0.0032
J_{6xz}	-1.29010222357792E-005	218.642951947916	----	0.00466040910538727	3.86439315464633	0.0039
J_{6yz}	0.000820441198475142	3.30838131966642	-0.00011149560566302	0.000631001643519202	27.1286492805487	-1.0794e-04
$J_{7yy}+J_{6zz}+0.0077\,m_7$	0.0250459324632208	0.177258273822703	0.030388912660974	0.0144969742256099	1.82265361166624	0.0204
$J_{7xx}-J_{7yy}$	-0.00176759053030307	2.73778209818965	0.00242165389244933	0.00236141379604848	7.3233918251556	2.6087e-04
J_{7xy}	0.0011853220557l447	2.0320331695229	-0.000395430294146216	0.00487483554650531	1.86998080609363	0.0011
J_{7xz}	0.00175781529296103	1.22993712958718	-0.00167239278929351	0.00233524165888097	4.88885570304326	0.0013
J_{7yz}	-0.000596936040654158	3.67559867359549	-0.00054879944912121	-0.00269495602553136	4.71739198098941	-0.0016
J_{7zz}	0.00127244042127159	2.61291364603081	0.00490936463159561	-0.00426963114218529	4.46804771117889	0.0022
$c_{2x}\,m_2$	-0.00536870991776331	0.155126356721021	-0.00203460783952851	0.0287859108662679	1.89257090745338	0.0280
$c_{2y}\,m_2-0.316\,m_4-0.316\,m_5-0.316\,m_6-0.316\,m_7-0.316\,m_3-c_{3x}\,m_3$	-3.10257562423396	0.000304900035457445	-3.10434336792372	-3.1362564554641	0.0105154837122877	-3.1353
$0.0825\,m_4+0.0825\,m_5+0.0825\,m_6+0.0825\,m_7+c_{3x}\,m_3$	0.687355500831219	0.00140435627736313	0.684156438770866	0.680273023107252	0.0480775032145037	0.6814
$c_{3y}\,m_3-c_{4z}\,m_4$	0.0233765365133791	0.0352910904024845	0.0282266905981094	-0.0121106270706461	1.22013217990525	-0.0103
$c_{4x}\,m_4-0.0825\,m_6-0.0825\,m_7-0.0825\,m_5$	-0.488836350422897	0.00156834564623313	-0.490080870249493	-0.472630292902374	0.0367798072434234	-0.4742
$0.384\,m_5+0.384\,m_6+0.384\,m_7+c_{4y}\,m_4+c_{5z}\,m_5$	1.71848566994063	0.00045070085766665	1.72068233187568	1.75173249241485	0.0192516301067527	1.7525
$c_{5x}\,m_5$	-0.0100041967045902	0.0643953740324495	-0.0146541895222708	-0.000503912493509591	26.569855964617	0.0015
$c_{5y}\,m_5-c_{6z}\,m_6$	0.0770531295113475	0.0073674847168779	0.0823861609591039	0.0816837992151493	0.136237435683l2	0.0815
$0.088\,m_7+c_{6x}\,m_6$	0.165680379211635	0.0033594455916706	0.164967969875474	0.163721445813231	0.0744111762455879	0.1639
$c_{6y}\,m_6-c_{7z}\,m_7$	-0.0677117737044103	0.00825760446226429	-0.0688323213466581	-0.0622560121354882	0.15272568276745	-0.0613
$c_{7x}\,m_7$	0.00623108688342054	0.0743590105306178	0.00773512225159871	0.00133680246679693	6.57812339642614	6.3142e-04
$c_{7y}\,m_7$	-0.00048808593311965	0.949033575145546	-0.003127688864l6296	0.0047667686234629	1.67825171955328	0.0033
$f_{v,1}$				0.0665266534059956	1.96584151621768	0.0628
$f_{v,2}$				0.198746668762208	0.961542935001025	0.2088
$f_{v,3}$				0.039926055264758	3.49705294041023	0.0361
$f_{v,4}$				0.225710406362974	0.912430413683524	0.2174
$f_{v,5}$				0.102266664468244	1.58679668848623	0.1021
$f_{v,6}$				-0.0132100723685054	12.2606754655529	1.6128e-04
$f_{v,7}$				0.0637878416456024	2.13884517364764	0.0632
$f_{c,1}$				0.245023479550843	0.588949258058454	0.2549
$f_{c,2}$				0.152329042413405	0.960241278437485	0.1413
$f_{c,3}$				0.182652732613744	0.732398902308952	0.1879
$f_{c,4}$				0.359108046187031	0.415181633314249	0.3625
$f_{c,5}$				0.266921997663113	0.46678968591951	0.2728
$f_{c,6}$				0.165805633616467	0.807745692948203	0.1529
$f_{c,7}$				0.210904095871819	0.705536701742261	0.2097
$f_{o,1}$				-0.107311659739619	0.693156685250328	-0.1069
$f_{o,2}$				-0.156635309821668	2.954935137883l6	-0.1601
$f_{o,3}$				-0.0686346720012536	1.156561578850l6	-0.0718
$f_{o,4}$				-0.252209609088585	0.956307919122076	-0.2562
$f_{o,5}$				0.00445732602659933	19.4632616331215	0.0079
$f_{o,6}$				0.0910111791131084	0.839029697813094	0.0935
$f_{o,7}$				-0.0126699291188763	6.26279708120922	-0.0070

discarded due to their large standard deviations (see the corresponding second

column in the table).

In conclusion, the similarity between the results obtained with the classical method and with our approach indirectly confirms the validity of the set of dynamic coefficients in Table A.2 for the Panda robot.

A.2 Comparison of dynamic parameters

The dynamic parameters of the Panda robot, i.e., the mass, center of mass, and inertia tensor of each link, have been retrieved by solving the nonlinear optimization problem presented in Section 7.4 as well as by using the LMI-SDP framework presented in [91], based on the Python code available at https://github.com/cdsousa/wam7_dyn_ident. This code has been slightly modified in order to include the triangular inequalities on the inertia tensors, according to [24]. The results are reported in the two Tables A.3 and A.4.

In Tab. A.3, the value of c_{1z} is reported as "$*$" (*don't care*) since it does not have any influence on the dynamics (i.e., it does not appear in the E-L model, and can take any value).

The used lower (LB) and upper (UB) bounds and the final value $\hat{\boldsymbol{p}}$ obtained with our optimization algorithm are shown in the second to fourth columns. The final parameters $\boldsymbol{\delta}$ (including 21 friction parameters) obtained with the LMI-SDP method are reported in the fifth columns. In particular, these latter values were retrieved from the dynamic coefficients $\boldsymbol{\beta}^{\star e}$ by means of eq. (49) in [91], using the same bounds (properly manipulated) that we adopted for our algorithm. The resulting $\boldsymbol{\delta}$ set was then slightly manipulated, in order to obtain for each link the center of mass from the its first moment of inertia and the barycentric inertia tensor from the inertia tensors w.r.t. the link frame.

Figure A.1 shows a comparison between the torques measured by the joint torque sensors of the Panda robot during a validation trajectory and the estimated torques generated from a Newton-Euler (N-E) routine using the $\hat{\boldsymbol{p}}$ parameters and the $\boldsymbol{\delta}$ parameters. In order to further validate the parameter sets $\hat{\boldsymbol{p}}$ and $\boldsymbol{\delta}$, we also designed 10 new validation trajectories (each lasting for 10 seconds), performing the same joint torque comparisons and then computing the mean square error (MSE) for each trajectory. The concatenated joint trajectories are reported in Fig. A.3, the concatenated joint torques in Fig. A.2, and the total MSE in Table A.5. These results confirm that both estimates are reliable and consistent.

Table A.3 – Lower and upper bounds for masses and centers of mass of the Panda robot and their corresponding estimated values using our approach ($\hat{\boldsymbol{p}}$) and the one in [91] ($\boldsymbol{\delta}$).

parameter	LB	UB	$\hat{\boldsymbol{p}}$	$\boldsymbol{\delta}$ (from [91])	units
m_1	0	10	4.970684	2.0643e-05	kg
m_2	0	10	0.646926	9.2017	kg
m_3	0	10	3.228604	1.4575	kg
m_4	0	10	3.587895	4.5856	kg
m_5	0	10	1.225946	0.5231	kg
m_6	0	10	1.666555	2.0604	kg
m_7	0	10	7.35522e-01	0.1718	kg
c_{1x}	-0.05	0.05	3.875e-03	5.202e-08	m
c_{1y}	-0.05	0.05	2.081e-03	5.202e-08	m
c_{1z}	-0.4	0.05	*	-0.1750	m
c_{2x}	-0.05	0.05	-3.141e-03	0.0015	m
c_{2y}	-0.15	0.05	-2.872e-02	-0.0578	m
c_{2z}	-0.05	0.05	3.495e-03	-0.0384	m
c_{3x}	-0.05	0.15	2.7518e-02	0.1096	m
c_{3y}	-0.05	0.05	3.9252e-02	0.05	m
c_{3z}	-0.1	0.05	-6.6502e-02	-0.1	m
c_{4x}	-0.15	0.05	-5.317e-02	-0.0485	m
c_{4y}	-0.05	0.15	1.04419e-01	0.15	m
c_{4z}	-0.05	0.05	2.7454e-02	0.0147	m
c_{5x}	-0.05	0.05	-1.1953e-02	-0.004	m
c_{5y}	-0.05	0.05	4.1065e-02	0.045	m
c_{5z}	-0.05	0.05	-3.8437e-02	-0.05	m
c_{6x}	-0.05	0.15	6.0149e-02	0.0732	m
c_{6y}	-0.05	0.05	-1.4117e-02	-0.0251	m
c_{6z}	-0.05	0.05	-1.0517e-02	-0.0276	m
c_{7x}	-0.05	0.05	1.0517e-02	-0.0326	m
c_{7y}	-0.05	0.05	-4.252e-03	0.0087	m
c_{7z}	0.04	0.15	6.1597e-02	0.0396	m

Table A.4 – Lower and upper bounds for the inertia tensor elements of the Panda robot and their corresponding estimated values using our approach ($\hat{\boldsymbol{p}}$) and the one in [91] ($\boldsymbol{\delta}$).

parameter	LB	UB	$\hat{\boldsymbol{p}}$	$\boldsymbol{\delta}$ (from [91])	units
I_{1xx}	0	1	7.0337e-01	0.6	kg·m^2
I_{1xy}	-1	1	-1.3900e-04	-3.2497e-14	kg·m^2
I_{1xz}	-1	1	6.7720e-03	-6.4280e-09	kg·m^2
I_{1yy}	0	1	7.0661e-01	0.6	kg·m^2
I_{1yz}	-1	1	1.9169e-02	-6.4280e-09	kg·m^2
I_{1zz}	0	1	9.1170e-03	2.0353e-06	kg·m^2
I_{2xx}	0	1	7.9620e-03	0.028	kg·m^2
I_{2xy}	-1	1	-3.9250e-03	5.5650e-05	kg·m^2
I_{2xz}	-1	1	1.0254e-02	0.0056	kg·m^2
I_{2yy}	0	1	2.8110e-02	0.0291	kg·m^2
I_{2yz}	-1	1	7.0400e-04	-2.7636e-04	kg·m^2
I_{2zz}	0	1	2.5995e-02	0.0011	kg·m^2
I_{3xx}	0	1	3.7242e-02	2.0154e-06	kg·m^2
I_{3xy}	-1	1	-4.7610e-03	-5.9144e-09	kg·m^2
I_{3xz}	-1	1	-1.1396e-02	1.1523e-08	kg·m^2
I_{3yy}	0	1	3.6155e-02	2.0243e-06	kg·m^2
I_{3yz}	-1	1	-1.2805e-02	6.0956e-09	kg·m^2
I_{3zz}	0	1	1.0830e-02	2.0164e-06	kg·m^2
I_{4xx}	0	1	2.5853e-02	2.0242e-06	kg·m^2
I_{4xy}	-1	1	7.7960e-03	7.1405e-09	kg·m^2
I_{4xz}	-1	1	-1.3320e-03	8.8917e-10	kg·m^2
I_{4yy}	0	1	1.9552e-02	2.0036e-06	kg·m^2
I_{4yz}	-1	1	8.6410e-03	-2.2039e-09	kg·m^2
I_{4zz}	0	1	2.8323e-02	2.0260e-06	kg·m^2
I_{5xx}	0	1	3.5549e-02	2.0057e-06	kg·m^2
I_{5xy}	-1	1	-2.1170e-03	1.6453e-10	kg·m^2
I_{5xz}	-1	1	-4.0370e-03	-3.6824e-10	kg·m^2
I_{5yy}	0	1	2.9474e-02	2.0037e-06	kg·m^2
I_{5yz}	-1	1	2.2900e-04	2.3080e-09	kg·m^2
I_{5zz}	0	1	8.6270e-03	2.0028e-06	kg·m^2
I_{6xx}	0	1	1.9640e-03	2.0026e-06	kg·m^2
I_{6xy}	-1	1	1.0900e-04	1.5823e-09	kg·m^2
I_{6xz}	-1	1	-1.1580e-03	2.0530e-09	kg·m^2
I_{6yy}	0	1	4.3540e-03	2.0070e-06	kg·m^2
I_{6yz}	-1	1	3.4100e-04	-4.9387e-10	kg·m^2
I_{6zz}	0	1	5.4330e-03	2.0073e-06	kg·m^2
I_{7xx}	0	1	1.2516e-02	2.0026e-06	kg·m^2
I_{7xy}	-1	1	-4.2800e-04	2.6159e-10	kg·m^2
I_{7xz}	-1	1	-1.1960e-03	1.4055e-09	kg·m^2
I_{7yy}	0	1	1.0027e-02	2.0037e-06	kg·m^2
I_{7yz}	-1	1	-7.4100e-04	-4.2047e-10	kg·m^2
I_{7zz}	0	1	4.8150e-03	2.0020e-06	kg·m^2

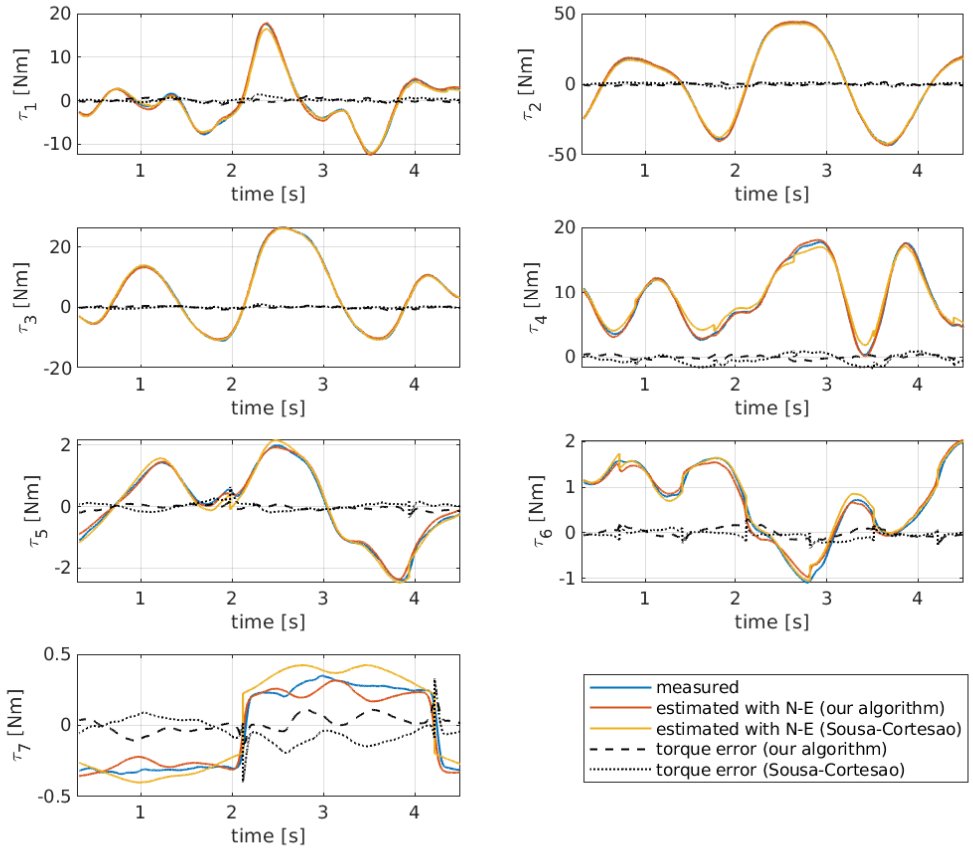


Figure A.1 – Comparison between measured joint torques during a validation trajectory (blue solid lines) and estimated torques generated from a N-E routine using the $\hat{\boldsymbol{p}}$ parameters from our parameter retrieval method in [8] (see Chap. 7) (solid red lines) and the parameters $\boldsymbol{\delta}$ from the LMI-SDP framework in [91] (solid yellow lines). The torque errors are reported as well.

Table A.5 – Mean Square Error (MSE) $[(\text{Nm})^2]$ of torque predictions using the parameters $\hat{\boldsymbol{p}}$ of our method [8] illustrated in chapter 7 and the parameters $\boldsymbol{\delta}$ of LMI-SDP approach [91] in validation experiments of Fig. A.3.

	Joint 1	**Joint 2**	**Joint 3**	**Joint 4**	**Joint 5**	**Joint 6**	**Joint 7**
MSE $\hat{\boldsymbol{p}}$	0.4068	1.0282	0.5951	0.4231	0.1749	0.0403	0.0267
MSE $\boldsymbol{\delta}$	0.4955	1.0155	0.2751	0.6627	0.0828	0.0508	0.0284

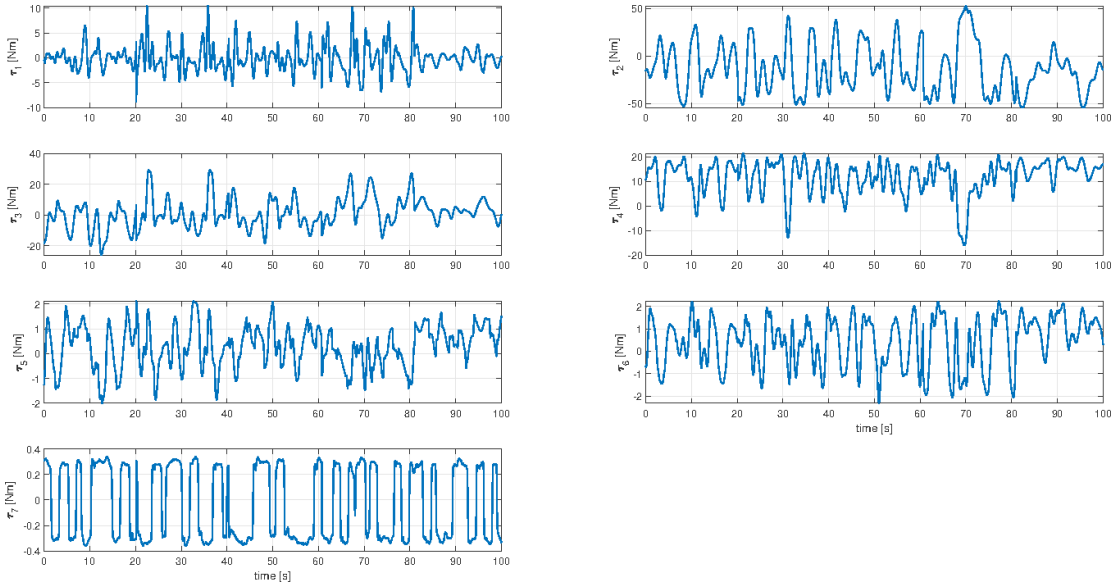


Figure A.2 – Joint torques recorded during the 10 validation trajectories reported in Fig. A.3.

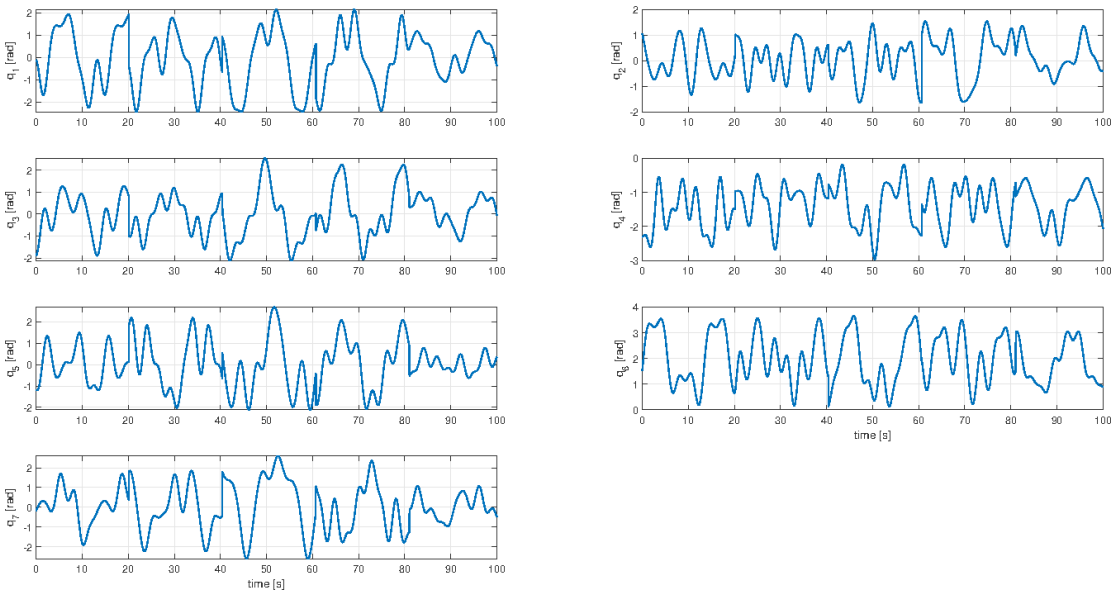


Figure A.3 – Joint positions recorded during 10 validation trajectories (10 seconds each). The associated joint torques are reported in Fig. A.2.

Appendix

B Parameters retrieval using nonlinear and conditional constraints

In order to retrieve a feasible set of dynamic parameters $\hat{\boldsymbol{p}}$ for the Panda robot, only linear constraints were used, since the non-convex shapes of the links would allow their centers of mass to be located even outside the link themselves. We present here a simple example of retrieval of dynamic parameters where the use of nonlinear constraints would be preferable, and which fits then in the framework of our method, presented in Chapter 7.

Consider the spatial 2R robot with orthogonal joint axes in Fig. B.1. Its (standard) Denavit-Hartenberg kinematic parameters are reported in Table B.1. The robot body consists of two links: link 1 is a cylinder of height $l_1 = 1.5$ m and radius $r_1 = 0.15$ m; link 2 has a total length of $l_2 = 0.8$ m and is composed by a cylinder of length $l_2 - l_2'$ and radius $r_2 = 0.08$ m, connected at the end with a truncated cone of length $l_2' = 0.2$ m and with radius $r_3 = 0.02$ m for its smaller base.

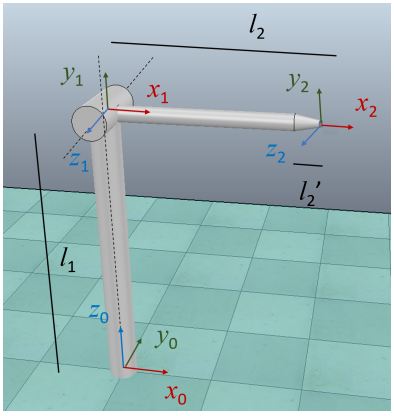


Figure B.1 – A spatial 2R robot with joints along axes z_0 and z_1.

The assumed ground truth values of the dynamic parameters $\boldsymbol{p}_{\mathbf{real}}$ are given in the fourth column of Table B.2.

Table B.1 – DH table of the spatial 2R robot.

i	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	l_1	q_1
2	l_2	0	0	q_2

Table B.2 – Lower and upper bounds for the dynamic parameters of the 2R robot in Fig. B.1, with real $\boldsymbol{p}_{\mathbf{real}}$ and estimated $\hat{\boldsymbol{p}}$ values.

parameter	LB	UB	$\boldsymbol{p}_{\mathbf{real}}$	$\hat{\boldsymbol{p}}$	units
m_1	0	10	8	4.5788	kg
m_2	0	5	3	3.7828	kg
c_{1x}	$-\infty$	∞	0.01	-0.016124	m
c_{1y}	-1.42	0.08	-0.75	-0.74408	m
c_{1z}	$-\infty$	∞	-0.02	-3.8e-04	m
c_{2x}	-0.8	0	-0.1	-0.24487	m
c_{2y}	$-\infty$	∞	0	6.16e-06	m
c_{2z}	$-\infty$	∞	0	-3.66e-03	m
I_{1xx}	0	1	0.03	0.61296	kg·m^2
I_{1xy}	-1	1	0	0.071738	kg·m^2
I_{1xz}	-1	1	0	-0.0099797	kg·m^2
I_{1yy}	0	1	0.03	0.022982	kg·m^2
I_{1yz}	-1	1	0	0.031154	kg·m^2
I_{1zz}	0	1	0.01	0.6172	kg·m^2
I_{2xx}	0	1	0.01	0.019819	kg·m^2
I_{2xy}	-1	1	0	-4.525e-06	kg·m^2
I_{2xz}	-1	1	0	-0.0077412	kg·m^2
I_{2yy}	0	1	0.04	0.35386	kg·m^2
I_{2yz}	-1	1	0	-5.9342e-05	kg·m^2
I_{2zz}	0	1	0.04	0.34412	kg·m^2

The symbolic form of the dynamic coefficients $\boldsymbol{\pi}_{2\mathrm{R}} \in \mathbb{R}^8$ of this robot is:

$$\boldsymbol{\pi}_{2\mathrm{R}} = \begin{pmatrix} J_{1yy} + J_{2yy} - 0.64m_2 \\ J_{2xx} - J_{2yy} + 0.64m_2 \\ J_{2xy} \\ J_{2xz} - 0.8c_{2z}m_2 \\ J_{2yz} \\ J_{2zz} - 0.8m_2 \\ 0.8m_2 + c_{2x}m_2 \\ c_{2y}m_2 \end{pmatrix}.$$

In order to estimate the values of the dynamic coefficients $\boldsymbol{\pi}_{2\mathrm{R}}$, a numerical simulation is performed using $\boldsymbol{p}_{\mathbf{real}}$ and the joint torques are recorded during an exciting motion. Then the dynamic coefficients are properly estimated with an ordinary least squares method.

At this stage, the algorithm for the retrieval of the dynamic parameters $\hat{\boldsymbol{p}}$ is

launched, using the bounds on physical feasibility reported in Table B.2, upper and lower bounds on the total mass, $1 \leq m_1 + m_2 \leq 15$, and linear constraints on the inertia tensors (exploiting the triangular inequality). When using box constraints on the position of the center of mass of each of the two links, the solution would be searched in an extra volume (i.e., a parallelepiped) which is $V_{\text{link1}} = 4/\pi \simeq 1.27$ times larger than the volume of the cylindric link 1 and $V_{\text{link2}} = 4/\pi\,(1 + 2l'_2/l_2) \simeq 2$ times larger than the one of link 2. On the other hand, considering that the distance of the center of mass from the major link axis must be less than its radius, the previous (approximate) box constraints on the center of mass of the two links can be replaced by the following (exact) nonlinear and conditional (if-else) constraints g_{CoM_1} and g_{CoM_2}, to be used within our core optimization **Algorithm 1** in chapter 7:

$$\alpha_1 = r_1 - \sqrt{c_{1y}^2 + c_{1z}^2},$$

$$\alpha_2 = \begin{cases} r_2 - \sqrt{c_{2y}^2 + c_{2z}^2}, & \text{if } c_{2x} < -l'_2 \\ \left(r_2 + \frac{r_3 - r_2}{l'_2}(l'_2 + c_{2x})\right) - \sqrt{c_{2y}^2 + c_{2z}^2}, & \text{otherwise,} \end{cases}$$

$$g_{\text{CoM}_1} = -\min\{0, \alpha_1\}, \quad g_{\text{CoM}_2} = -\min\{0, \alpha_2\}.$$

Taking advantage of the knowledge of the link shapes, these nonlinear constraints ensure that each center of mass will lie inside the corresponding link shape but that no feasible position is being excluded, thus yielding a complete solution. A validation test was finally performed, comparing the joint torques on sinusoidal trajectories that are computed using the real parameters with the joint torques estimated by means of a N-E routine fed with the parameters $\hat{\boldsymbol{p}}$ reported in the fifth column of Table B.2. The retrieved solution generates just the same dynamics as obtained with the real parameters.

Another interesting quantity that can be estimated only when using the N-E inverse dynamics algorithm is the wrench acting at the robot joint level, i.e., the exchanged forces/moments between two successive links connected by a joint. To this aim, it is very important to retrieve a set of dynamic parameters that is as close as possible to the real one. To show this, we relaxed the constraints in order to obtain a second solution set of dynamic parameters $\hat{\boldsymbol{p}}'$, which, however, contains also non-feasible parameters (i.e., parameters that have no physical meaning).

In Fig. B.2, the yellow lines represent the estimated joint torques during a validation experiment coming from a N-E routine that is being fed with the unfeasible parameters $\hat{\boldsymbol{p}}'$. Despite this, reliable motion torque estimations can still be appreciated. On the contrary, a strong inconsistency may be observed when estimating the internal forces acting on the joints, obtained as a byproduct of

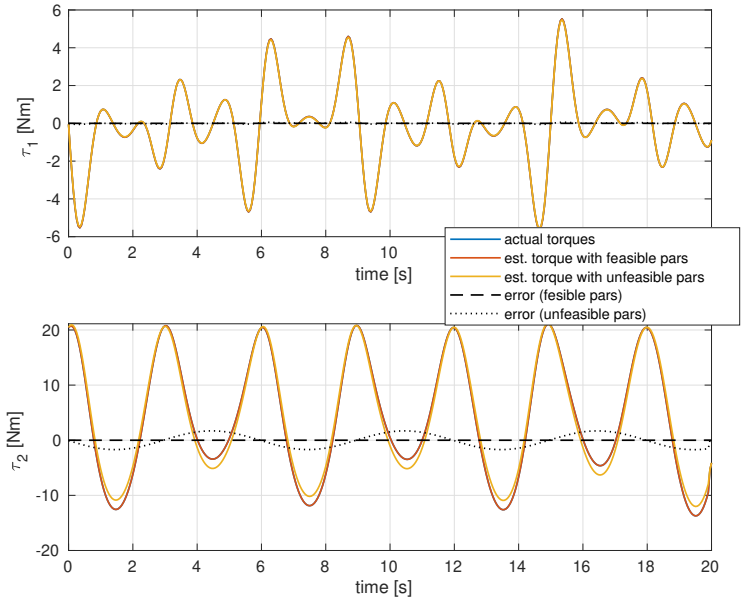


Figure B.2 – Validation of estimated parameters in Table B.2. A N-E routine fed with the feasible parameters $\hat{\boldsymbol{p}}$ and with some unfeasible parameters $\hat{\boldsymbol{p}}'$ returns similar estimated torques (respectively, red and yellow lines) along the motion. When comparing these estimates with the actual torques recorded along a sinusoidal joint trajectory (blue lines, almost overlapped by the red line), we notice that both sets provide a good torque estimation.

the standard N-E algorithm. As shown by the values of these estimated forces in Fig. B.3, it can be seen that estimations obtained from the feasible parameters $\hat{\boldsymbol{p}}$ (red lines) are close to the real forces (blue lines), while estimations retrieved from the unfeasible parameters $\hat{\boldsymbol{p}}'$ (yellow lines) are almost 10^6 times larger than the real values "felt" at the joints. This confirms that a physically-consistent solution which is close to the real one is mandatory for estimating the joint wrenches.

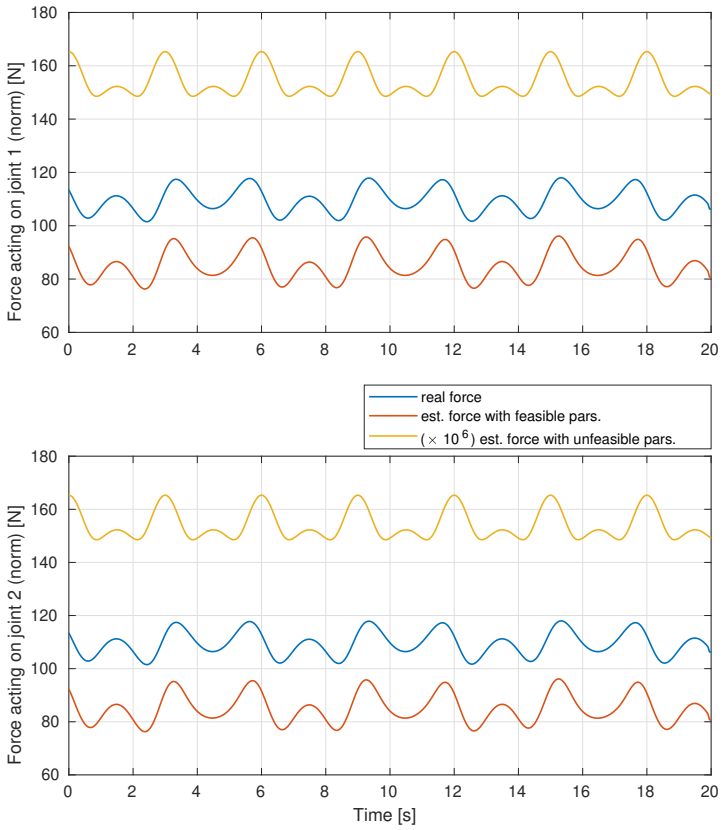


Figure B.3 – Internal forces acting on the joints of the simulated 2R robot during a sinusoidal joint trajectory. Estimates of the internal forces obtained from a N-E routine are significantly closer to the real internal forces (blue lines) if feasible (i.e., physically consistent) dynamic parameters are provided to the routine, while estimates obtained from an unfeasible set $\hat{\boldsymbol{p}}'$ can be even 10^6 times larger than real internal forces (note that the yellow line was scaled for the ease of reading).

Bibliography

[1] R. Bajcsy, "Integrating vision and touch for robotics applications," in *Artificial Intelligence Applications for Business: Proceedings of the NYU Symposium*, May 1983, pp. 297 – 323.

[2] A. Cherubini and D. Navarro-Alarcon, "Sensor-based control for collaborative robots: Fundamentals, challenges, and opportunities." *Front. Neurorobot.*, 2021.

[3] J. Baeten and J. De Schutter, *Integrated Visual Servoing and Force Control - The Task Frame Approach*. Springer, 01 2003.

[4] R. P. Paul, *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*. MIT Press, 1981.

[5] Y. Nakabo and M. Ishikawa, "Visual impedance using 1 ms visual feedback system," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 3, 1998, pp. 2333–2338 vol.3.

[6] A. A. Oliva, P. Robuffo Giordano, and F. Chaumette, "A general visual-impedance framework for effectively combining vision and force sensing in feature space," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4441–4448, 2021.

[7] A. A. Oliva, E. Aertbeliën, J. de Schutter, P. Robuffo Giordano, and F. Chaumette, "Towards dynamic visual servoing for interaction control and moving targets," *2022 IEEE International Conference on Robotics and Automation (to appear)*, May 2022.

[8] C. Gaz, M. Cognetti, A. Oliva, P. Robuffo Giordano, and A. De Luca, "Dynamic identification of the Franka Emika Panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Robotics and Automation Lett.*, 2019.

[9] A. A. Oliva, F. Spindler, P. Robuffo Giordano, and F. Chaumette, "FrankaSim: A Franka Emika's Panda sobot simulator with visual-servoing

enabled capabilities," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (submitted)*, 2022.

[10] E. Marchand, F. Spindler, and F. Chaumette, "Visp for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robotics and Automation Magazine*, vol. 12, no. 4, pp. 40–52, December 2005.

[11] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[12] N. Hogan, "Impedance control: An approach to manipulation: Part I-Theory," *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, pp. 1–7, 03 1985.

[13] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modeling, Planning and Control*, 3rd ed. London: Springer, 2008.

[14] M. H. Raibert and J. J. Craig, "Hybrid position/force control of manipulators," *Journal of Dynamic Systems, Measurement, and Control*, vol. 103, no. 2, pp. 126–133, 06 1981.

[15] M. T. Mason, "Compliance and force control for computer controlled manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 6, pp. 418–432, 1981.

[16] S. Chiaverini and L. Sciavicco, "The parallel approach to force/position control of robotic manipulators," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 4, pp. 361–373, 1993.

[17] 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*, vol. 7, no. 4, pp. 18–33, 1988.

[18] J. Denavit and R. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Trans ASME Journal of Applied Mechanics*, p. 215–221, June 1955.

[19] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. Pearson Prentice Hall, 2005.

[20] C. Gaz, F. Flacco, and A. De Luca, "Extracting feasible robot parameters from dynamic coefficients using nonlinear optimization methods," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2016, pp. 2075–2081.

[21] W. Khalil and E. Dombre, *Modeling, Identification and Control of Robots*. Hermes Penton London, 2002.

[22] S. Traversaro, S. Brossette, A. Escande, and F. Nori, "Identification of fully physical consistent inertial parameters using optimization on manifolds," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 5446–5451.

[23] P. Wensing, S. Kim, and J.-J. E. Slotine, "Linear matrix inequalities for physically-consistent inertial parameter identification: A statistical perspective on the mass distribution," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, 07 2017.

[24] C. Sousa and R. Cortesão, "Inertia tensor properties in robot dynamics identification: A linear matrix inequality approach," *IEEE/ASME Transactions on Mechatronics*, vol. PP, pp. 1–1, 01 2019.

[25] A. De Luca, C. Manes, and F. Nicolò, "A task space decoupling approach to hybrid control of manipulators," *IFAC Proceedings Volumes*, vol. 21, no. 16, pp. 157–162, 1988, 2nd IFAC Symposium on Robot Control 1988 (SYROCO '88), Karlsruhe, FRG, 5-7 October. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667017546030

[26] T. Yoshikawa, "Dynamic hybrid position/force control of robot manipulators–description of hand constraints and calculation of joint driving force," *IEEE Journal on Robotics and Automation*, vol. 3, no. 5, pp. 386–392, 1987.

[27] T. Yoshikawa, T. Sugie, and M. Tanaka, "Dynamic hybrid position/force control of robot manipulators-controller design and experiment," *IEEE Journal on Robotics and Automation*, vol. 4, no. 6, pp. 699–705, 1988.

[28] A. De Luca, C. Manes, and G. Ulivi, "Robust hybrid dynamic control of robot arms," in *Proceedings of the 28th IEEE Conference on Decision and Control,*, 1989, pp. 2641–2646 vol.3.

[29] A. De Luca and C. Manes, "Hybrid force-position control for robots in contact with dynamic environments," *IFAC Proceedings Volumes*, vol. 24,

no. 9, pp. 177–182, 1991, 3rd IFAC Symposium on Robot Control 1991 (SYROCO'91), Vienna, Austria, 16-18 September 1991. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667017510526

[30] W. D. Fisher and M. S. Mujtaba, "Hybrid position/force control: A correct formulation," *The International Journal of Robotics Research*, vol. 11, no. 4, pp. 299–311, 1992. [Online]. Available: https://doi.org/10.1177/027836499201100403

[31] L. D. Joly and A. Micaelli, "Hybrid position/force control, velocity projection, and passivity," *IFAC Proceedings Volumes*, vol. 30, pp. 325–331, 1997.

[32] C. Collewet and E. Marchand, "Photometric visual servoing," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 828–834, 2011.

[33] Q. Bateux and E. Marchand, "Histograms-based visual servoing," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 80–87, 2017.

[34] Q. Bateux, E. Marchand, J. Leitner, F. Chaumette, and P. Corke, "Visual Servoing from Deep Neural Networks," in *RSS 2017 - Robotics : Science and Systems, Workshop New Frontiers for Deep Learning in Robotics*, Boston, United States, Jul. 2017, pp. 1–6. [Online]. Available: https://hal.inria.fr/hal-01589887

[35] J. Hill and W. T. Park, "Real time control of a robot with a mobile camera," *Proc. 9th ISIR*, pp. 233–246, Mar. 1979.

[36] P. Corke, *Visual Control of Robots: High Performance Visual Servoing*. Research Studies Press Ltd, 1996.

[37] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach. (Second edition)*. Prentice Hall, Nov. 2011. [Online]. Available: https://hal.inria.fr/hal-01063327

[38] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003.

[39] F. Chaumette, "Visual Servoing," in *Robot Manipulators: Modeling, Performance Analysis and Control*, Dombre, Etienne, Khalil, and Wissama, Eds. ISTE, 2007, pp. 279–336. [Online]. Available: https://hal.inria.fr/hal-00920418

[40] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 313–326, 1992.

[41] F. Chaumette, "Image moments: a general and useful set of features for visual servoing," *IEEE Transactions on Robotics*, vol. 20, no. 4, pp. 713–723, 2004.

[42] F. Chaumette, P. Rives, and B. Espiau, "Classification and realization of the different vision-based tasks," in *Visual Servoing*, ser. World Scientific Series in Robotics and Intelligent Systems, 1993, vol. 7, pp. 199 – 228. [Online]. Available: https://hal.inria.fr/hal-01548352

[43] E. Malis, "Visual servoing invariant to changes in camera-intrinsic parameters," *Robotics and Automation, IEEE Transactions on*, vol. 20, pp. 72 – 81, 03 2004.

[44] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[45] ——, "Visual servo control. ii. advanced approaches [tutorial]," *IEEE Robotics Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007.

[46] S. Vandernotte, A. Chriette, P. Martinet, and A. S. Roos, "Dynamic sensor-based control," in *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2016, pp. 1–6.

[47] F. Fusco, O. Kermorgant, and P. Martinet, "A comparison of visual servoing from features velocity and acceleration interaction models," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4447–4452.

[48] F. Chaumette, S. Hutchinson, and P. Corke, "Visual servoing," *Handbook of Robotics, 2nd edition*, pp. 841–866, 2016.

[49] V. Lippiello, B. Siciliano, and L. Villani, "A position-based visual impedance control for robot manipulators," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 2068–2073.

[50] K. Nottensteiner, A. Sachtler, and A. Albu-Schäeffer, "Towards autonomous robotic assembly: Using combined visual and tactile sensing for adaptive task execution," *Journal of Intelligent & Robotic Systems*, vol. 101, 03 2021.

[51] K. Yu and A. Rodriguez, "Realtime state estimation with tactile and visual sensing. application to planar manipulation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7778–7785.

[52] K.-T. Yu and A. Rodriguez, "Realtime state estimation with tactile and visual sensing for inserting a suction-held object," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1628–1635.

[53] B. Nelson, J. Morrow, and P. Khosla, "Robotic manipulation using high bandwidth force and vision feedback," *Mathematical and Computer Modelling*, vol. 24, no. 5, pp. 11–29, 1996. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0895717796001136

[54] D. Lawrence, "Impedance control stability properties in common implementations," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, 1988, pp. 1185–1190 vol.2.

[55] G. Morel, E. Malis, and S. Boudet, "Impedance based combination of visual and force control," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 2, 1998, pp. 1743–1748 vol.2.

[56] D. E. Whitney, "Force Feedback Control of Manipulator Fine Motions," *Journal of Dynamic Systems, Measurement, and Control*, vol. 99, no. 2, pp. 91–97, 06 1977. [Online]. Available: https://doi.org/10.1115/1.3427095

[57] Y. Mezouar, M. Prats, and P. Martinet, "External hybrid vision/force control," *Intl. Conference on Advanced Robotics*, 2007.

[58] V. Perdereau and M. Drouin, "A new scheme for hybrid force-position control," in *RoManSy 9*, A. Morecki, G. Bianchi, and K. Jaworek, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 150–159.

[59] G. Morel and P. Bidaud, "A reactive external force loop approach to control manipulators in the presence of environmental disturbances," vol. 2, 05 1996, pp. 1229 – 1234 vol.2.

[60] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar, "A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks," in *2009 International Conference on Advanced Robotics*, 2009, pp. 1–6.

[61] O. Khatib, L. Sentis, J. Park, and J. Warren, "Whole-body dynamic behavior and control of human-like robots," *Int. J. Humanoid Robotics*, vol. 1, pp. 29–43, 2004.

[62] D. J. Agravante, A. Cherubini, A. Bussy, P. Gergondet, and A. Kheddar, "Collaborative human-humanoid carrying using vision and haptic sensing," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 607–612.

[63] C. Cai and N. Somani, "Visual servoing in a prioritized constraint-based torque control framework," in *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2018, pp. 309–314.

[64] J. D. Schutter, T. D. Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx, "Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty," *The International Journal of Robotics Research*, vol. 26, no. 5, pp. 433–455, 2007. [Online]. Available: https://doi.org/10.1177/0278364907078091O7

[65] E. Aertbeliën and J. De Schutter, "etasl/etc: A constraint-based task specification language and robot controller using expression graphs," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 1540–1546.

[66] C. Samson, M. Le Borgne, and B. Espiau, *Robot Control: The Task Function Approach*. Caledron Press, Oxford, 1991.

[67] H. Bruyninckx and J. De Schutter, "Specification of force-controlled actions in the "task frame formalism"-a synthesis," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 581–589, 1996.

[68] T. de Laet, R. Smits, H. Bruyninckx, and J. de Schutter, "Constraint-based task specification and control for visual servoing application scenarios," vol. 60, no. 5, pp. 260–269, 2012. [Online]. Available: https://doi.org/10.1524/auto.2012.0996

[69] S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, 2017.

[70] Hong Zhang and J. P. Ostrowski, "Visual servoing with dynamics: control of an unmanned blimp," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 1, 1999, pp. 618–623 vol.1.

[71] M. Vukobratović and D. Stokić, "Is dynamic control needed in robotic systems, and, if so, to what extent?" *The International Journal of Robotics Research*, vol. 2, no. 2, pp. 18–34, 1983. [Online]. Available: https://doi.org/10.1177/027836498300200202

[72] M. Ishikawa, A. Namiki, T. Senoo, and Y. Yamakawa, "Ultra high-speed robot based on 1 khz vision system," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5460–5461.

[73] N. Shahriari, S. Fantasia, F. Flacco, and G. Oriolo, "Robotic visual servoing of moving targets," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 77–82.

[74] A. D. Luca, G. Oriolo, and P. Robuffo Giordano, "On-line estimation of feature depth for image-based visual servoing schemes," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 2823–2828.

[75] E. Rohmer, S. P. N. Singh, and M. Freese, "Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013. [Online]. Available: www.coppeliarobotics.com

[76] T. Larsen, N. Andersen, O. Ravn, and N. Poulsen, "Incorporation of time delayed measurements in a discrete-time kalman filter," in *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)*, vol. 4, 1998, pp. 3972–3977 vol.4.

[77] E. Simetti and G. Casalino, "A novel practical technique to integrate inequality control objectives and task transitions in priority based control," *Journal of Intelligent & Robotic Systems*, vol. 84, pp. 877–902, 2016.

[78] F. Fusco, O. Kermorgant, and P. Martinet, "Integrating features acceleration in visual predictive control," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5197–5204, 2020.

[79] A. Maciejewski, "Dealing with the ill-conditioned equations of motion for articulated figures," *IEEE Computer Graphics and Applications*, vol. 10, no. 3, pp. 63–71, 1990.

[80] R. Featherstone, "An empirical study of the joint space inertia matrix," *I. J. Robotic Res.*, vol. 23, pp. 859–871, 09 2004.

[81] V. Lippiello, G. A. Fontanelli, and F. Ruggiero, "Image-based visual-impedance control of a dual-arm aerial manipulator," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1856–1863, 2018.

[82] R. Carelli, E. Oliva, C. Soria, and O. Nasisi, "Combined force and visual control of an industrial robot," *Robotica*, vol. 22, no. 2, p. 163–171, 2004.

[83] C. Gaz and A. De Luca, "Payload estimation based on identified coefficients of robot dynamics — with an application to collision detection," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3033–3040.

[84] S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, Dec 2017.

[85] E. Magrini and A. De Luca, "Hybrid force/velocity control for physical human-robot collaboration tasks," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Oct. 2016.

[86] J. Hollerbach, W. Khalil, and M. Gautier, "Model identification," in *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008, pp. 321–344.

[87] A. Janot, P. Vandanjon, and M. Gautier, "A generic instrumental variable approach for industrial robot identification," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 1, pp. 132–145, Jan 2014.

[88] J. Jovic, A. Escande, K. Ayusawa, E. Yoshida, A. Kheddar, and G. Venture, "Humanoid and human inertia parameter identification using hierarchical optimization," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 726–735, June 2016.

[89] V. Bonnet, P. Fraisse, A. Crosnier, M. Gautier, A. González, and G. Venture, "Optimal exciting dance for identifying inertial parameters of an an-

thropomorphic structure," *IEEE Transactions on Robotics*, vol. 32, no. 4, pp. 823–836, Aug 2016.

[90] V. Mata, F. Benimeli, N. Farhat, and A. Valera, "Dynamic parameter identification in industrial robots considering physical feasibility," *Advanced Robotics*, vol. 19, no. 1, pp. 101–119, 2005.

[91] C. Sousa and R. Cortesão, "Physical feasibility of robot base inertial parameter identification: A linear matrix inequality approach," *Int. J. of Robotics Research*, vol. 33, no. 6, pp. 931–944, 2014. [Online]. Available: https://doi.org/10.1177/0278364913514870

[92] Franka Emika, "Ros integration for franka emika research robots," https://github.com/frankaemika/franka_ros.

[93] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[94] M. Spong, "Modeling and control of elastic joint robots," *ASME J. Dyn. Syst. Meas. Control*, vol. 109, no. 4, pp. 310–319, 1987.

[95] C. Gaz, F. Flacco, and A. De Luca, "Identifying the dynamic model used by the KUKA LWR: A reverse engineering approach," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 1386–1392.

[96] J. Swevers, W. Verdonck, and J. De Schutter, "Dynamic model identification for industrial robots," *IEEE Control Systems Mag.*, vol. 27, no. 5, pp. 58–71, 2007.

[97] A. Jubien, M. Gautier, and A. Janot, "Dynamic identification of the Kuka LightWeight robot: Comparison between actual and confidential Kuka's parameters," in *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, July 2014, pp. 483–488.

[98] T. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[99] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2009.

[100] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. Athena Scientific, Boston, MA, 1996.

[101] M. Santos Pessoa de Melo, J. Gomes da Silva Neto, P. Jorge Lima da Silva, J. M. X. Natario Teixeira, and V. Teichrieb, "Analysis and comparison of robotics 3d simulators," in *2019 21st Symposium on Virtual and Augmented Reality (SVR)*, 2019, pp. 242–251.

[102] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 1082–10 828.

[103] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8973–8979.

[104] M. Križmančić, "Franka_gazebo," https://github.com/mkrizmancic/franka_gazebo.

[105] S. Sidhik, "Panda_simulator: Gazebo simulator for franka emika panda robot supporting sim-to-real code transfer," https://github.com/justagist/panda_simulator, DOI: 10.5281/zenodo.3818280.

[106] ——, "Mujoco panda," https://github.com/justagist/mujoco_panda.

[107] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[108] A. Trabelsi, J. Sandoval, M. Ghiss, and M. A. Laribi, "Development of a franka emika cobot simulator platform (csp) dedicated to medical applications," in *Advances in Service and Industrial Robotics*, S. Zeghloul, M. A. Laribi, and J. Sandoval, Eds. Cham: Springer International Publishing, 2021, pp. 95–103.

[109] L. Pitonakova, M. Giuliani, A. Pipe, and A. Winfield, "Feature and performance comparison of the v-rep, gazebo and argos robot simulators," in *Towards Autonomous Robotic Systems*, M. Giuliani, T. Assaf, and M. E.

Giannaccini, Eds. Cham: Springer International Publishing, 2018, pp. 357–368.

[110] J. Collins, D. Howard, and J. Leitner, "Quantifying the reality gap in robotic manipulation tasks," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6706–6712.

[111] P. Corke, "The machine vision toolbox: a matlab toolbox for vision and vision-based control," *IEEE Robotics Automation Magazine*, vol. 12, no. 4, pp. 16–25, 2005.

[112] P. M. Khiabani, B. S. Aghdam, J. Ramezanzadeh, and H. D. Taghirad, "Visual servoing simulator by using ros and gazebo," in *2016 4th International Conference on Robotics and Mechatronics (ICROM)*, 2016, pp. 308–312.

[113] R. Smits, "KDL: Kinematics and Dynamics Library," http://www.orocos.org/kdl.

[114] P. Corke, "Robotics, Vision & Control," Springer 2017, ISBN 978-3-319-54413-7.

[115] A. De Luca and L. Ferrajoli, "A modified newton-euler method for dynamic computations in robot fault detection and control," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 3359–3364.

[116] A. De Luca, A. Albu-Schäffer, S. Haddadin, and G. Hirzinger, "Collision detection and safe reaction with the DLR-III lightweight robot arm," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006, pp. 1623–1630.

[117] W. Chung, L.-C. Fu, and S.-H. Hsu, *Motion Control*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 133–159. [Online]. Available: https://doi.org/10.1007/978-3-540-30301-5_7

[118] P. Schmidt, J. Artigas, M. De Stefano, R. Balachandran, and C. Ott, "Increasing the performance of torque-based visual servoing by applying time domain passivity," *IFAC-PapersOnLine*, vol. 48, no. 19, pp. 13–18, 2015, 11th IFAC Symposium on Robot Control SYROCO 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896315026270

[119] D. Ma, S. Dong, and A. Rodriguez, "Extrinsic contact sensing with relative-motion tracking from distributed tactile measurements," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 262–11 268.

[120] H. Bruyninckx, S. Demey, S. Dutré, and J. D. Schutter, "Kinematic models for model-based compliant motion in the presence of uncertainty," *The International Journal of Robotics Research*, vol. 14, no. 5, pp. 465–482, 1995. [Online]. Available: https://doi.org/10.1177/027836499501400505

[121] S. Tabrik, M. Behroozi, L. Schlaffke, S. Heba, M. Lenz, S. Lissek, O. Güntürkün, H. R. Dinse, and M. Tegenthoff, "Visual and tactile sensory systems share common features in object recognition," *eNeuro*, vol. 8, no. 5, 2021. [Online]. Available: https://www.eneuro.org/content/8/5/ENEURO.0101-21.2021

[122] W. Yuan, S. Dong, and E. H. Adelson, "Gelsight: High-resolution robot tactile sensors for estimating geometry and force," *Sensors*, vol. 17, no. 12, 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/12/2762

[123] A. De Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger, "Collision detection and safe reaction with the dlr-iii lightweight manipulator arm," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 1623–1630.

[124] S. Haddadin, A. Albu-Schaffer, A. De Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical human-robot interaction," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 3356–3363.

[125] L. Manuelli and R. Tedrake, "Localizing external contact using proprioceptive sensors: The contact particle filter," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 5062–5069.

[126] J. Bimbo, C. Pacchierotti, N. G. Tsagarakis, and D. Prattichizzo, "Collision detection and isolation on a robot using joint torque sensing," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7604–7609.

[127] M. Gautier and W. Khalil, "Direct calculation of minimum set of inertial parameters of serial robots," *IEEE Trans. on Robotics and Automation*, vol. 6, no. 3, pp. 368–373, 1990.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : Intégration de la commande visuel et de la force pour l'interaction physique avec des manipulateurs robotiques

Mot clés : Asservissement visuel; Commande de la forces; commande de conformité et d'impédance; commande basé sur des capteurs; Manipulateurs

Résumé : Dispositions combinées vision/force deviennent de plus en plus courants dans les petites et moyennes entreprises à mesure que les "Cobots" deviennent plus abordables. Caméras peuvent fournir une description détaillée de la scène, tandis qu'un capteur de force fournis des informations locales mais très précises sur le contact. Cependant, en raison de la nature très différente de ces modalités de détection, il n'est pas simple d'obtenir une utilisation combinée efficace. Dans cette thèse, nous traitons du couplage de la vision et de la détection de la force en tirant parti de leur complémentarité et en permettant à un robot manipulateur de réaliser activement un mouvement conforme aux directions de la tâche visuelle. Nous avons développé des schémas de contrôle avancés avec des estimateurs qui gèrent la différence de taux de mise à jour entre les capteurs. De plus, nous contribuons à la récupération des paramètres du modèle dynamique pour les manipulateurs, qui sont des ingrédients clés à la fois dans les contrôleurs proposés, qui prennent en compte la dynamique du manipulateur, ainsi que pour effectuer des simulations plus précises. Les résultats rapportés, tant dans le simulateur développé que dans les expériences réelles, montrent l'efficacité des techniques proposées.

Title: Integration of vision and force control for physical interaction with robotic manipulators

Keywords: Visual-servoing; Force control; compliance and impedance control; Sensor-based control; Manipulators

Abstract: Combined vision/force arrangements are becoming more and more common in small and medium sized businesses as "Cobots" becomes more affordable. Cameras can provide a detailed description of the scene, whereas force sensing can provide local but highly precise information about the contact. However, due to the very different nature of these sensing modalities , obtaining an effective combined use is not straightforward. In this thesis we deal with the coupling of vision and force sensing leveraging their complementarity and allowing a robot manipulator to actively achieve compliant motion along the visual task directions. We have developed advanced control schemes with estimators that cope with the update rate discrepancy between the sensors. furthermore, we contribute to the dynamic model parameter retrieval for manipulators, which are key ingredients both in the proposed controllers, whose take into account the manipulator's dynamics, as well as for performing more accurate simulations. The reported results, both in the developed simulator and with real experiments show the effectiveness of the proposed techniques.