Solutions for safe human-robot collaboration

Benjamin Navarro

Centre Val de Loire UNIVERSITÉ Spirit of the Loire Valley

UNIVERSITÉ D'ORLÉANS

UNIVERSITE D'ORLEANS

ÉCOLE DOCTORALE MATHÉMATIQUES, INFORMATIQUE, PHYSIQUE THÉORIQUE ET INGÉNIERIE DES SYSTÈMES

Laboratoire : PRISME – Pôle IRAuS

Thèse présentée par :

Benjamin Navarro

soutenue le : **14 décembre 2017**

pour obtenir le grade de : **Docteur de l'Université d'Orléans**

Discipline/ Spécialité : **Robotique**

Solutions for safe human-robot collaboration

Thèse dirigée par :

Gérard POISSON	Professeur, Université d'Orléans
Philippe FRAISSE	Professeur, Université de Montpellier

RAPPORTEURS :

Véronique PERDEREAU	Professeur, UPMC
Angelika PEER	Professeur, UWE, Angleterre

JURY :

Nikolaos TSAGARAKIS	Directeur de recherche, IIT, Italie, examinateur
Aïcha FONTE	Maître de conférences, UO, co-encadrante
Andrea CHERUBINI	Maître de conférences HDR, UM, co-encadrant

Acknowledgments

Contents

Introduction **11**

1 Background and State of the art **15**
1.1 Human-robot interactions . . . 15
1.1.1 Terminology . . . 15
1.1.2 Social interaction . . . 16
1.1.3 Physical interaction . . . 16
1.1.4 Physical collaboration . . . 17
1.2 Compliant actuators . . . 17
1.3 Control for safe physical human-robot interaction and collaboration . . . 18
1.4 Interacting with robots other than serial manipulators . . . 20
1.4.1 Mobile manipulators . . . 20
1.4.2 Robotic hands . . . 21

2 Joint torque control and external force estimation **23**
2.1 External torques and forces estimation . . . 24
2.2 Torque control of a Kuka LWR4+ . . . 25
2.3 Torque control scheme . . . 28
2.4 Experiments . . . 31
2.5 Conclusion . . . 40

3 Task space control solutions **41**
3.1 Two-layer safe damping control framework . . . 42
3.2 Force inputs . . . 44
3.2.1 Interaction forces . . . 45
3.2.2 Virtual stiffness and mass . . . 45
3.2.3 Potential field method . . . 45
3.3 Velocity inputs . . . 46
3.3.1 Reference trajectory . . . 46
3.3.2 Force control . . . 46
3.4 Constraints . . . 47
3.4.1 Emergency stop . . . 47
3.4.2 Velocity limitation . . . 47
3.4.3 Acceleration limitation . . . 48
3.4.4 Power limitation . . . 48
3.4.5 Force limitation . . . 49
3.4.6 Kinetic energy limitation . . . 50

3.4.7 Separation distance . . . 51
3.5 Software implementation . . . 51
3.5.1 Project organization . . . 52
3.5.2 Example . . . 52
3.5.3 Benchmarks . . . 54
3.5.4 Sum up . . . 54
3.6 Experiment . . . 55
3.7 Conclusion . . . 61

4 Extending to other robots **63**
4.1 Mobile comanipulation framework . . . 63
4.1.1 End-effector control . . . 63
4.1.2 Whole body control strategy . . . 64
4.1.3 Constraints . . . 64
4.1.4 Distance to singularities . . . 64
4.1.5 Manipulability . . . 65
4.1.6 Distance to objects . . . 65
4.1.7 Angular deviation . . . 66
4.1.8 Constraint deactivation . . . 66
4.1.9 Merging the constraints . . . 67
4.1.10 Experiments . . . 67
4.1.11 Validation . . . 67
4.1.12 Real experimental setup . . . 71
4.1.13 Results . . . 72
4.1.14 Conclusion on mobile comanipulation . . . 73
4.2 Hand control . . . 74
4.2.1 Tactile sensing . . . 74
4.2.2 Grasp motion and force control . . . 76
4.2.3 Test cases . . . 77
4.2.4 Conclusion on hand control . . . 82
4.3 Conclusion . . . 84

Conclusion **85**

A Polynomial interpolation and trajectory generation **87**
A.1 Fith-order polynomials . . . 87
A.2 Interpolation . . . 89
A.3 Trajectory generation . . . 89
A.3.1 Constrained trajectory generation . . . 90
A.3.2 Synchronization . . . 91
A.3.3 The case of orientations . . . 92
A.3.4 Path tracking . . . 93
A.3.5 Benchmarks . . . 93

List of Figures

1 Lightweight robots . . . 11
2 General block Diagram of the contents of the thesis and its structure. . . . 13

1.1 Difference between divisible and interactive tasks . . . 16
1.2 Stiff, series elastic and variable stiffness actuators. . . . 18
1.3 Illustration of the ISO15066 standard. . . . 19

2.1 Average torque tracking error and standard deviation on a Kuka LWR4+. . 26
2.2 Torque tracking accuracy on a Kuka LWR4+. . . . 27
2.3 Average stiction torques and standard deviation of a Kuka LWR4+. . . . 28
2.4 Stiction torques for each joint of a Kuka LWR4+. . . . 29
2.5 Proposed torque control scheme with stiction compensation. . . . 31
2.6 Joints numbering of a Kuka LWR4+ . . . 32
2.7 Position command $\mathbf{q}^*$ and response $\mathbf{q}$, feedforward only (case 1). . . . 33
2.8 Torque commands, feedforward only (case 1). . . . 34
2.9 Torque commands, feedforward with friction compensation (case 2). . . . 34
2.10 Position command and response, feedforward with friction compensation (case 2). . . . 35
2.11 Position command and response, feedforward with friction compensation and a collision (case 3). . . . 36
2.12 Torque commands, feedforward with friction compensation and a collision (case 3). . . . 37
2.13 External torques, feedforward with friction compensation and a collision (case 3). . . . 37
2.14 Position command and response, feedforward with friction compensation plus PD control and a collision (case 4). . . . 38
2.15 Torque commands, feedforward with friction compensation plus PD control and a collision (case 4). . . . 39
2.16 External torques, feedforward with friction compensation plus PD control and a collision (case 4). . . . 39

3.1 Overview of the proposed controller. . . . 42
3.2 Examples of interaction, stiffness and repulsive forces. . . . 44
3.3 Safe and unsafe power values. . . . 49
3.4 Velocity limitation depending on separation distance. . . . 51
3.5 Benchmarks of the controller running on a 7 degrees of freedom manipulator. 55
3.6 Setup for the experiment. . . . 56

3.7 Finite state machine used for the experiment. A + sign indicates an addition to the controller (new constraint or new input) while a - indicates a removal. 57
3.8 Snapshots of the experiment: teaching (a-c) and replay (d-h) phases. . . . 58
3.9 Relevant variables during the experiment. 60

4.1 Distance to singularity simulation. Top: smallest singular value σ_m and damping factor λ, middle: singularity constraints $a_s = a_{s,v_x} = a_{s,v_y} = a_{s,\omega_z}$, bottom: velocity commands along the x axis. t_{act} is the time at which the constraint gets activated. 68
4.2 Manipulability simulation. Top: manipulability measure m, middle: manipulability constraints $a_m = a_{m,v_x} = a_{m,v_y} = a_{m,\omega_z}$, bottom: Velocity commands along the x axis. t_{act} is the time at which the constraint gets activated. 69
4.3 Distance to objects simulation. Top: Minimal distance d_x, middle: workspace constraints, bottom: Velocity commands along the x axis; t_{act} is the time at which the constraint gets activated. 69
4.4 Angular deviation simulation. Top: Angular error $\Delta\theta_z$, middle: workspace constraints, bottom: Velocity commands (z axis); t_{act} is the time at which the constraint gets activated. 70
4.5 The Bazar mobile manipulator. 71
4.6 Snapshots of the experiment. 72
4.7 Experimental results. From top to bottom: velocity command $\dot{\mathbf{x}}$ (m/s, rad/s), arm velocity command $\dot{\mathbf{x}}_{arm}$ (m/s, rad/s), base velocity command $\dot{\mathbf{x}}_{base}$ (m/s, rad/s) and constraint values a_{v_x}, a_{v_y} and a_{ω_z}. 73
4.8 BioTac with its attached reference frame. 74
4.9 Electrodes on a BioTac sensor. 75
4.10 BioTac sensor calibration. 76
4.11 Grasping FSM 77
4.12 EMG controlled robotic hand setup 78
4.13 Raw and filtered EMG signals. 79
4.14 Hand configurations. From left to right: open hand, palmar pinch and key-grip. 80
4.15 EMG based control mode 5 80
4.16 Force regulation on the little finger (Z axis). 82
4.17 FSM states during the collaborative screwing experiment. 83
4.18 FSM for the collaborative screwing experiment. 83

A.1 Interpolation function f_{int} for $x^- = 0.5$, $x^+ = 2$, $y^- = 0$ and $y^+ = 0.25$. . . 89
A.2 Comparison of the synchronization mechanisms. 91
A.3 Trajectories' coefficients computation benchmark. 95

List of Tables

3.1 Detailed project hierarchy. 52

4.1 EMG-based control modes. 79

A.1 Trajectories' waypoints. 94
A.2 Trajectories. 94
A.3 Number of iterations to compute the trajectories. 94

Introduction

The need for collaborative robots (cobots) is becoming more and more important over the years. The industry seeks cobots that partially automate current manual tasks, to help human workers during difficult tasks by reducing pain, fatigue and the associated risk of injury. Factories also want to increase their flexibility by allowing non-robotics experts to program new behaviors through learning by demonstration, hence allowing skilled workers to transfer their knowledge to robots. Health care institutions could also benefit from collaborative robot technology for surgery, physiotherapy and domestic assistance of elderly or disabled people, to mention a few.

Despite the high demand for cobots, there is one aspect that is limiting their proliferation: *safety*. Indeed, cobots must be capable of ensuring the safety of their operators, other human beings in their surroundings and lastly their own, before being suited for a general adoption. Safety first comes from mechanical design, starting with the so-called lightweight robots. These robots present round shapes and low link inertia to lower their kinetic energy, reducing injuries upon impact. Many robot manufacturers provide such robots in different configurations: single or dual arms and with or without a mobile base, as shown in Fig. 1. Fixed single arm robots are closer to common industrial ones and thus easier to integrate in today's factories. They also generally present basic safety measures (e.g. collision detection, velocity and power limitation), to simplify the integration regarding current safety standards. However, since these robots are meant to collaborate with humans, more human-like shapes can be beneficial. Fixed-base dual-arm robots adopt a human upper-body structure and can act as a human helper for part assembly or inspection. While fixed-base robots can already be found in some places, mobile manipu-

Figure 1 – Examples of lightweight robots, from left to right: Kuka LBR iiwa (2013), Rethink Robotics Sawyer (2015), ABB YuMi (2015), Pal Robotics TIAGo (2015), Kawada Industries HRP-4 (2011).

lators and humanoids are still absent from the factory floor. Both types benefit from their increased mobility (using either wheels or legs), e.g., to help with object transportation or large product inspections. On one hand, wheeled cobots, with their inherent stability and the strong research community working on localization, path planning and control for decades, are almost ready to be utilized in factories. On the other hand, they are limited to relatively flat surfaces and cannot accommodate to any obstacle (e.g. stairs, ladders). Humanoid robots, by adopting a complete human-like structure (legs, torso, arms, head) are intended to deal with any environment accessible to humans. However, they are still research products only, since many challenges need to be overcome before their adoption (e.g. locomotion on uneven terrain, stability, fall recovery, mechanical limitations).

When considering collaborative applications, lightweight robots clearly represent an improvement over the classical robotic manipulators currently in use in factories, but they can be enhanced using passive compliance to better absorb shocks due to unexpected collisions. Passive compliance can take different forms, from soft covers to elastic joint transmissions. The former can easily be added to existing robots with a relatively low cost, but provides only a limited range of action, while the latter, by deforming the whole structure, has a higher range of action but requires a specific and costly joint design and cannot be added to already existing robots. In any case, the purpose of mechanical compliance is to serve as a fast impact absorption mechanism before the robot's controller takes over. Indeed, control can help in various ways to increase the system's safety. First, preventive actions (typically protective stops or collision avoidance) can be taken to avoid dangerous situations. Then, if a physical contact occurs, additional compliance can be introduced to overcome the mechanical compliance limits. This is often performed using impedance or admittance control or one of their many variations. However, during collaborative tasks, physical contacts between the robot and a human are often necessary. In that case, different safety measures must be taken, such as velocity, power or force monitoring. Moreover, the robot must remain stable once the contact with the human is established. For example, during manual guidance, nothing forbids the operator to move the robot to a singular configuration, an action which will lead to instability if not properly accounted for.

This thesis focuses on the control of collaborative robots and solutions to enforce safe behavior, particularly in the presence of humans.

Figure 2 gives an overview of the work discussed in this thesis and its repartition among the chapters. In this figure, two switches are present, S_{MM} and S_{HA}, both represented in their *off* state. The first one allows to switch between controlling a single robot (*off*) and controlling a mobile manipulator (*on*), using a special redundancy solution. The second one switches between the control of an arm using torque commands (*off*) and a hand via position commands (*on*). Since it does not make sense to use the hand along with a mobile base, the case $S_{MM} = S_{HA} = on$ is forbidden.

Some background on human-robot interaction and collaboration will be given in Chapter 1. Then, in Chapter 2, we investigate how torque control and external forces estimation can be performed on real robots in the presence of non-modelisable static frictions. Chapter 3 presents a generic framework to design collaborative tasks using a serial manipulator while ensuring various safety criteria. Finally, extensions from the classic serial manipulator to mobile manipulators and robotic hands are presented in Chapter 4.

This thesis has been supported by ANR (French National Research Agency) SISCob

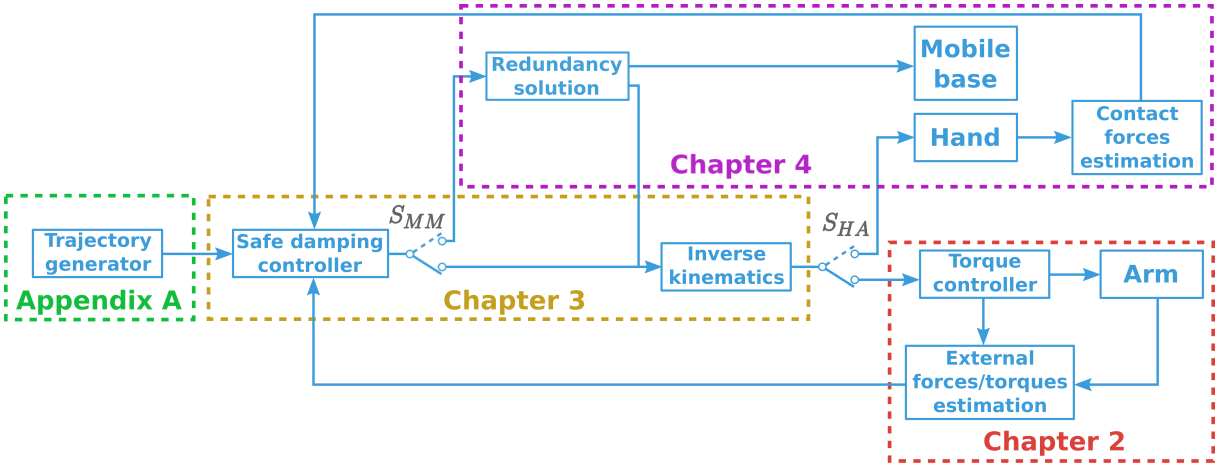


Figure 2 – General block Diagram of the contents of the thesis and its structure.

(Safety Intelligent Sensor for Cobots) project[1]. This project aimed at increasing the safety of collaborative robots by improving their compliance, either though mechanical design with a novel passive compliant device or through control, as discussed in this thesis.

The main contributions of this work are:

- A procedure for an active calibration of tactile sensors mounted on a robotic hand, presented in [1] and in Section 4.2.1.
- A safe adaptive damping control framework answering the constraints of the ISO10218-2011 standard. The results were published in [2].
- A framework for the collaboration between a human operator and a mobile manipulator, with a strong emphasis on the intuitiveness of operation. The work has been published in [3] and is presented in Section 4.1.
- A generalization of the safe adaptive damping framework, able to deal with many collaborative scenario and freely distributed within the OpenPHRI software. It has been submitted to Robotics and Automation Magazine for the special issue on Human-robot collaboration for production environments. It is presented in Chapter 3.
- The control of a robotic hand by individuals with tetraplegia using an EMG interface. The study has been published in [4] and presented in 4.2.3.
- The control in torque of a robot in presence of non-modelisable static frictions, presented in Chapter 2.
- A polynomial-based trajectory generator with velocity and acceleration constraints, suitable for joint or task space trajectories, presented in Appendix A.

[1] ANR-14-CE27-0016: `http://anr-siscob.prd.fr`

Chapter 1

Background and State of the art

The contribution of this thesis is toward better physical interactions and collaborations between robots and human, while ensuring safety criteria to protect both the operators and the environment. This can be achieved in several ways, starting with specifically designed actuators or dedicated devices to higher level control schemes. Some background on physical human-robot interaction (pHRI) and collaboration (pHRC) is given in Sect. 1.1. In Sect. 1.2, we will detail how mechanical design and low level controllers has been used as a first step to enable human-robot interaction. Next, in Sect. 1.3, we will see how higher level control solutions has been tailored for pHRI with robotic manipulators, and how safe operation has been ensured. Finally, in Sect. 1.4, we present works where pHRI has been extended to robotic systems other than serial manipulators.

1.1 Human-robot interactions

Human-robot interaction (HRI) has received increasing attention in recent years from both the academic research and the industry [5–7]. HRI is a very vast domain since interactions between a human and a robot can and will occur in various scenarios, either in homes or at work. These interactions can be split in two main groups, social and physical interactions. Some terminology employed in HRI will be given in 1.1.1, then a brief overview of social human-robot interactions will be given in subsection 1.1.2 before moving to physical interactions (pHRI) (1.1.3) and physical collaborations (pHRC) (1.1.4) which are the two main topics of this thesis.

1.1.1 Terminology

In this thesis, we adopt the task taxonomy presented in [8]. Some slight modifications to that taxonomy have been performed here, in order to target precisely the work of this thesis without altering the meanings. A task performed between two agents (humans and/or robots) can be described in two different ways:

1. divisible vs. interactive task,
2. competitive vs. cooperative task.

A *divisible* task employs multiple agents working without any conflict between them, whereas for an *interactive* task, the work has to be performed jointly and simultaneously.

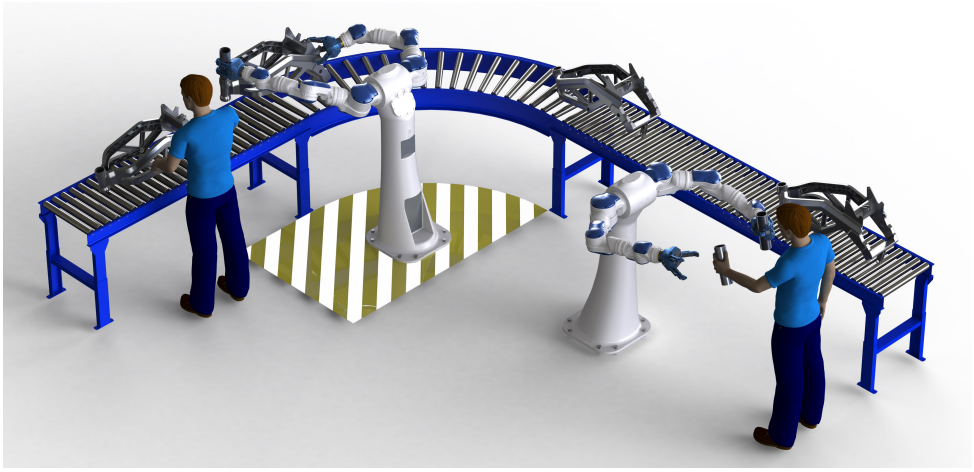


Figure 1.1 – Difference between divisible and interactive tasks[1].

For example, surface related tasks, such as cleaning, mowing or painting, can be divided among the agents while large object transportation can only be performed in an interactive way. Figure 1.1 illustrate *divisible* vs. *interactive* tasks, where the robot on the left operates on its own part and has its own workspace, whereas the robot on the right is interacting with a human operator to handover a part. On the other hand, *competitive* tasks require the agents to work against each other, while joint work is required to perform a *cooperative* task. A chess game is an example of a competitive task since both agents pursue their own goal. On the other hand, an object assembly, e.g. one agent holding an object and the other putting parts on it, is a cooperative task. This thesis addresses interactive and cooperative tasks.

1.1.2 Social interaction

Social human-robot interactions are essentially built around communication between the agents. This communication can be performed in several ways, including speech [9, 10], gestures [9, 11], gaze [12] or even emotions through voice and facial analysis [13, 14]. The exchange is bidirectional, since each agent tries to transfer information or intentions to the other one while getting some feedback. In order for the robot to communicate properly with a human, researchers have first looked into human-human interactions in order to highlight the key aspects that need to be mimicked [15, 16]. This is crucial, since humans tend to expect the robot to have a human-like behavior and will even attribute it some personality traits depending on its actions [17], which can improve or degrade the interactions. Social human-robot interaction is a very important key towards better collaborative robots but has not been dealt with in this thesis since the focus has been made on physical interaction.

1.1.3 Physical interaction

Physical human-robot interactions refer to situations where a physical contact occurs between the two agents. This contact can be either intentional or undesired from the

[1]Source: `www.robotics.org`

human perspective. Undesired contacts generally happen when the human enters the robot's workspace while no presence detection system (e.g. light barriers, floor mat, laser scanner) is used and can lead to severe injuries and ultimately death, as it already happened multiple times with industrial robots [18]. Voluntary physical interactions, on the contrary, emerge when the person makes contact with the robot to stop it, to guide it or to teach it a behavior for instance. These two types of physical interaction require different design or control strategies to ensure the humans' safety, as will be detailed respectively in Sections 1.2 and 1.3.

1.1.4 Physical collaboration

Collaboration can be seen as a special case of interaction. In this thesis, we will refer to physical human-robot collaboration for any task performed jointly by a human and a robot, that is both interactive and cooperative. Collaborations including multiple robots and/or multiple humans are out of scope here. pHRC has a great potential in many areas where the robot can enhance the human skills or lower the task's difficulty and the associated health risks [19]. This includes robots used for object transportation, as assistive tools, rehabilitation devices or exoskeletons. During collaborative tasks, it is still crucial that the robot presents a safe behavior but it is also necessary to be intuitive to use and to ease the task completion. In [20], the robot adapts its configuration during a collaborative load carrying task to decrease static joint torques in the human body to limit human fatigue and the risk of injury. In [21], human muscular fatigue is estimated in order to provide a higher assistance level when necessary, by adjusting the control parameters. Such considerations are very important for assistive robots to be effectively accepted and introduced in the industry, health care centers or at home.

1.2 Compliant actuators

In order to be fast and precise, classic industrial robotic manipulators are designed to be very stiff at both joint and structural levels. This leads to little impact absorption in case of collision. Moreover, their shape may present sharp edges and the high inertia of their links may induce high kinetic energy dissipation upon impact, leading to severe injuries [22]. This is why mechanical design is the first step toward safer robots that can be used for interactive tasks. This question is already partly solved since the birth of so-called light-weight robots [23], that present low inertia, thanks to the use of advanced materials, and round shapes to mitigate the injuries and the damages in case of impact. However, most of these robots still rely on stiff actuation and require attention on the control part to truly behave safely. Some compliant joint designs have been proposed [24–26]. These include series elastic actuators (SEA), where a passive compliant element (e.g., a spring) is introduced to absorb the high frequency impact forces. In most designs, the passive element stiffness can be adjusted, to be rigid and precise when moving at low velocities or compliant during high speed motions to limit the injuries or damages induced by an impact [27]. A schematic view of these types of actuators is given in Fig. 1.2. The elastic element can also be used to store, then release, energy for more efficient walk or for throwing objects for instance, as demonstrated in [28] and [29]. The major drawback of such mechanisms is that they limit the torque control bandwidth,

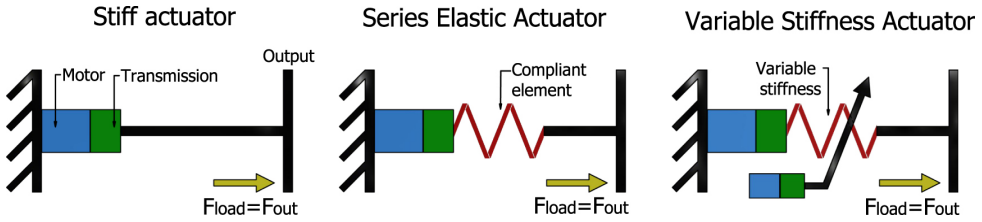


Figure 1.2 – Stiff, series elastic and variable stiffness actuators[1].

introducing inaccuracies in the controlled position, even in absence of collision. To deal with this problem, distributed macro-mini actuators (DM^2) have been proposed [24]. With this solution, the joint actuation is divided between two motors: a powerful one being fixed at the robot base, actuating the joint trough cables and elastic coupling, and another smaller one at the joint level with a stiff, low friction and high bandwidth actuation. This gives the same properties as SEA, while being able to provide high frequency torques thanks to the joint level actuators, thus increasing accuracy, force control performance and disturbances rejection. Also, having only small motors at the joint level decreases the overall inertia, making the robot safer to work with. The main issue here (that limited the proliferation of this solution) is the added complexity and cost to doubling the actuators. But even with a proper mechanical design, control needs to be taken into account for safe operations, since predictive actions can be performed and mechanical compliancy, when present, has a limited range of action and can only be used as the first security measure before the control can take over. The goal of the ANR SISCob project, in which our laboratories are involved, is to increase the robots' safety with on one hand, the design of a new modular device that brings safety using intrinsic compliance, in the vein of the SEAs, and on the other hand, to approach safety using control. Control techniques for pHRI are reviewed in the next section.

1.3 Control for safe physical human-robot interaction and collaboration

During interaction and collaboration with a human, the robot must adopt a safe behavior to minimize the risk of injuries to close coworkers. However, until recently, precise requirements for a collaborative robot were not specified. In 2011, in the last revision of the ISO10218 standard [31], the International Organization for Standardization included requirements for a safe industrial robot. This standard specifies that any robot must respect velocity, power and contact force limits at the tool control point (TCP) in the presence of a human. In the original standard specification, numerical values were given for these limitations (0.25m.s^{-1}, 80W, 150N) but these are now left to be fixed by the end-user, depending on the performed task and on the degree of collaboration between operator and robot. The ISO/TS 15066 [32] further extends the ISO10218 by defining four types of collaborative operation, illustrated in Fig. 1.3, that can also be combined:

- *Safety-rated monitored stop*: interactions with the robot are only allowed when the robot is stopped. Automatic operation is resumed when the operator leaves the

[1]Source: [30].

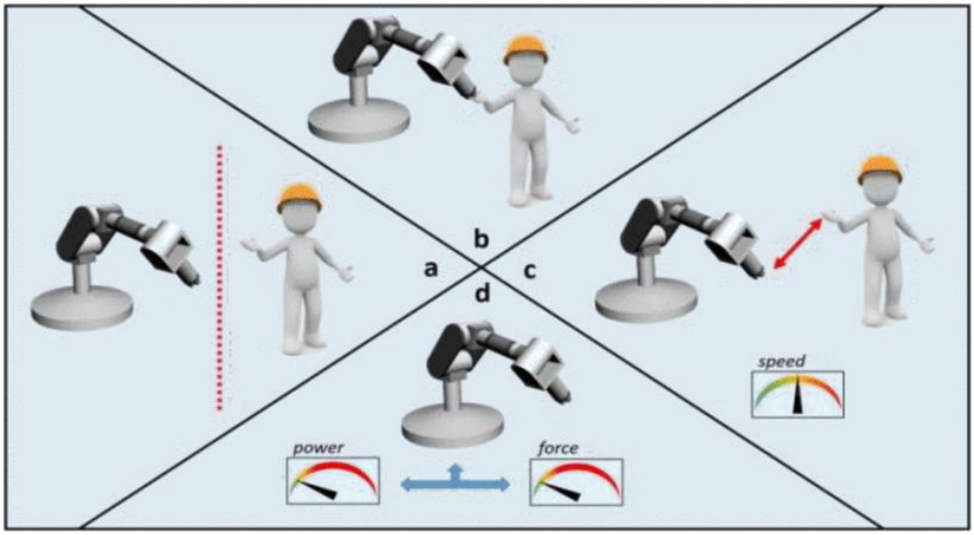


Figure 1.3 – Illustration of the ISO15066 standard[1]. a) *Safety-rated monitored stop*, b) *Hand-guiding operation*, c) *Speed and separation monitoring*, d) *Power and force limitation*.

collaborative workspace.

- *Hand-guiding operation*: the operator guides the robot using physical contact.
- *Speed and separation monitoring*: the robot speed is reduced as the operator gets closer to the robot. A protective stop is issued when a potential contact occurs.
- *Power and force limitation*: exerted force and transmitted power are limited to avoid any harm to the operator in the case of accidental contact with the robot. A risk assessment for each body region must be performed to derive the imposed limitations.

The *safety-rated monitored stop* is what most present-day robot manufacturers [34] provide, since it is very simple to integrate with presence detection systems already available. Any intrusion detected in the robot workspace will trigger a protective stop. *Hand-guiding operation* is also proposed by some manufacturers [34], since it allows non-programming experts to perform tasks with a collaborative robot (e.g. heavy object transportation) hence increasing the production line flexibility. These kinds of tasks fall into what is called teaching-by-demonstration. Teaching-by-demonstration methods allow the robot to learn from a human specific behaviors to be reproduced. These range from simple pick and place [35] tasks to both motion and control gain learning [36] making the robot behave "as humanly" as possible. *Speed and separation monitoring*, to the best of our knowledge, is not directly implemented by robot manufacturers but has been investigated in the research literature, e.g., in [37]. Finally, *power and force limitation* is a much more challenging problem since it requires first a complete risk assertion study to determine which power and force limits to use, and second specific measures to ensure these limitations. Robot manufacturers generally implement a fixed electrical power limitation

[1]Source [33].

(e.g. 80W for the original ISO10218, as with the Kuka LWR4+) but this may be too restrictive since more power may be needed when no operator is present. Control solutions can help to better monitor the exchanged power and force and dynamically adapt the limitations. Force limitation methods can be employed for torque control robots, as it has been shown in [38]. Otherwise, a protective stop or a fallback strategy can be triggered when a force threshold is trespassed.

After this overview of the literature in safe pHRI, it can be seen that no single solution capable of dealing with the four collaborative operations defined by the ISO15066 exists. In Chapter 3, we will detail how a unique framework can be constructed to bridge this gap.

1.4 Interacting with robots other than serial manipulators

Even if most of the physical human-robot interaction research is performed using serial manipulators, other types of robots can be of interest. In this survey, as a complement of the classical serial manipulators studied in the first part, we consider mobile manipulators, that benefit from the increased workspace offered by their mobile base, and robotic hands, that can handle more complex objects than simple grippers and that can also rely on tact to sense the environment and interact with humans. To our knowledge of the literature, applications using dual manipulators in pHRI are not numerous and authors generally consider them as two separate arms [38].

1.4.1 Mobile manipulators

Mobile manipulators benefit from the dexterity of a standard manipulator with the extended workspace of a mobile platform. Locomotion can be realized by wheeled, legged or flying bases. However, mobile manipulators are over-actuated robots that need specific control algorithms to deal with their redundancy. Several approaches have been proposed to deal with this issue in non-collaborative cases, depending on the type of mobile base that is used. For wheeled bases, differential drive actuation introduces non-holonomic constraints due to the rolling without slipping of the wheels on the ground. These constraints limit the set of velocities that can be realized by the mobile base, and that need to be integrated in the controller. This has been addressed with several approaches, e.g., using a path planning strategy [39] or producing a complete kinematics model together with a redundancy scheme [40]. Instead, for mobile bases with steerable wheels, a global kinematics model cannot be used directly, since velocities on the steering axes do not induce velocities on the robotics platform. This has been investigated in [41], where the Jacobian null space projection and a global input-output linearization with dynamic feedback have been tested and compared.

Legged robots equipped with an arm generally fall into two categories, biped and quadruped. Biped robots, usually adopt a humanoid structure. For these systems, locomotion and manipulation are tightly coupled, since the robot balance needs to be guaranteed. Generally, for dealing with the system's high redundancy, researchers use optimization with a set of tasks (e.g., base velocity and hand/s pose/s) and constraints (e.g.,

stability and self collision avoidance). This strategy has been applied in [42] and in [43], to achieve human-humanoid interaction. Quadruped robots equipped with a manipulator have been investigated since [44], where the base is modeled as a parallel robot to define the pose of the arm's base frame and inverse kinematics are used to control the whole robot.

Recently, researchers have embedded manipulators on aerial robots [45, 46]. In such scenarios, the dynamic effects of the arm motion must be taken into account, along with redundancy, to keep the robot stable.

When considering physical human interaction with a mobile manipulator, only a few works have been published. In [47], a fully omnidirectional wheeled robot is made compliant using force control. Physical interaction with mobile manipulators has also been studied in [48], where force thresholds are used to decide if the base, arm or both have to move, and in [49] where the use of a tactile skin permits full body compliance.

In section 4.1, we will introduce a solution exploiting redundancy for mobile manipulators with omnidirectional bases that focuses on the intuitiveness of operation during human-robot collaborations with a mobile manipulator.

1.4.2 Robotic hands

Hands, robotic or human ones, can be used in various ways, including communicating between agents, environment sensing and object grasp/handover. Human-robot interactions can greatly benefit from such features. Take an example where a hand-arm system is used to perform an assembly, with some parts being out of reach. When the robot has to grasp an object outside of its workspace, it can point it to inform an operator that it requires assistance to perform the task, as in [50]. This non-verbal and non-physical interaction is very powerful, since it is easily understandable by anyone, in contrast with voice based communication. Once the operator picked up the part, he has to hand it over to the robot, leading to a physical interaction between the two agents. Human-robot handover has been investigated by the research community and several solutions are available [51, 52]. Tactile sensing can then be useful to successfully grasp the object, but can also be used as a way to enable physical communication, as shown in Sect. 4.2.1, where a finger's tactile sensor is used to trigger the various tasks required to insert screws in a wood piece. Handshaking is another example of physical communication implying robotics hands and had been investigated in [53, 54].

In section 4.2, we will detail how tactile sensing capabilities of a robotic hand can be used to derive the contact force at the fingertips and how it can be used to enable object grasping and tactile communication with a human operator.

From this review of the state-of-the-art in safe physical human-robot interaction and collaboration, we can notice that a lot of work still need to be done on both the hardware and the control sides to obtain truly safe robots. This thesis focuses on control, first with serial manipulators at both a low level, as discussed in Chapter 2, and at a higher level, with the unified framework for safe pHRI presented in Chapter 3. And finally, in Chapter 4, extensions of this framework to omnidirectional mobile manipulators and robotic hands will be presented.

Chapter 2

Joint torque control and external force estimation

As previously mentioned, the robot dynamic model is often required for safe pHRI, since its knowledge allows:

- low level torque control,
- estimation of the external (often, human-applied) forces and torques $\mathbf{f}_{ext} \in \mathbb{R}^6$,
- derivation of interesting metrics, e.g., the reflected inertia [55], explained in Sect. 3.4.6.

However, the robot dynamic model is generally not provided by the manufacturers[1] and therefore an identification procedure is required. Furthermore, any modification to the robot, such as mounting a tool at the end-effector, requires updating the model. Collaborative robots may be subject to frequent tool exchange if the operators require different tools to perform different tasks using the same robot. In such cases, the robot must be able to quickly identify its tool to update its dynamic model in order to gain accuracy in its positioning and in the estimation of the interaction forces and torques.

To this end, we developed a novel approach for identifying the inertial parameters (links' mass, center of mass, inertia matrix) and joint friction coefficients (viscous and dry friction) of a Kuka LWR4+ using an optimal exciting motion[2]. In that work, we compared three different cost functions to maximize the identification accuracy of the identified parameters, while minimizing the exciting motion duration. We also considered the geometric model identification of a tool based on a look-up table and on an inverse geometric model of the robot.

Since the modeling and estimation part of this work has been mainly developed by the co-authors, in this thesis we will focus on the estimation of the external forces/torques and on torque control, which are the aspects I contributed to. In particular, we show the difficulties that arise when dealing with static friction for torque control. We propose a solution to this, and validate it in experiments, including one where the robot collides with a human operator, a case study of major interest in safe pHRI.

[1] Some may provide approximate parameters based on CAD data.

[2] This joint work with Katsumata et al. has been submitted to Robotics and Autonomous Systems.

2.1 External torques and forces estimation

In order to detect physical interaction with the environment or the operator intention, two approaches are generally considered.

The first one consists in mounting a force/torque sensor at the robot end-effector to measure interaction forces and torques. If a tool is attached to the sensor, its inertial parameters must be identified in order to remove its effects on the measurements. Physical interaction with a human usually happen with a slowly moving robot, so the identification of the tool mass and center of mass is sufficient to get a good compensation. This translates to:

$$\mathbf{f}_{ext} = \mathbf{f}_{FT} - \begin{bmatrix} m_{tool}\ {}^{T}\mathbf{R}_{B}\ \mathbf{g} \\ \mathbf{c}_{tool} \times m_{tool}\ {}^{T}\mathbf{R}_{B}\ \mathbf{g} \end{bmatrix}, \tag{2.1}$$

where $\mathbf{f}_{FT} \in \mathbb{R}^6$ is the force/torque vector measured by the sensor, m_{tool} and $\mathbf{c}_{tool}$ are the mass and center of mass of the tool, given in the sensor frame, $\mathbf{g}$ is the Earth gravity vector expressed in the robot base frame and ${}^{T}\mathbf{R}_{B}$ is the rotation matrix between the robot base and tool frames.

The second approach is applicable when joint torque sensors are mounted between the gearbox output and the attached link. In such a case, as it is with the Kuka LWR4+, the knowledge of the robot dynamic model allows for individual joint external torque estimation and for reconstructing the external forces/torques at the end-effector, using the Jacobian matrix. Let us first recall the dynamic model expression for a rigid serial manipulator in the free space (i.e., when no external forces/torques are applied):

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \boldsymbol{\tau}_f(\dot{\mathbf{q}}) = \boldsymbol{\tau}_{dyn}, \tag{2.2}$$

with $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n\times n}$ the robot inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n\times n}$ the Coriolis and centripetal matrix, $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^n$ the vector of gravity torques and $\boldsymbol{\tau}_f \in \mathbb{R}^n$ the joint torques due to dry and viscous friction effects. n is the joint space dimension. In the presence of external interaction torques $\boldsymbol{\tau}_{ext}$ or forces $\mathbf{f}_{ext}$:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{dyn} + \boldsymbol{\tau}_{ext} \tag{2.3}$$

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{dyn} + \mathbf{J}^{\top}(\mathbf{q})\mathbf{f}_{ext}, \tag{2.4}$$

where $\boldsymbol{\tau} \in \mathbb{R}^n$ indicates the torques acting on the joints and $\mathbf{J} \in \mathbb{R}^{6\times n}$ the manipulator Jacobian matrix tied to the point of application of $\mathbf{f}_{ext}$, generally assumed to be the end-effector. In (2.3), $\boldsymbol{\tau}_{ext} \in \mathbb{R}^n$ is the vector of externally applied torques. In (2.4), $\mathbf{f}_{ext} \in \mathbb{R}^6$ is the external wrench applied at the end-effector, mapped to the joint space using the manipulator's Jacobian matrix. Using (2.2), the above equations become:

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \boldsymbol{\tau}_f(\dot{\mathbf{q}}) + \boldsymbol{\tau}_{ext} \tag{2.5}$$

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \boldsymbol{\tau}_f(\dot{\mathbf{q}}) + \mathbf{J}^{\top}(\mathbf{q})\mathbf{f}_{ext}. \tag{2.6}$$

With an accurate dynamic model and joint torque sensors, the external joint torques and external wrench, both needed to detect interactions at the end-effector or along the kinematic chain, can be computed using:

$$\boldsymbol{\tau}_{ext} = \boldsymbol{\tau} - \boldsymbol{\tau}_{dyn} \tag{2.7}$$

$$\mathbf{f}_{ext} = \mathbf{J}^{-\top}(\mathbf{q})\boldsymbol{\tau}_{ext} \tag{2.8}$$

In the case of a redundant manipulator, the Moore-Penrose pseudo-inverse can be used in (2.8) instead of the classic matrix inversion. In the case of the Kuka LWR4+, both $\boldsymbol{\tau}_{ext}$ and $\mathbf{f}_{ext}$ can be computed thanks to the presence of joint torque sensors.

To conclude on this section, it is important to note that the two presented approaches are not perfect. Indeed, using a force/torque sensor at the end-effector often requires integration work for attaching the device to the robot, dealing with the sensor cable, etc. Moreover, it cannot be used to detect collisions or interactions forces on points along the robot other than the end-effector, leading to potential security issues. On the other hand, as mentioned previously, joint torque sensors are not always available due to their cost and to the increased integration complexity. Besides, even when available, they cannot be used to estimate the external wrench at the end-effector in the presence of kinematic singularities, as can be seen from (2.8), nor the location and magnitude of a contact along the kinematic chain. The same applies to the first approach, unless a tactile skin covers the robot joints, providing the contact point position. For the best sensing capabilities, both solutions should be used jointly but this, of course, comes with an increased cost and system complexity. In our case, we prefer to rely on a force/torque sensor at the end-effector for a more precise estimation of the interaction forces and to use external joint torques only for collision detection. A precise end-effector wrench estimation allows for better hand guiding or force control for instance while even an approximate estimation of the external joint torques is sufficient to detect unexpected collisions along the robot's body that might occur during pHRI and then take a preventive action (e.g. a *safety-rated monitored stop*).

2.2 Torque control of a Kuka LWR4+

As seen in the previous section, to estimate the interaction forces using joint torque sensors, the knowledge of the joints torque commands is necessary. Since this data is not generally offered by the robots embedded controllers, torque control must be performed on the user side. In this section, we will detail how torque control can be achieved on a Kuka LWR4+.

The Kuka LWR4+ controller (KRC) is shipped with three different control modes:

- joint position control,
- joint impedance control,
- cartesian impedance control.

It can be seen that no torque control mode is directly available. This means that workarounds should be implemented on both the KRC side and the PC control interface library FRI[1], to control the robot actuator torques. Among the three available control modes, joint impedance control is the only one that can be adapted to perform joint torque control. First, let us recall the joint impedance control equation, given in the KRC manual and FRI documentation, and rewritten for clarity as:

$$\boldsymbol{\tau}^* = \mathbf{K}_j \Delta\mathbf{q} + \mathbf{D}(\mathbf{d}_j) + \boldsymbol{\tau}_{dynamics}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{g}) + \boldsymbol{\tau}_{FRI}. \tag{2.9}$$

[1]FRI (Fast Research Interface), available at `http://cs.stanford.edu/people/tkr/fri/html/`

In this equation, $\boldsymbol{\tau}^* \in \mathbb{R}^7$ are the command torques sent to the actuators, $\mathbf{K}_j$ is a 7x7 diagonal matrix of stiffness parameters, $\mathbf{D} \in \mathbb{R}^7$ is a vector relative damping torques[1] parametrized by $\mathbf{d}_j \in [0,1]^7$, $\Delta\mathbf{q} = \mathbf{q}_r - \mathbf{q} \in \mathbb{R}^7$ is the tracking error, $\boldsymbol{\tau}_{dynamics}$ is the embedded dynamic model[2] and $\boldsymbol{\tau}_{FRI}$ is an additional input torque that can be set through FRI. In usual, $\boldsymbol{\tau}_{FRI}$ is set to zero in order to meet the expected joint impedance behavior but it is thanks to this variable that we can achieve torque control.

It is clear from (2.9) that to achieve torque control, all the terms on the right hand side of the equation, except for $\boldsymbol{\tau}_{FRI}$, must be canceled. This is done by setting $\mathbf{K}_j = \mathbf{0}$, $\mathbf{d}_j = \mathbf{0}\ \forall j = \{1, \dots, 7\}$ and $\mathbf{g} = \mathbf{0}$, where $\mathbf{g}$ is Earth's gravity vector configured in the KRC and used by $\boldsymbol{\tau}_{dynamics}$. Doing so leaves us with:

$$\boldsymbol{\tau}^* = \boldsymbol{\tau}_{FRI}. \tag{2.10}$$

Equation (2.10) can then be used to send the desired torque commands directly to the robot actuators. Modifications of the FRI library and KRC scripts to enable torque control are available online[3].

However, it is also important to note that the torque tracking accuracy on this robot is far from perfect, as can be seen from our tests, that are presented in figures 2.1 and 2.2. The first figure gives the average error and standard deviation for each joint torque. The second one displays the difference between the torques sent to the robot ($\boldsymbol{\tau}^*$) and the ones measured by the torque sensors ($\boldsymbol{\tau}$). The data in Fig. 2.2 has been prefiltered using a low pass 2nd order Butterworth filter with 10Hz cutoff frequency, to improve readability. Since Kuka provides no details about the KRC controller, there is no way to pin-point the exact cause of these errors and they are most probably tied to force sensor inaccuracies, unmodeled and/or uncompensated joint frictions and/or to the robot's torque regulation loop. Even if these errors may seem reasonable (< 1 N.m), they are problematic for motions with low accelerations producing low torque variations, as we will show in Sect. 2.3.

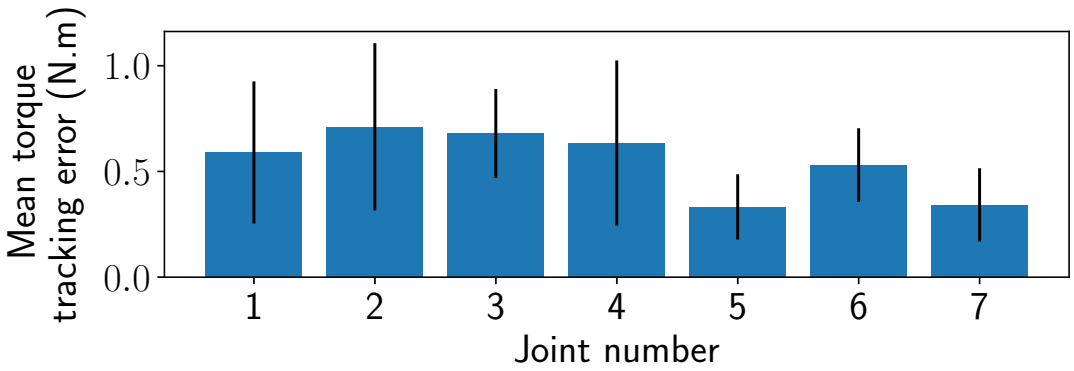


Figure 2.1 – Average torque tracking error and standard deviation on a Kuka LWR4+.

[1] The actual damping torques are computed relative to the stiffness coefficients.
[2] This is not detailed in the documentation, but probably corresponds only to gravity compensation.
[3] `https://github.com/BenjaminNavarro/api-driver-fri`

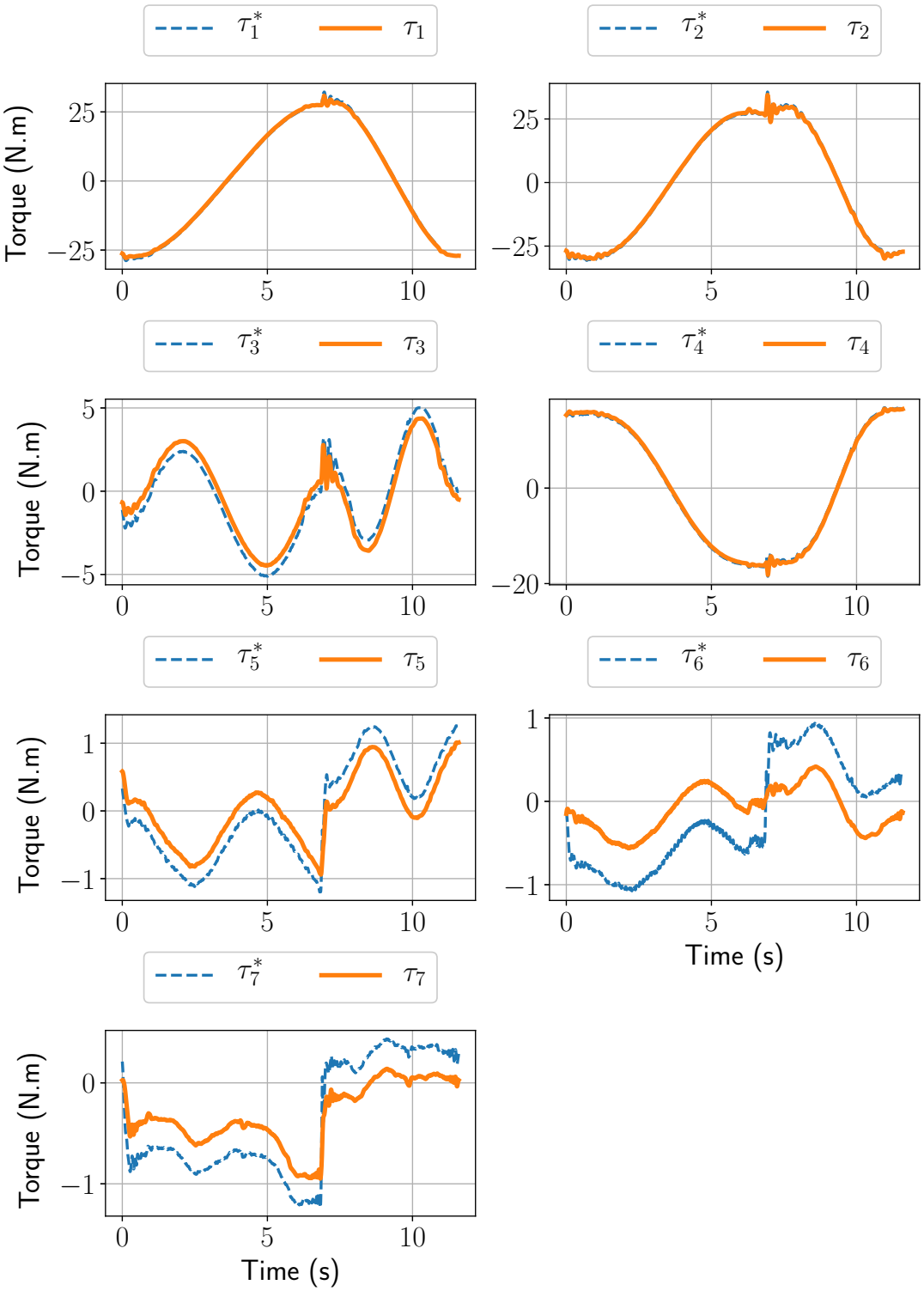


Figure 2.2 – Torque tracking accuracy on a Kuka LWR4+.

2.3 Torque control scheme

In this section, we detail the torque control scheme used to drive the Kuka LWR4+ to a target joint configuration.

A classic approach to realize joint positioning using torque control is the computed torque method [56]:

$$\boldsymbol{\tau}^* = \mathbf{M}(\mathbf{q})(\ddot{\mathbf{q}}_r + \mathbf{K}_p\Delta\mathbf{q} + \mathbf{K}_v\Delta\dot{\mathbf{q}}) + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \boldsymbol{\tau}_f(\dot{\mathbf{q}}). \tag{2.11}$$

In this equation, $\mathbf{K}_p$ and $\mathbf{K}_v$ are positive diagonal gain matrices used to apply position and velocity feedback in addition to the feedforward acceleration term $\ddot{\mathbf{q}}_r$. Vector $\boldsymbol{\tau}_f$ is generally expressed as:

$$\boldsymbol{\tau}_f(\dot{\mathbf{q}}) = \mathbf{F}_v\dot{\mathbf{q}} + \mathbf{F}_c\,\mathrm{sign}(\dot{\mathbf{q}}), \tag{2.12}$$

where $\mathbf{F}_v$ and $\mathbf{F}_c$ are the diagonal matrices of viscous and static friction coefficients, respectively. These coefficients must be identified on a robot basis.

However, using (2.12) is not a perfect real-world solution since stiction (static friction) torques are not constant across the robot configuration space. To demonstrate this, we set-up an experiment with a Kuka LWR4+ mounted horizontally with the joint axes laying in the horizontal plane so that the gravity does not interfere, leading to all joints being at their zero position. Then each joint is sequentially driven to five different positions (-2, -1, 0, 1 and 2 radians) before being brought back to zero using simple gravity compensation with PD control law:

$$\boldsymbol{\tau}^* = \mathbf{K}_p\Delta\mathbf{q} + \mathbf{K}_v\Delta\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) \tag{2.13}$$

At each position, the PD gains for the *test joint* are zeroed and the desired torque value is slowly increased, starting from 0 with 0.01 N.m steps, until a motion is detected. This way, the minimum positive torque needed to overcome stiction is estimated. The same is performed with reversed sign, to measure the minimum negative torque that overcomes stiction. To assess if, at a given position, the stiction torques are constant or not, the experiment was run three times. The results are given in figures 2.3 and 2.4.

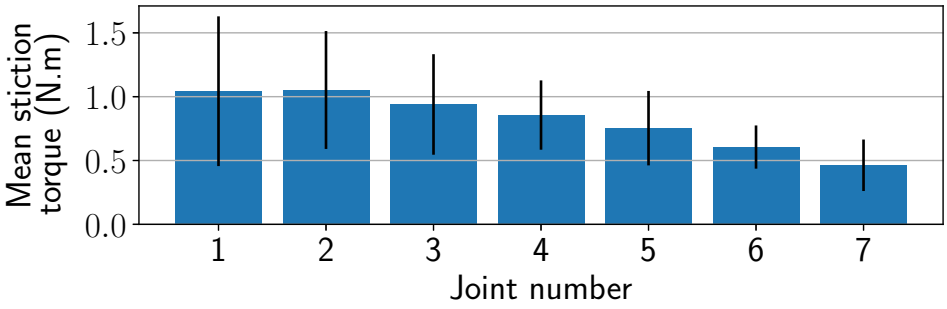


Figure 2.3 – Average stiction torques and standard deviation of a Kuka LWR4+.

Several important aspects can be derived from these graphs. First, for a given joint, the stiction torque is not constant. This forbids the use of constant values in $\mathbf{F}_c$ as it would be insufficient in some cases or lead to overcompensation and thus instability in the others. Secondly, for a given joint at a given position, the stiction torques are generally

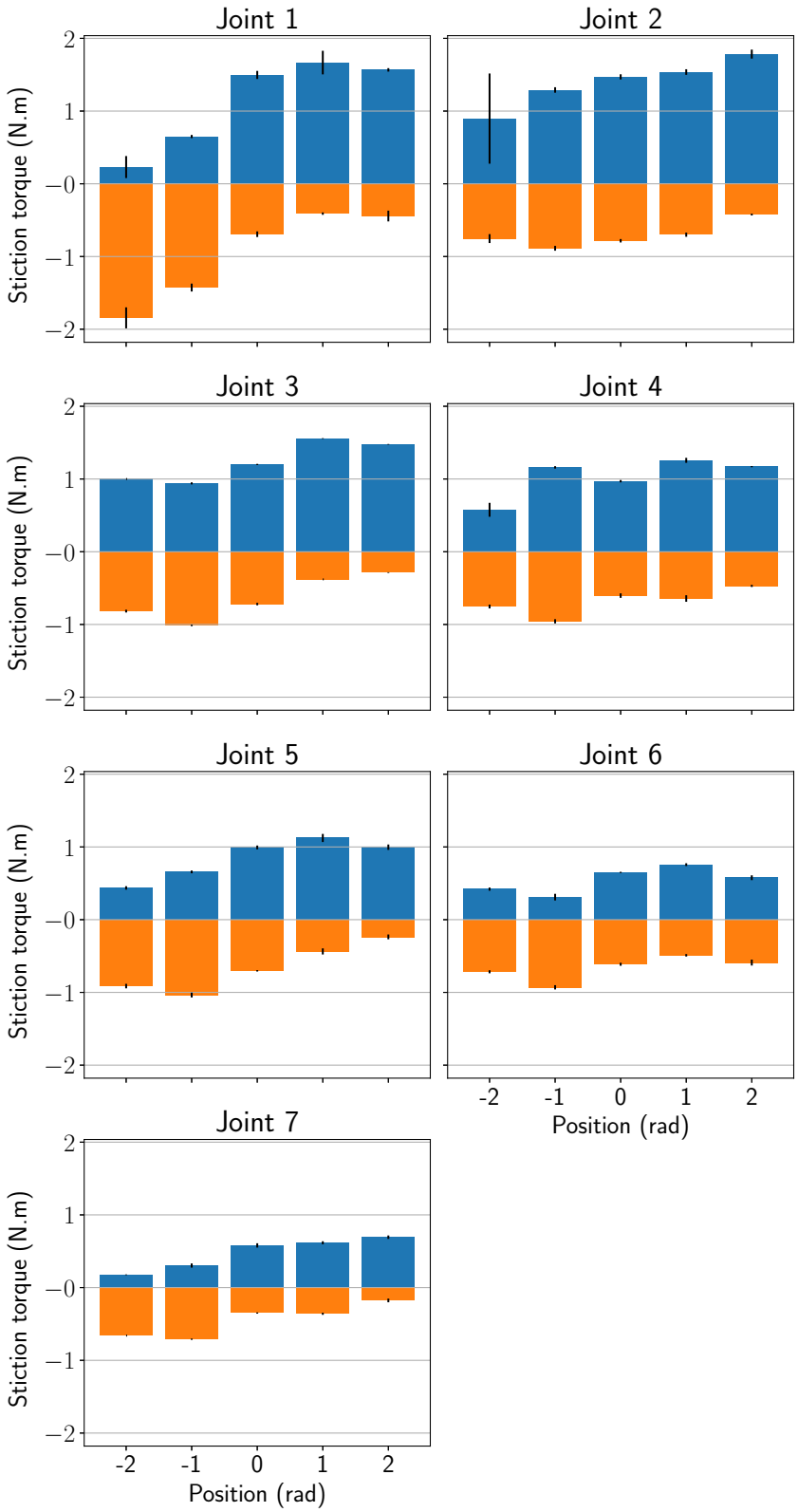


Figure 2.4 – Stiction torques for each joint of a Kuka LWR4+.

not symmetric. This avoids the use of the sign function in (2.12). Finally, important variations can be measured between experiments, as can be seen for example for joint 2

at -2 radians in figure 2.4. In summary, it is clear that the stiction torques cannot be compensated using (2.12) and that another solution must be considered.

In this regard, a first improvement over (2.11) would be to apply:

$$\boldsymbol{\tau}^* = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_r + \mathbf{K}_p\Delta\mathbf{q} + \mathbf{K}_v\Delta\dot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \mathbf{F}_v\dot{\mathbf{q}}. \tag{2.14}$$

With (2.14), the stiction is compensated using only the feedback terms ($\mathbf{F}_c = \mathbf{0}$) and without pre-multiplying them by the inertia matrix, thus producing always the same torque for a given tracking error. However, this may not be satisfactory in all cases. Indeed, to quickly compensate the deviations from the trajectory due to joint stiction, very high gains are required. This leads to a very stiff position control scheme. The solution we propose here is to decouple the stiction compensation problem from the more general perturbation rejection problem. The main idea is to use two PD control laws: a first one with very high gains and limited torque output to overcome stiction and a second one that can be tuned freely to obtain the desired perturbation rejection behavior.

For the friction compensation part, this translates to:

$$\boldsymbol{\tau}'_f(\dot{\mathbf{q}},\Delta\mathbf{q}) = \mathbf{F}_v\dot{\mathbf{q}} +_{-\boldsymbol{\tau}^c_{max}} \lfloor \mathbf{K}^c_p\Delta\mathbf{q} + \mathbf{K}^c_v\Delta\dot{\mathbf{q}} \rceil^{\boldsymbol{\tau}^c_{max}}. \tag{2.15}$$

Now, the static friction is compensated using a PD controller with gains $\mathbf{K}^c_p$ and $\mathbf{K}^c_v$ and output limited to the $[-\boldsymbol{\tau}^c_{max}, \boldsymbol{\tau}^c_{max}]$ range. This presents the advantage of not relying on any stiction parameters estimation and cannot lead to overcompensation.

We then propose two alternative control schemes, based on equations (2.11), (2.14) and (2.15):

$$\boldsymbol{\tau}^* = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_r + \mathbf{K}_p\Delta\mathbf{q} + \mathbf{K}_v\Delta\dot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \boldsymbol{\tau}'_f(\dot{\mathbf{q}},\Delta\mathbf{q}) \tag{2.16}$$

$$\boldsymbol{\tau}^* = \mathbf{M}(\mathbf{q})(\ddot{\mathbf{q}}_r + \mathbf{K}_p\Delta\mathbf{q} + \mathbf{K}_v\Delta\dot{\mathbf{q}}) + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \boldsymbol{\tau}'_f(\dot{\mathbf{q}},\Delta\mathbf{q}). \tag{2.17}$$

The choice between (2.16) and (2.17) depends on the desired perturbation rejection behavior. With the first equation, an opposing torque will be created when a tracking error appears. This enables the limitation of the torque transmitted to the environment by choosing safe fixed values for $\mathbf{K}_p$ and $\mathbf{K}_v$ or even tuning their values online to keep the exerted torque below a given limit. However, if a consistent perturbation rejection behavior is desired, the second control law is recommended, since the feedback action is passed through the inertia matrix and will produce the same motion, regardless of the current robot configuration. These two control schemes are depicted in Fig. 2.5, where the choice between one and the other is made through the S_{pos} switch. This switch should not be toggled online as this can lead to discontinuities in the control output.

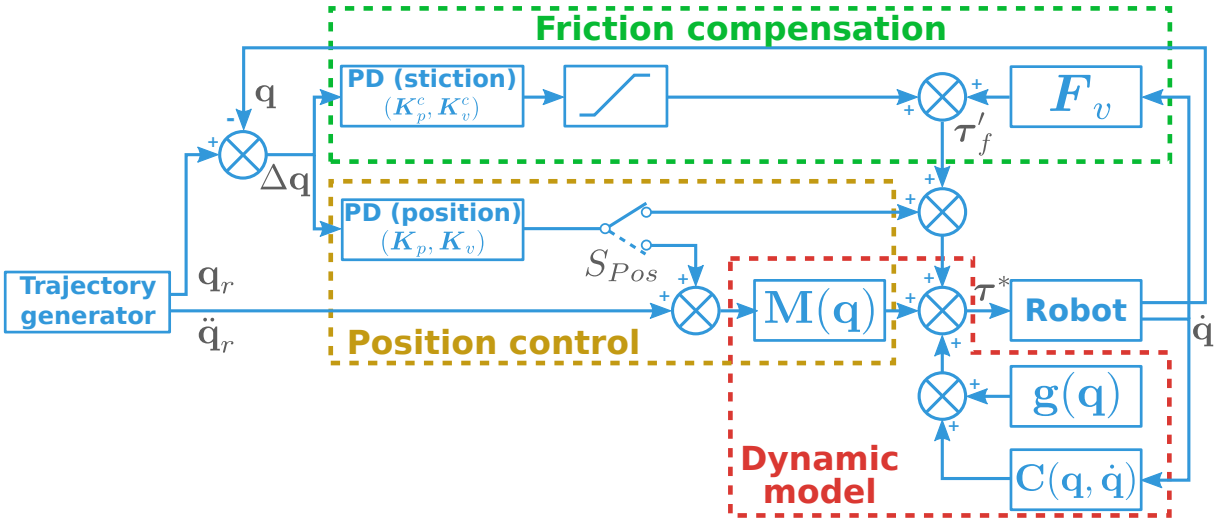


Figure 2.5 – Proposed torque control scheme with stiction compensation.

2.4 Experiments

To demonstrate the effectiveness of the proposed solution, a trajectory following experiment has been conducted using (2.16) in the following scenarios:

Case 1: All feedback gains set to zero ($\mathbf{K}_p = \mathbf{K}_v = \mathbf{K}_p^c = \mathbf{K}_v^c = \mathbf{0}$).

Case 2: Static PD friction compensation only ($\mathbf{K}_p = \mathbf{K}_v = \mathbf{0}, \mathbf{K}_p^c \neq \mathbf{0}, \mathbf{K}_v^c \neq \mathbf{0}$).

Case 3: Static PD friction compensation only with a collision (human operator, $\mathbf{K}_p = \mathbf{K}_v = \mathbf{0}$, $\mathbf{K}_p^c \neq \mathbf{0}, \mathbf{K}_v^c \neq \mathbf{0}$).

Case 4: Static PD friction compensation and PD control with a collision (human operator, $\mathbf{K}_p \neq \mathbf{0}$, $\mathbf{K}_v \neq \mathbf{0}$, $\mathbf{K}_p^c \neq \mathbf{0}$, $\mathbf{K}_v^c \neq \mathbf{0}$).

(2.16) has been chosen over (2.17) in this case to have a non-configuration dependent reaction to externally applied torques. The generated trajectory drives all the joints from -1 to 1 radian with a maximum acceleration of 0.5 rad.s^{-2} and a maximum velocity of 0.5 rad.s^{-1}. The values of $\mathbf{M}(\mathbf{q})$, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, $\mathbf{g}(\mathbf{q})$ and $\mathbf{F}_v$ in (2.16) were obtained by dynamic model identification. The friction compensation parameters, when used, are $\mathbf{K}_p^c = \text{diag}(10000, 5000, 2000, 2000, 500, 200, 200)$, $\mathbf{K}_v^c = \text{diag}(50, 100, 30, 30, 20, 10, 10)$ and $\boldsymbol{\tau}_{max}^c = [3, 4, 2, 2, 2, 1, 1]$. These parameters were manually tuned to try to get the best possible compensation. The numbering of the Kuka LWR4+ joints is given in Fig. 2.6.

Results from the first test (case 1) are shown in figures 2.7 and 2.8. In Fig. 2.8, gravity torques as well as torques generated from the other feedforward terms are displayed separately. It can be seen that the torques computed to perform the motion (i.e. excluding gravity compensation) are low and, considering the problems mentioned previously, are not sufficient to drive the joints to the target positions. This is confirmed by Fig. 2.7, where the target and actual position of each joint is plotted. Indeed, except for joint 2 where the braking torque makes the joint move in the wrong direction towards the end of the trajectory, all joints stay stationary. This clearly demonstrates the need for a stiction compensation mechanism.

Figures 2.9 and 2.10 give the results for the second experiment (case 2). We notice from Fig. 2.9 the presence of the friction compensation torques. Thanks to these additional torques, the trajectory can be followed way more precisely than in the previous case, as can be seen in Fig. 2.10. We can see that at the beginning of the trajectories, the errors start to grow but are kept low thanks to the friction compensation mechanism described in (2.15). In this particular case, the errors generally stay below 0.005 radians (0.3 degrees), except for joint 2 where the error goes up to 0.008 radians (0.45 degrees) before being brought back to 0.

The next experiment (case 3) uses the same configuration as the previous one but includes a collision with a human operator at the end-effector. Results are given in figures 2.11-2.13. As expected, the collision induces large tracking errors (Fig. 2.11) while keeping the external torques relatively low, with a maximum of 12 N.m at around 2s. (Fig. 2.13). The behavior is very close to an ideal control with no feedback. However, the presence of the friction compensation torques (Fig. 2.12) makes the robot slowly converge to the target when the perturbation disappears (Fig. 2.11).

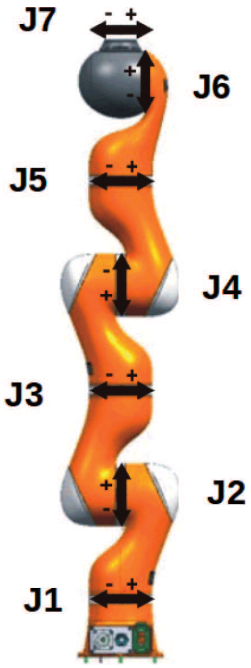


Figure 2.6 – Joints numbering of a Kuka LWR4+[1]

In the last experiment (case 4), we introduce the PD controller to enhance the position tracking accuracy using the following gains: $\mathbf{K}_p = \text{diag}(150, 500, 100, 200, 50, 20, 20)$, $\mathbf{K}_v = \text{diag}(5, 5, 3, 3, 2, 1, 1)$. These gains are tuned so that the robot still presents some compliance. It can be seen from Fig. 2.14 that the tracking error is improved, even in the presence of a collision. Since the robot really reacts to the collision by generating high torques, the external torques are also higher than previously, as seen in figures 2.15 and 2.16.

These experiments confirm that special care is required when dealing with real world static friction and that the proposed approach allows for an effective compensation without

[1]Source [57].

enforcing any perturbation rejection behavior, i.e. the robot can be as soft or as stiff as desired. A video of these experiments is joint to the manuscript[2].

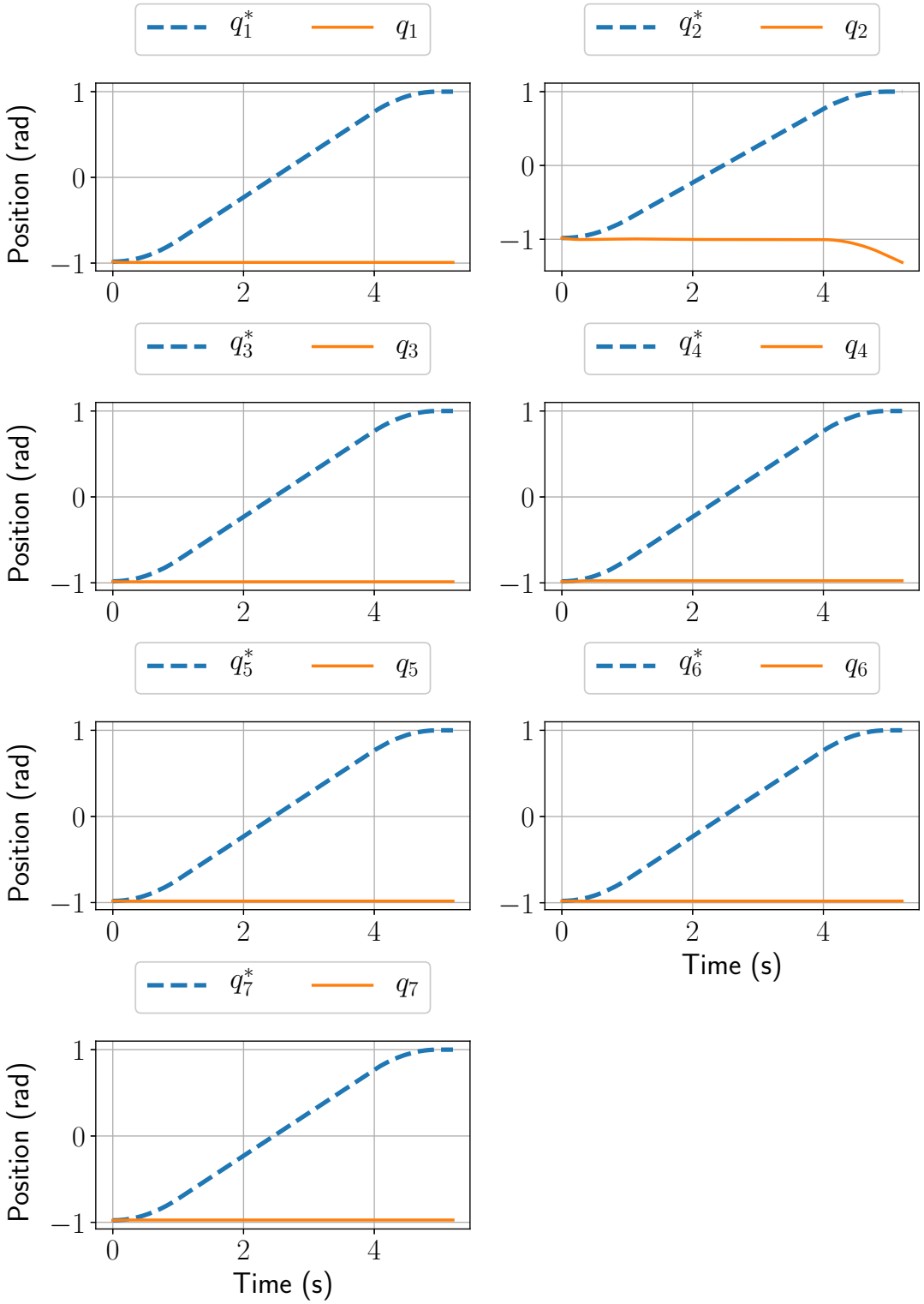


Figure 2.7 – Position command $\mathbf{q}^*$ and response $\mathbf{q}$, feedforward only (case 1).

[2]Also avaible at `https://youtu.be/fvHh1080I5I`

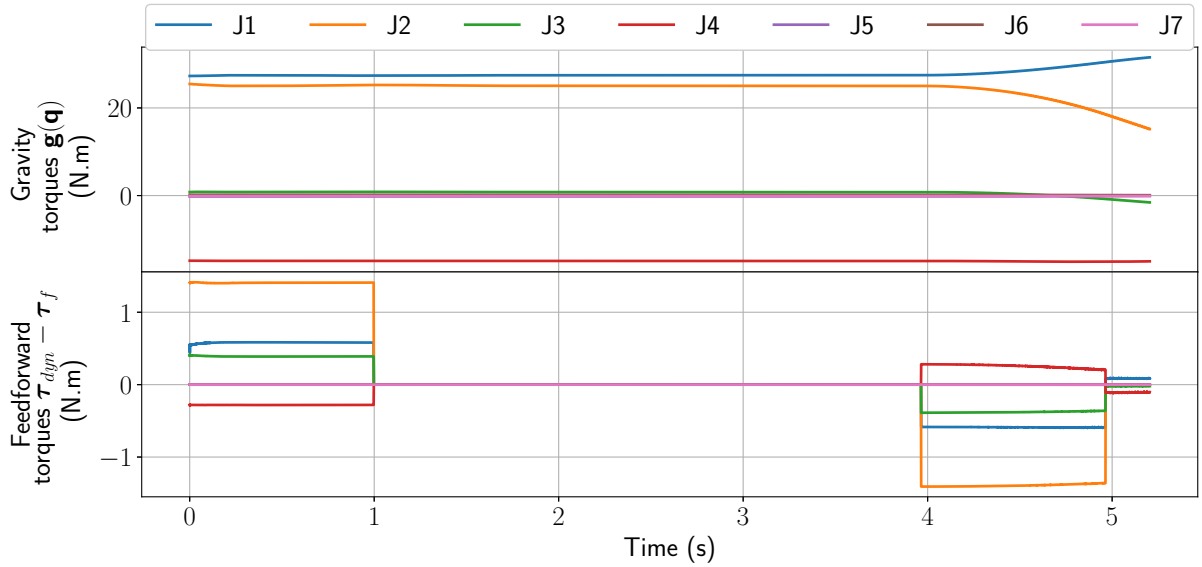


Figure 2.8 – Torque commands, feedforward only (case 1).

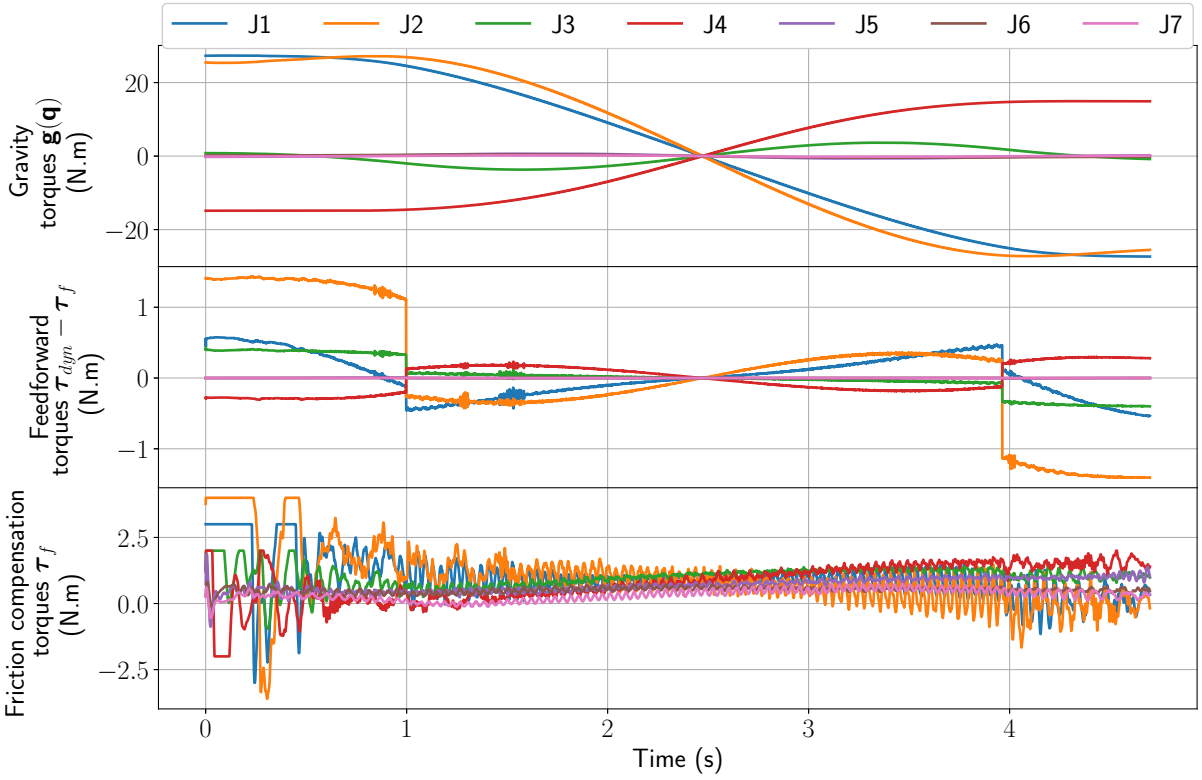


Figure 2.9 – Torque commands, feedforward with friction compensation (case 2).

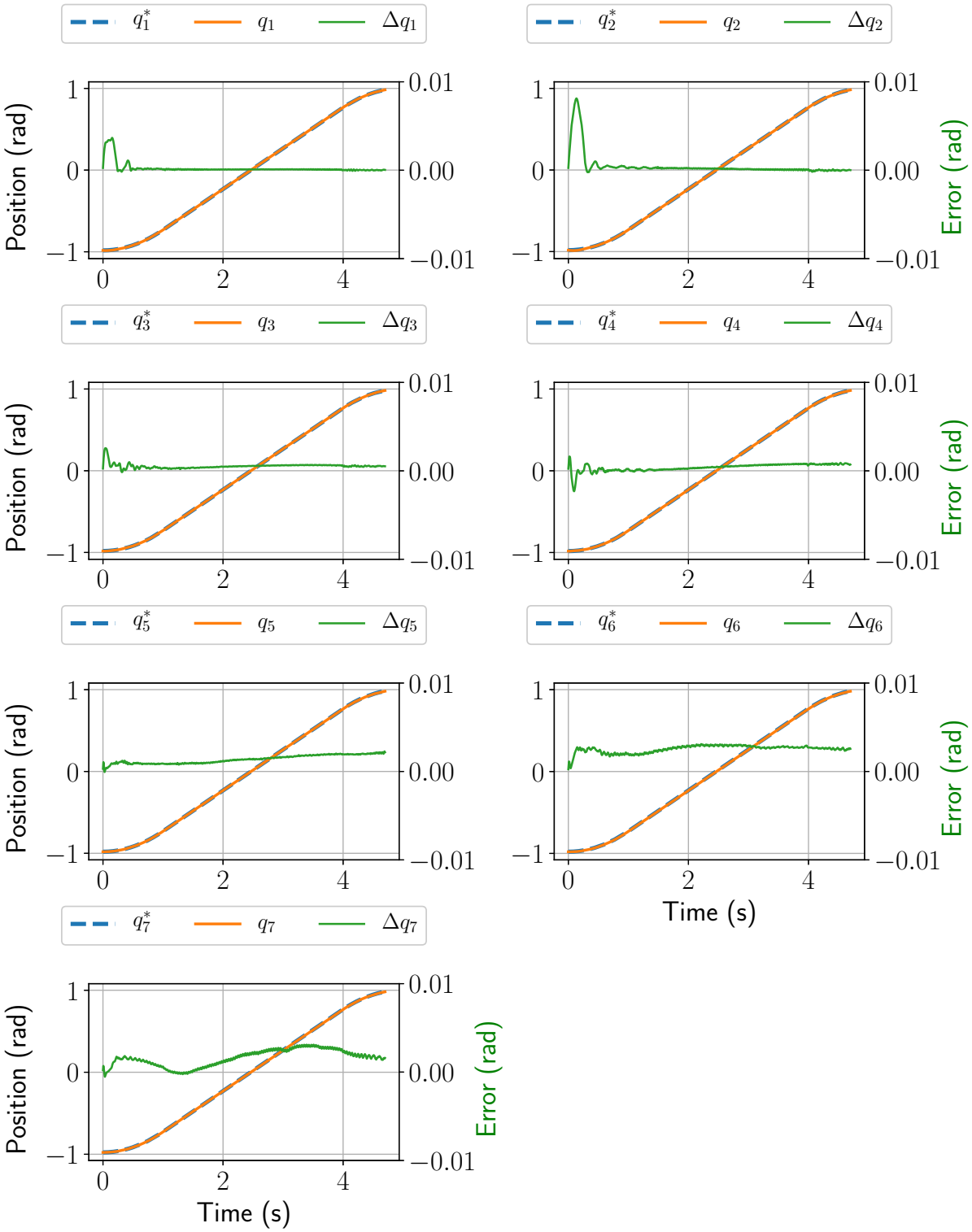


Figure 2.10 – Position command and response, feedforward with friction compensation (case 2).

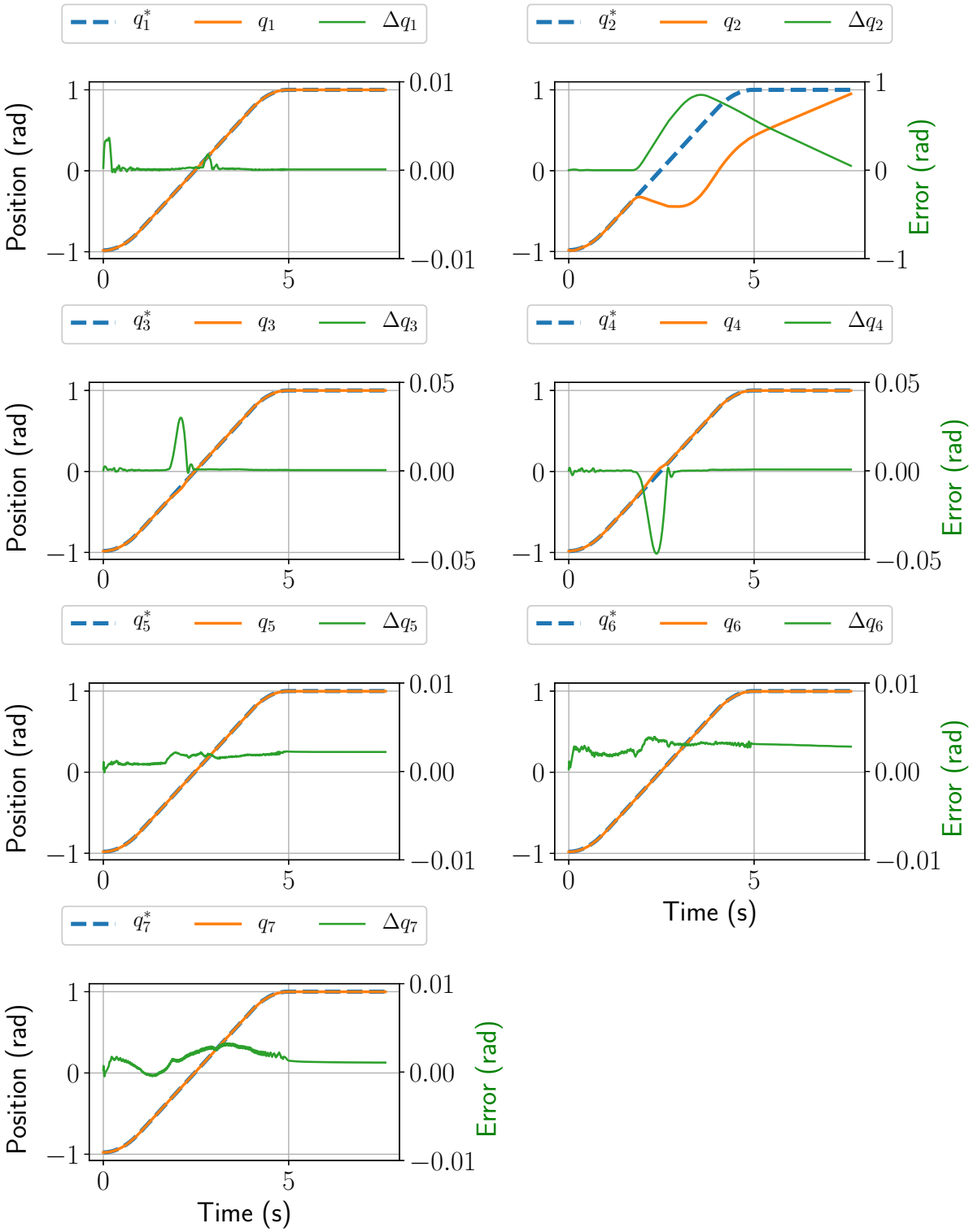


Figure 2.11 – Position command and response, feedforward with friction compensation and a collision (case 3).

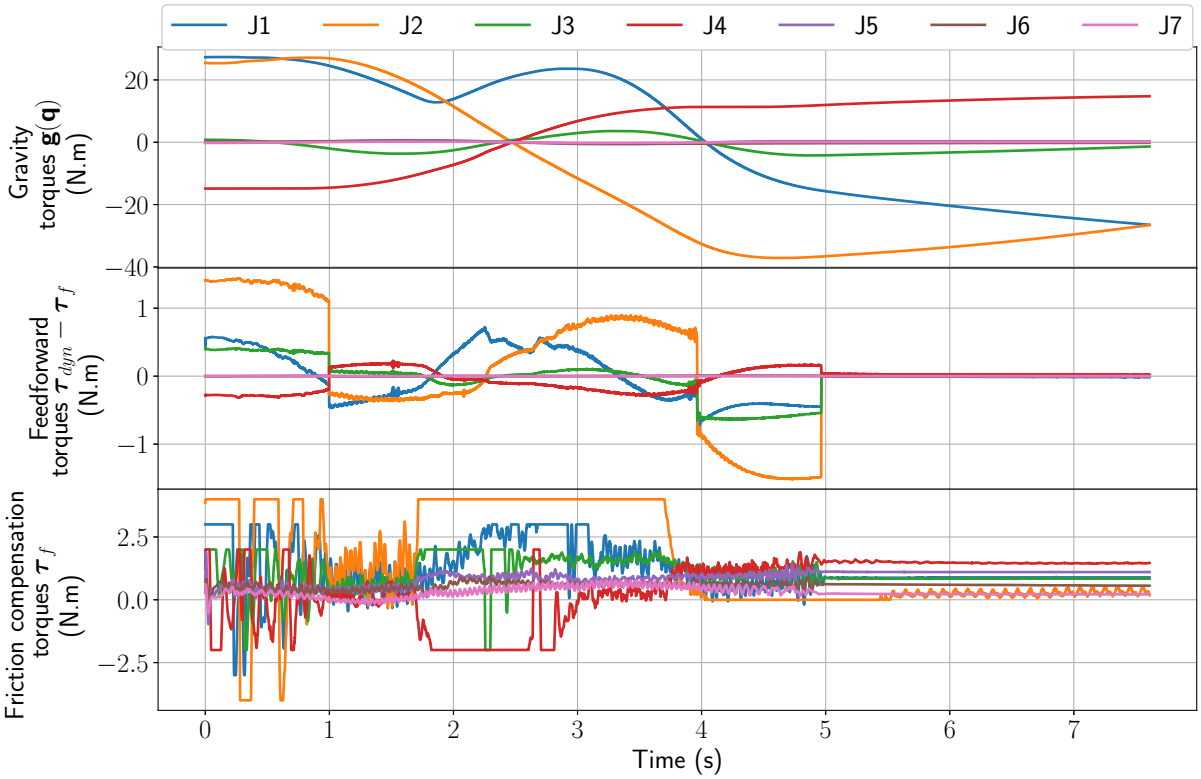


Figure 2.12 – Torque commands, feedforward with friction compensation and a collision (case 3).

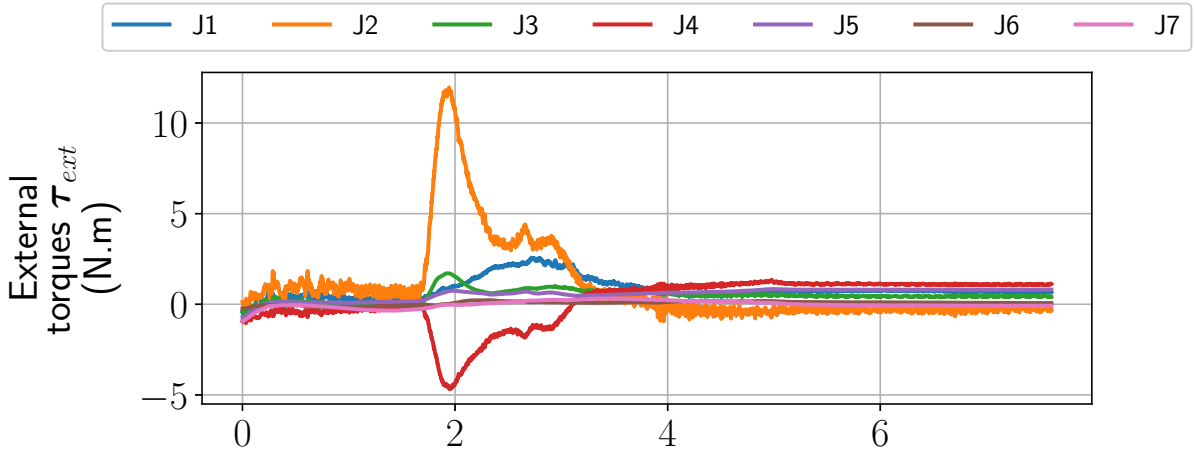


Figure 2.13 – External torques, feedforward with friction compensation and a collision (case 3).

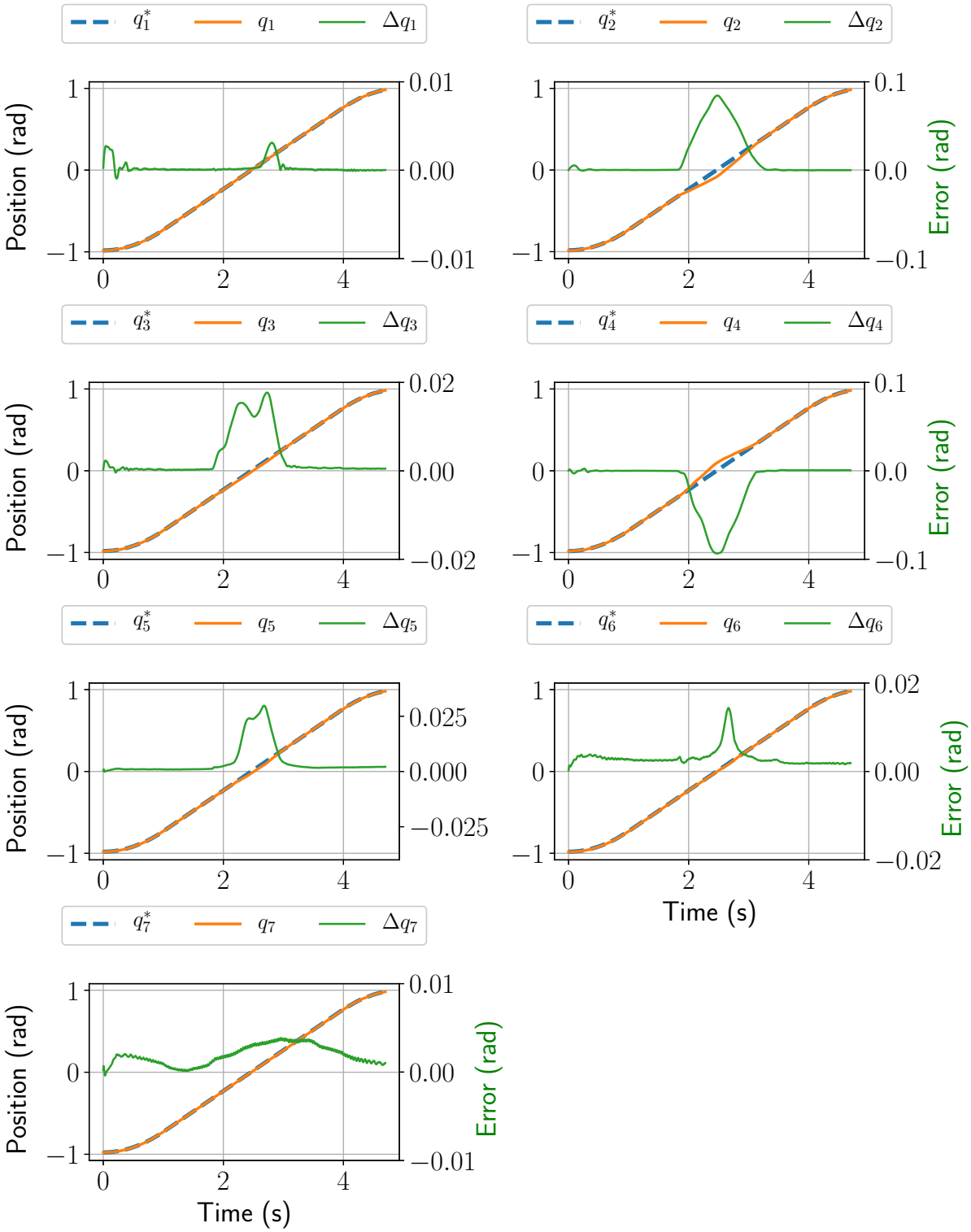


Figure 2.14 – Position command and response, feedforward with friction compensation plus PD control and a collision (case 4).

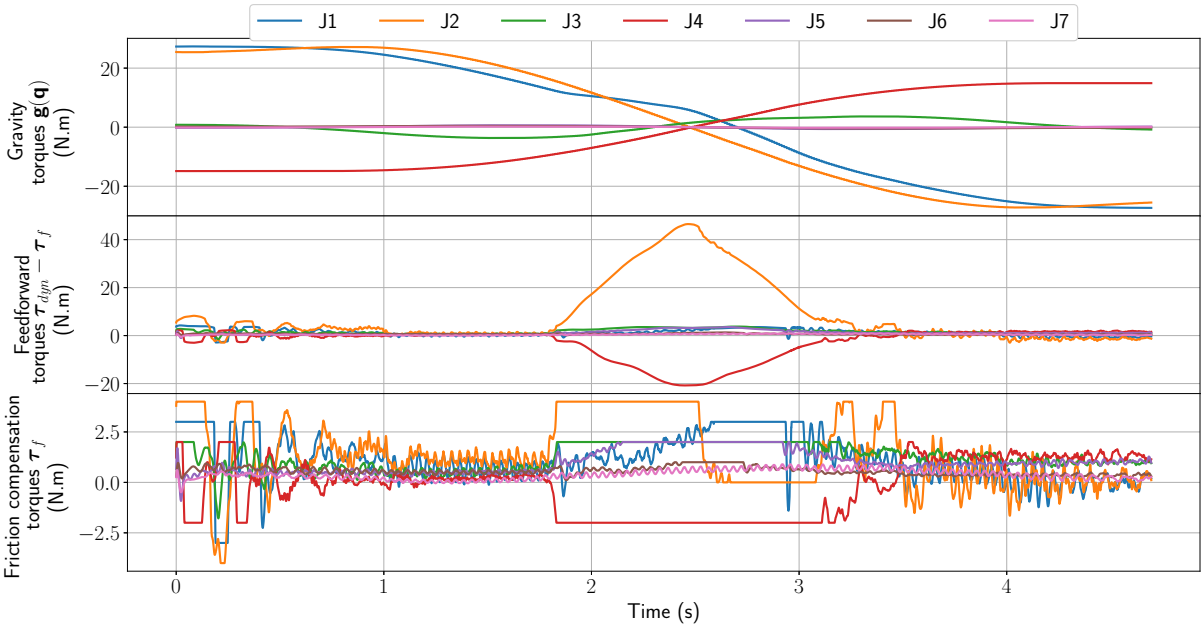


Figure 2.15 – Torque commands, feedforward with friction compensation plus PD control and a collision (case 4).

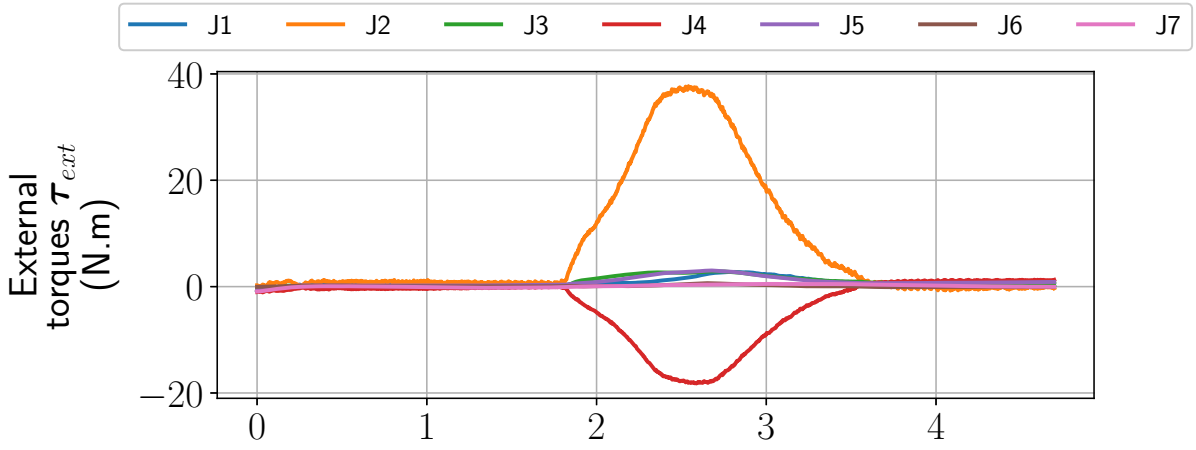


Figure 2.16 – External torques, feedforward with friction compensation plus PD control and a collision (case 4).

2.5 Conclusion

In this chapter, we discussed about the importance of having a dynamic model including static friction to provide external forces and torques estimation and to enable low level torque control. It was noticed that static friction cannot be compensated using a Coulomb friction model, since it is not uniform across the joint space. Instead, we proposed the use of an output-limited PD stiction controller to quickly compensate deviations from the trajectory. This compensator was incorporated into a computed torque scheme to enable precise joint positioning. Experiments showed that a good stiction compensation strategy is required otherwise the joints stay still when subject to low acceleration profiles. The proposed solution was proven effective to quickly compensate for the stiction related errors without enforcing a stiff joint positioning.

In the next chapter, we will expose a framework for safe human-robot physical collaboration. In this framework, the control output is the vector of target joint positions that can be used as input to (2.16). The internal position controller of the Kuka LWR4+ is very sensitive to the input and could not be used to achieve the fast reactive motions often required during pHRI. Instead, the use of the proposed controller allowed to realize any motion within the mechanical limits of the robot.

Chapter 3

Task space control solutions

In this chapter, we present a novel controller tailored for safe physical human-robot interaction and collaboration. This controller, being kinematics-based, outputs a joint velocity vector used as an input for the torque control scheme presented in the previous chapter (Fig. 2.5).

This controller can take into account velocities and forces (real or virtual ones, e.g. repulsive force) at both the joint and task level, along with a velocity reduction method to ensure a set of safety criteria. Sect. 3.1 presents the controller, starting with its derivation from impedance control [58], then showing how joint and task space motions are fused and how safety constraints can be incorporated. Sect. 3.2 and Sect. 3.3 detail some possible force and velocity inputs that can be fed to the controller. Then, various safety constraints are presented in Sect. 3.4, ranging from simple emergency stop to task space kinetic energy limitation. An overview of the open source software implementation of the methods presented in this chapter is given in Sect. 3.5. Finally, an experiment featuring the controller in a mock-up collaborative scenario is presented in Sect. 3.6.

We consider a robot (open kinematic chain) with j degrees of freedom (dof) and one control point (CP) on the terminal segment (TS), and denote: $\mathbf{q} \in \mathbb{R}^j$ its joint values, ${}^B\mathbf{x} \in \mathbb{SE}(3)$ the TS pose at the CP, and ${}^B\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{v}^\top & \boldsymbol{\omega}^\top \end{bmatrix}^\top$ the TS velocities, both expressed in the robot base frame (B). Velocities can be mapped between the CP frame (T) and the base frame using:

$$
\begin{aligned}
{}^B\dot{\mathbf{x}} &= {}^B\mathbf{V}_T \, {}^T\dot{\mathbf{x}} && (3.1)\\
{}^T\dot{\mathbf{x}} &= {}^T\mathbf{V}_B \, {}^B\dot{\mathbf{x}} = {}^B\mathbf{V}_T^\top \, {}^B\dot{\mathbf{x}}, && (3.2)\\
& && (3.3)
\end{aligned}
$$

with ${}^B\mathbf{V}_T$ being the spatial motion transform matrix defined by:

$$
{}^B\mathbf{V}_T = \begin{bmatrix} {}^B\mathbf{R}_T & \mathbf{0}^3 \\ \mathbf{0}^3 & {}^B\mathbf{R}_T \end{bmatrix}, \tag{3.4}
$$

where ${}^B\mathbf{R}_T$ is the rotation matrix between the CP and the base frame. The mapping between joint space and task space velocities is achieved using the Jacobian matrix $\mathbf{J} \in \mathbb{R}^{6\times j}$:

$$
{}^B\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}. \tag{3.5}
$$

However, to obtain the inverse mapping, the inversion of the Jacobian matrix must be performed and it is well known that kinematic singularities can lead to instabilities if the Jacobian's (pseudo-)inverse is used. When interacting with a human operator, there is no guarantee that s/he does not bring the robot in singular configurations, which might be a threat to his safety, as the Jacobian's inversion is not properly handled. To solve this issue, we use, as in [59], an adaptive damped least squares pseudo-inverse instead of the classical one:

$$\mathbf{J}^{\dagger} = \mathbf{J}^{\top}(\mathbf{J}\mathbf{J}^{\top} + \lambda^2 \mathbf{I}^6)^{-1}, \tag{3.6}$$

with λ^2 being calculated as:

$$\lambda^2 = \begin{cases} 0 & \text{if } \sigma_m \geq \varepsilon, \\ (1 - (\frac{\sigma_m}{\varepsilon})^2)\lambda^2_{max} & \text{otherwise.} \end{cases} \tag{3.7}$$

In (3.7), σ_m is the smallest singular value of $\mathbf{J}$, which can be obtained from its singular values decomposition, ε is a threshold that activates the damping effect, and λ_{max} is the maximum value for λ. The advantage of using (3.7) over a constant value for λ is that the arm performance is not degraded away from singularity.

3.1 Two-layer safe damping control framework

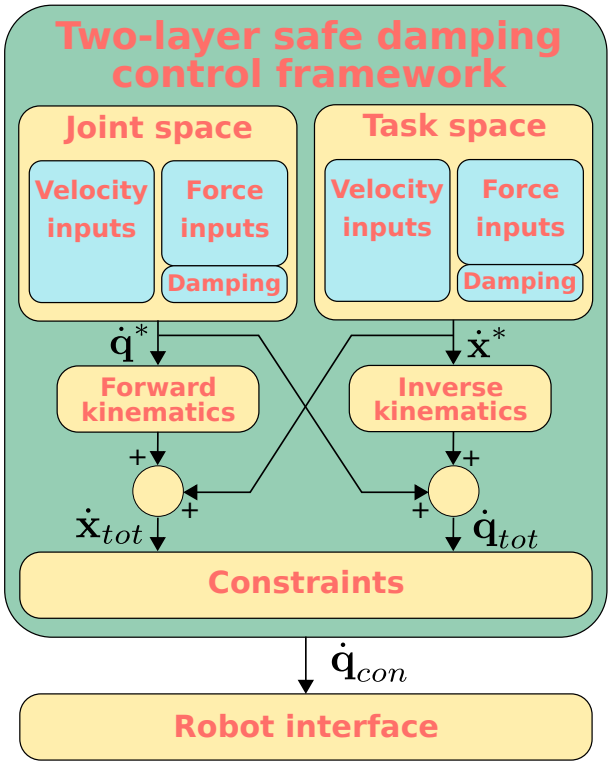


Figure 3.1 – Overview of the proposed controller.

The controller proposed in this section, outlined in Fig. 3.1, is based on damping control, a special case of impedance control [58]. Impedance control relies in general on

a mass-spring-damper system, as reminded in (3.8), that relates the forces $\mathbf{f}^*$ applied on the CP with its displacement $\Delta\mathbf{x}$ from the reference pose. The impedance system is:

$$\mathbf{f}^* = \mathbf{K}_t\Delta\mathbf{x} + \mathbf{B}_t\Delta\dot{\mathbf{x}} + \mathbf{M}_t\Delta\ddot{\mathbf{x}}, \tag{3.8}$$

with $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}_r$, $\mathbf{K}_t, \mathbf{B}_t, \mathbf{M}_t \in \mathbb{R}^{6\times6}$ diagonal positive matrices of stiffness, damping and mass parameters respectively. Here, we consider only damping, and extend the paradigm to both task and joint spaces, using:

$$\mathbf{f}^* = \mathbf{B}_t\Delta\dot{\mathbf{x}} \tag{3.9}$$

$$\boldsymbol{\tau}^* = \mathbf{B}_j\Delta\dot{\mathbf{q}}, \tag{3.10}$$

with $\mathbf{B}_j \in \mathbb{R}^{j\times j}$ a diagonal positive matrix of damping parameters, $\Delta\dot{\mathbf{x}} = \dot{\mathbf{x}} - \dot{\mathbf{x}}_r$ and $\Delta\dot{\mathbf{q}} = \dot{\mathbf{q}} - \dot{\mathbf{q}}_r$ the errors between the current and reference velocities. Vectors $\mathbf{f}^*$ and $\boldsymbol{\tau}^*$ indicate the forces at the CP and joint level, respectively. The generic term force will be used when dealing with both forces and torques at the CP and joint levels in order to simplify both the notations and the explanations. Also, throughout this chapter, subscripts t and j indicate respectively task space and joint space related variable.

Equations (3.9) and (3.10) can be rewritten in order to output velocity commands based on input (i.e., measured) forces and reference velocities:

$${}^T\dot{\mathbf{x}}^* = \mathbf{B}_t^{-1}\ {}^T\mathbf{f}_{ext} + {}^T\dot{\mathbf{x}}_r. \tag{3.11}$$

$$\dot{\mathbf{q}}^* = \mathbf{B}_j^{-1}\ \boldsymbol{\tau}_{ext} + \dot{\mathbf{q}}_r. \tag{3.12}$$

While (3.11) and (3.12) were proven to be useful to comply with interaction forces while following a predefined trajectory, they can be extended to fit many more scenarios, as we will show in this chapter.

To this end, we build a more generic two-layer controller that includes sets of force inputs $\mathcal{F}$ and Γ (torques) and of velocity inputs $\mathcal{V}$ and ω (angular velocities):

$${}^T\dot{\mathbf{x}}^* = \mathbf{B}_t^{-1}\sum_{\mathbf{f}_i\in\mathcal{F}} {}^T\mathbf{f}_i + \sum_{\dot{\mathbf{x}}_i\in\mathcal{V}} {}^T\dot{\mathbf{x}}_i \tag{3.13}$$

$$\dot{\mathbf{q}}^* = \mathbf{B}_j^{-1}\sum_{\boldsymbol{\tau}_i\in\Gamma} \boldsymbol{\tau}_i + \sum_{\dot{\mathbf{q}}_i\in\omega} \dot{\mathbf{q}}_i, \tag{3.14}$$

with $|\mathcal{F}|, |\mathcal{V}|, |\Gamma|, |\omega| \in \mathbb{N}$. In this framework, the total task and joint velocities can be computed with:

$${}^T\dot{\mathbf{x}}_{tot} = {}^T\dot{\mathbf{x}}^* + {}^T\mathbf{V}_B\mathbf{J}\dot{\mathbf{q}}^* \tag{3.15}$$

$$\dot{\mathbf{q}}_{tot} = \mathbf{J}^\dagger\ {}^B\mathbf{V}_T\ {}^T\dot{\mathbf{x}}^* + \dot{\mathbf{q}}^*. \tag{3.16}$$

It is important to note that (3.15) and (3.16) are related by:

$${}^T\dot{\mathbf{x}}_{tot} = {}^T\mathbf{V}_B\mathbf{J}\dot{\mathbf{q}}_{tot}, \tag{3.17}$$

and thus represent the same motion expressed in two different spaces. Both (3.15) and (3.16) are needed since, for example, one can design a trajectory at the joint level in ω and add some compliance at the CP by including the external force in $\mathcal{F}$.

With our method, real world forces can be combined with virtual ones, and it is possible to add velocity sources in $\mathcal{V}$ and ω other than the reference – real – joint or task space trajectories. In this work, the focus has been on :

- interaction forces,
- virtual mass and stiffness effects,
- attractive and repulsive forces generated by potential fields,
- velocities generated by a trajectory generator and a force control law.

This is of course not restrictive and many other inputs can be considered to fit more scenarios.

When considering safety during human-robot interaction, most solutions can be expressed as some form of velocity reduction. This includes: stopping the robot upon contact, reducing its velocity when nearby operators are approaching, and imposing constraints on velocity, kinematic energy or exchanged power. To cope with these issues, we consider the following *velocity scaling factor* $\alpha \in [0, 1]$:

$$\alpha = \min(1, \min(\mathcal{C})). \tag{3.18}$$

with $\mathcal{C}$ the set of constraints $\mathcal{C}_i \in \mathbb{R}_{\geq 0}$ to fulfill, with $|\mathcal{C}| \in \mathbb{N}$.

This is finally used to reduce (if needed) the joint velocity that is sent to the robot actuators:

$$\dot{\mathbf{q}}_{con} = \alpha \ \dot{\mathbf{q}}_{tot}. \tag{3.19}$$

Equations (3.13) - (3.19) make up our generic framework for safe physical human-robot interaction and collaboration. In the next sections, we will detail the different constraints and control inputs that have been considered in our work.

3.2 Force inputs

This section presents joint or task space force inputs that, when included respectively in sets $\mathcal{F}$ in (3.13) and Γ in (3.14), let the robot comply with real world forces or react to virtual ones. An illustrative example is given in Fig. 3.2.

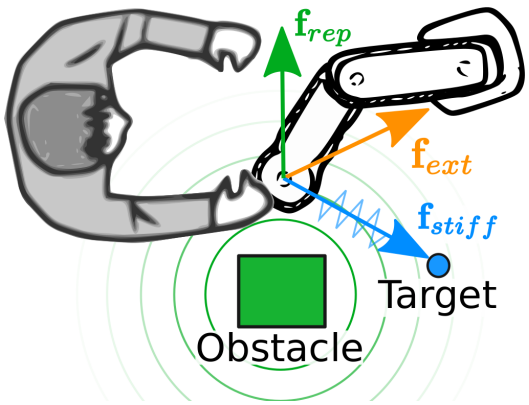


Figure 3.2 – Examples of interaction, stiffness and repulsive forces.

3.2.1 Interaction forces

In many cases, it is necessary to adapt the robot motion in the presence of external forces, e.g for kinesthetic guidance (*teaching by demonstration*). In such scenarios, the external force $\mathbf{f}_{ext}$ can be included in $\mathcal{F}$. If this is the only force input in $\mathcal{F}$, the controller is a classic damping controller. The same can be done at the joint level by including $\boldsymbol{\tau}_{ext}$ into Γ.

3.2.2 Virtual stiffness and mass

Using a full impedance model, including stiffness and mass effects in (3.9) has been intensively investigated in the literature and has proven useful in many cases [58, 60–62]. Let us first recall the complete impedance law [58]:

$$\mathbf{f} = \mathbf{K}_t \Delta\mathbf{x} + \mathbf{B}_t \Delta\dot{\mathbf{x}} + \mathbf{M}_t \Delta\ddot{\mathbf{x}}, \tag{3.20}$$

where $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}_r$, $\Delta\ddot{\mathbf{x}} = \ddot{\mathbf{x}} - \ddot{\mathbf{x}}_r$ and $\mathbf{K}$ and $\mathbf{M}$ are diagonal positive semi-definite matrices of stiffness and mass parameters, respectively. (3.20) can be rewritten in the following form:

$$\dot{\mathbf{x}} = \mathbf{B}_t^{-1}(\mathbf{f} - \mathbf{K}_t \Delta\mathbf{x} - \mathbf{M}_t \Delta\ddot{\mathbf{x}}) + \dot{\mathbf{x}}_r. \tag{3.21}$$

We can then derive the two force inputs needed to emulate (3.21) in our controller (3.13):

$$\mathbf{f}_{x,stiff} = -\mathbf{K}_t \Delta\mathbf{x} \tag{3.22}$$

$$\mathbf{f}_{x,mass} = -\mathbf{M}_t \Delta\ddot{\mathbf{x}}. \tag{3.23}$$

Using $\mathcal{F} = \{\mathbf{f}_{ext}, \mathbf{f}_{x,stiff}, \mathbf{f}_{x,mass}\}$ will result in classical admittance control. With a similar approach, stiffness and mass effects can be described at the joint level in (3.14) with:

$$\boldsymbol{\tau}_{q,stiff} = -\mathbf{K}_j \Delta\mathbf{q} \tag{3.24}$$

$$\boldsymbol{\tau}_{q,mass} = -\mathbf{M}_j \Delta\ddot{\mathbf{q}}. \tag{3.25}$$

3.2.3 Potential field method

If the robot has to avoid collisions with operators or if it has to perform a motion in a cluttered environment, a collision avoidance algorithm should be used. Here, we demonstrate how already existing obstacle avoidance mechanisms can be implemented in this framework. We will take the case of the potential field method (PFM) [63], where obstacles are sources of repulsive forces, and the target location is an attractive force source. Summing up all these forces results in a motion in the most promising direction, i.e the solution is not global and local minima may prevent the robot from reaching its goal. Dealing with this limitation would require a complete knowledge of the environment which is usually not available due to the use of limited field of view sensors (e.g., LIDAR or cameras) and of dynamic scenarios. The PFM can be summed up as:

$$\mathbf{f}_{att} = K_{att} \boldsymbol{\eta}_{att}, \tag{3.26}$$

$$\mathbf{f}_{rep} = \begin{cases} \sum_i K_{rep_i} \left(\frac{1}{d_{0_i}} - \frac{1}{d_{rep_i}} \right) \boldsymbol{\eta}_{rep_i} & \text{if } d_{rep_i} < d_0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.27}$$

In (3.26) and (3.27), K_{att} and K_{rep_i} are positive scalar gains, $\boldsymbol{\eta}_{att}$ and $\boldsymbol{\eta}_{rep_i}$ are unit vectors pointing from the robot to the target and to the i-th repulsive object, respectively. Further, d_{rep_i} is the distance to the i-th repulsive object and d_{0_i} is the distance at which the repulsive effect starts appearing. In this framework, $\mathbf{f}_{att}$ can be omitted if some stiffness effect $\mathbf{f}_{x,stiff}$ is already included since it will provide the same attractive effect. Since repulsive forces grow to infinity as $d_{rep} \to 0$, it is recommended to include the velocity constraint $\mathcal{C}_{x,vel}$ into the constraints set $\mathcal{C}$ to avoid excessively fast – hence dangerous – motions.

3.3 Velocity inputs

In this section, we describe possible joint and task space velocity inputs. These velocities can be, for instance, the result of a trajectory generator (Sect. 3.3.1) or of a force control law (Sect. 3.3.2).

3.3.1 Reference trajectory

Since trajectory generation and tracking are present in most robotics applications, it is crucial to have these in the presented framework. To this end, a polynomial-based trajectory generator has been developed and is detailed in Appendix A. It can be used to generate smooth joint or task space trajectories either with a given duration, or with given velocity and acceleration limits (in the latter case, the minimum duration is computed). Using (A.1) we can define the following inputs:

$$\dot{\mathbf{x}}_r(t) = \dot{\boldsymbol{\mathcal{P}}}(t, \mathbf{c}_x) \tag{3.28}$$

$$\dot{\mathbf{q}}_r(t) = \dot{\boldsymbol{\mathcal{P}}}(t, \mathbf{c}_q), \tag{3.29}$$

with $\mathbf{c}_x \in \mathbb{R}^{6\times 6}$ and $\mathbf{c}_q \in \mathbb{R}^{j\times 6}$ the coefficients of the task and joint space trajectories respectively. When needed, $\dot{\mathbf{x}}_r$ has to be included in $\mathcal{V}$ while $\dot{\mathbf{q}}_r$ has to be included in ω.

3.3.2 Force control

Force control is used extensively in various cases such as grinding, polishing, assembling, echographic monitoring, needle insertion or minimally invasive surgery. We will demonstrate how force control, or generally any control law outputting a velocity command can be used within our framework. Let us first define the target force $\mathbf{f}_r$ and its associated error vector:

$$\Delta\mathbf{f} = \mathbf{S}(\mathbf{f}_r - \mathbf{f}_{ext}), . \tag{3.30}$$

Here, $\mathbf{S}$ is a diagonal binary selection matrix with elements $\mathbf{S}(i,i) = \{0, 1\}$. It allows to select which task space components will be driven by the force controller. Then, PD control can be applied using (3.30) to compute the control point velocity to be included in $\mathcal{V}$:

$$\dot{\mathbf{x}}_{fc} = \mathbf{K}_P\Delta\mathbf{f} + \mathbf{K}_D\dot{\Delta}\mathbf{f}, \tag{3.31}$$

with $\mathbf{K}_P$ and $\mathbf{K}_D$ diagonal positive gain matrices. If $\Delta\mathbf{f}$ becomes large, high velocities can be generated. To mitigate this, velocity limitation (that will be presented in Sect. 3.4.2) can be included when force control is used.

3.4 Constraints

In this section, we will detail how different constraints (all of them being a form of velocity reduction) can be described in order to be included in our controller through (3.18).

3.4.1 Emergency stop

A simple way to provide some level of safety, is to stop the robot motion when a contact with a nearby operator occurs. This is mainly used in situations where a robot that relies only on proprioception works near humans and any physical contact between the two agents is prohibited. To provide such mechanism, we define the following constraint:

$$\mathcal{C}_{stop}(t) = \begin{cases} 0 & \text{if } \|\mathbf{f}_{ext}\| > \overline{F_{th}} \text{ or } \|\boldsymbol{\tau}_{ext}\| > \overline{\tau_{th}}, \\ 1 & \text{if } \|\mathbf{f}_{ext}\| < \underline{F_{th}} \text{ and } \|\boldsymbol{\tau}_{ext}\| < \underline{\tau_{th}}, \\ \mathcal{C}_{stop}(t - T_s) & \text{otherwise.} \end{cases} \tag{3.32}$$

In (3.32), $\mathbf{f}_{ext}$ and $\boldsymbol{\tau}_{ext}$ are the external task and joint forces applied to the robot, the upper and lower bars denote the activation and deactivation thresholds respectively and T_s is the controller sample time. Using (3.32) in (3.18) results in a complete stop of the robot when the external force passes one of the activation thresholds. Motion can be resumed only when both $\|\mathbf{f}_{ext}\|$ and $\|\boldsymbol{\tau}_{ext}\|$ go below the deactivation thresholds.

3.4.2 Velocity limitation

Another very common safety criterion is velocity limitation. This is often part of safety standards currently applied in robotics, such as the ISO10218-2011 [31]. Moreover, even with a carefully planed trajectory respecting the imposed velocity limitation, other inputs in the controller can lead the robot to an increase in velocity, breaking the safety requirements. To deal with this and respect any velocity limitation in any situation, we define the following constraint:

$$\mathcal{C}_{x,vel} = \begin{cases} \frac{V_{max}}{\|\mathbf{v}_{tot}\|} & \text{if } \|\mathbf{v}_{tot}\| > 0 \\ 1 & \text{otherwise,} \end{cases} \tag{3.33}$$

where $V_{max} \in \mathbb{R}_{>0}$ is the maximum velocity allowed. By using this formulation, the value of $\mathcal{C}_{x,vel}$ will stay above 1 as long as the robot total velocity is lower than V_{max}, hence will have no affect in (3.18).

Similarly, joint velocities can also be limited, e.g., to respect the mechanical constraints of the robot's actuators. To do so, the following constraint can be applied:

$$\mathcal{C}_{q,vel} = \min\left(\dot{\mathbf{q}}_{max} \oslash |\dot{\mathbf{q}}_{tot}|\right), \tag{3.34}$$

where $\oslash$ denotes component-wise division[1] and $\dot{\mathbf{q}}_{max} \in \mathbb{R}^j_{>0}$ is the vector of joint velocity limits.

[1]This operator returns one in the case of a division by zero.

3.4.3 Acceleration limitation

To prevent the robot from performing abrupt motions, its acceleration[1] can be constrained. To do so, a limitation similar to (3.33) can be written, by taking into account the previous control point velocity and the maximum velocity increase that can occur in one time step:

$$\mathcal{C}_{x,acc} = \begin{cases} \frac{\|\mathbf{v}(t-T_s)\| + A_{max}\ T_s}{\|\mathbf{v}_{tot}(t)\|} & \text{if } \|\mathbf{v}_{tot}\| > 0 \\ 1 & \text{otherwise,} \end{cases} \tag{3.35}$$

with $A_{max} \in \mathbb{R}_{>0}$ the maximum acceleration allowed. This limitation can also be described in the joint space using a similar approach:

$$\mathcal{C}_{q,acc} = \min\left([|\dot{\mathbf{q}}(t-T_s)| + \ddot{\mathbf{q}}_{max}\ T_s] \oslash |\dot{\mathbf{q}}_{tot}(t)|\right), \tag{3.36}$$

with $\ddot{\mathbf{q}}_{max} \in \mathbb{R}^j_{>0}$ the vector of joint acceleration limits. $\mathcal{C}_{x,acc}$ and $\mathcal{C}_{q,acc}$ can then be included in $\mathcal{C}$ if needed.

3.4.4 Power limitation

Another safety criteria in the ISO10218-2011 standard is power limitation. Power can be limited at a low level, e.g. electric power as with the Kuka LWR4+, or at control level, as we do here. One advantage of providing the limitation at the control level is that it can easily be tuned online or even deactivated to allow high dynamic motions when no operator is present. To this end, we propose a new constraint that can be added to the controller to limit the amount of exchanged power. Let us first consider the definition of power:

$$P = \left\langle {}^T\mathbf{f}_{ext}, {}^T\dot{\mathbf{x}}_{tot} \right\rangle. \tag{3.37}$$

Equation (3.37) can also be expressed at the joint level using:

$$P = \left\langle \boldsymbol{\tau}_{ext}, \dot{\mathbf{q}}_{tot} \right\rangle. \tag{3.38}$$

The choice between the two expressions depends on the sensors available on the robot. Then, using either (3.37) or (3.38), we can define the associated power constraint:

$$\mathcal{C}_{pow} = \begin{cases} \frac{P_{max}}{|P|} & \text{if } P < 0 \\ 1 & \text{otherwise,} \end{cases} \tag{3.39}$$

$P_{max} \in \mathbb{R}_{>0}$ being the maximum exchanged power allowed. It can be noticed that the limitation can be effective only when the power is negative, i.e. when energy is absorbed by the human. This way, the velocity will only be reduced when the robot represents a potential threat to the operator. This is illustrated in Fig. 3.3.

[1] We only consider positive acceleration: limiting the deceleration is not possible in our framework since we decide to only reduce the robot velocity. This choice is also motivated by the fact that limiting the deceleration could break other constraints.

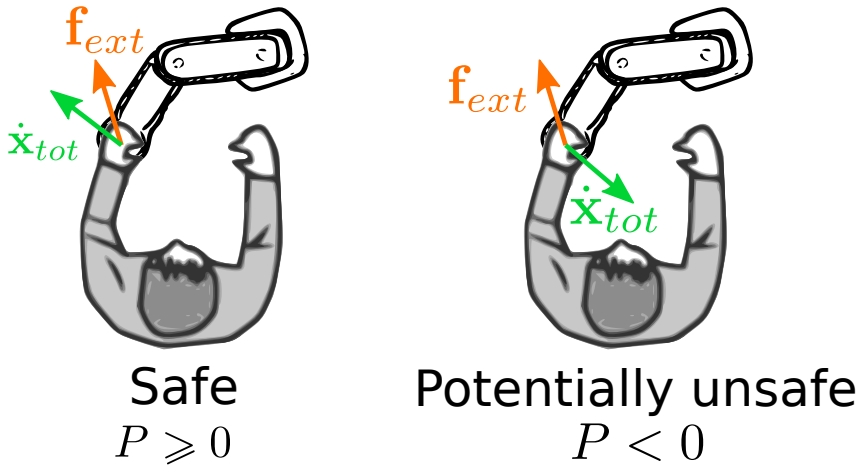


Figure 3.3 – Safe and unsafe power values.

3.4.5 Force limitation

Force limitation is the third and last constraint of the ISO10218-2011. True force limitation is a really challenging problem, since it requires a complete knowledge of the environment, including present humans. While for the environment a complete map (position, materials, etc) can be obtained, it is nearly impossible to have the same knowledge about humans, as this would require estimation of their motion and body impedance parameters which change over time (e.g. fatigue or muscular co-contraction may stiffen a joint) and from a person to another. Hence, we decided to adopt a reactive approach that does not rely on an environment model. By doing so, even if the external force can pass the limit, the robot will react and quickly move away from the impact, to reach a safe state. This approach has two steps. The first step consists in generating a very high velocity in the direction opposite to the external force to move away from the collision and to mitigate the other inputs. The second step consists in adding one or more constraints to limit the actual velocity ${}^{T}\dot{\mathbf{x}}$ generated by the first step, to a safe value.

Let us first describe the required velocity input to be included in $\mathcal{V}$:

$$ {}^{T}\dot{\mathbf{x}}_{F_{lim}}(t) = \begin{cases} -\beta \frac{\mathbf{f}_{ext}}{\|\mathbf{f}_{ext}\|} & \text{if } \|\mathbf{f}_{ext}\| \geq F_{max} \\ 0 & \text{if } \|\mathbf{f}_{ext}\| = 0 \\ {}^{T}\dot{\mathbf{x}}_{F_{lim}}(t - T_s) & \text{otherwise.} \end{cases} \tag{3.40} $$

In this equation, β is a very high gain (e.g. 10^{12} $m.s^{-1}$) and $F_{max} \in \mathbb{R}_{>0}$ is the force limit. In can be seen from (3.40) that a very high velocity is generated as soon as the external force passes the limit and maintained until the contact with the operator or environment disappear.

Then, in order for the robot to behave safely while executing ${}^{T}\dot{\mathbf{x}}_{F_{lim}}$, its velocity must be limited. This is done by including one or more velocity constraints in set $\mathcal{C}$. For example, to respect the ISO10218-2011, velocity and power limitations must be included:

$$ \mathcal{C}_{force} = \min(\mathcal{C}_{x,vel}, \mathcal{C}_{pow}). \tag{3.41} $$

$\mathcal{C}_{force}$ can then be included in $\mathcal{C}$ to provide the safety requirements.

3.4.6 Kinetic energy limitation

In the case of a robot colliding with a human operator, kinetic energy is closely related to the level of injury endured by the operator. Haddadin et al. have proposed a methodology to find a mapping between the kinetic energy, the impactor shape and the induced level of injury [22]. Kinetic energy is a major concern when it comes to safety, and as such should be limited. The kinetic energy of a rigid body in translation only is defined as:

$$E_k = \frac{1}{2} m \|\mathbf{v}\|^2 \tag{3.42}$$

where m is the mass of the body. Limiting the kinetic energy can be seen as a form of velocity limitation. As such, the following constraint can be derived from (3.33):

$$\mathcal{C}_{E_k} = \begin{cases} \frac{\sqrt{\frac{2E_{k\,max}}{m}}}{\|\mathbf{v}_{tot}\|} & \text{if } \|\mathbf{v}_{tot}\| > 0 \\ 1 & \text{otherwise,} \end{cases} \tag{3.43}$$

It can be seen that (3.43) is equivalent to (3.33) in the case $V_{max} = \sqrt{\frac{2E_{k\,max}}{m}}$. Plugging (3.43) into (3.18) yields to a total kinetic energy limited to E_{kmax}.

The case of manipulators

When controlling a manipulator, (3.42) cannot be applied to compute the robot kinetic energy since it cannot be represented as a point mass, but an alternative solution can be found thanks to the concept of equivalent mass m_{eq} presented below. Let us first recall the joint space dynamics of a rigid robot, previously detailed in 2.1:

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \boldsymbol{\tau}_f(\dot{\mathbf{q}}). \tag{3.44}$$

As it has been shown in [55], it is possible to define an operational space kinetic energy matrix:

$$\boldsymbol{\Lambda}(\mathbf{q}) = (\mathbf{J}_p(\mathbf{q})\mathbf{M}(\mathbf{q})^{-1}\mathbf{J}_p^\top(\mathbf{q}))^{-1}, \tag{3.45}$$

with $\mathbf{J}_p(\mathbf{q})$ the Jacobian matrix that relates operational space velocities at the expected collision point[1] p to joint velocities. This kinetic energy matrix can be decomposed [22] into:

$$\boldsymbol{\Lambda}^{-1}(\mathbf{q}) = \begin{bmatrix} \boldsymbol{\Lambda}_v^{-1}(\mathbf{q}) & \bar{\boldsymbol{\Lambda}}_{v\omega}(\mathbf{q}) \\ \bar{\boldsymbol{\Lambda}}_{v\omega}^\top(\mathbf{q}) & \boldsymbol{\Lambda}_\omega^{-1}(\mathbf{q}). \end{bmatrix} \tag{3.46}$$

The equivalent mass perceived at the impact point along the direction of the unit vector $\mathbf{u}$ pointing toward the closest operator is given in [22]:

$$m_{eq} = (\mathbf{u}^\top \boldsymbol{\Lambda}_v^{-1}(\mathbf{q})\mathbf{u})^{-1}. \tag{3.47}$$

The equivalent mass computed with (3.47) can then be used in (3.43) to limit the kinetic energy of the manipulator.

[1] This can be chosen as the robot point closest to the operator or as the robot end effector.

3.4.7 Separation distance

If the separation distance between the robot and near operators is monitored, then it can be used to adapt the limits imposed to the robot. Indeed, only a low level of security may be required if no one is present in the robot surrounding, whereas very strict limitations may be imposed when working closely or in collaboration with humans. A simple example is depicted in Fig. 3.4. To accommodate with this, we use the interpolation function

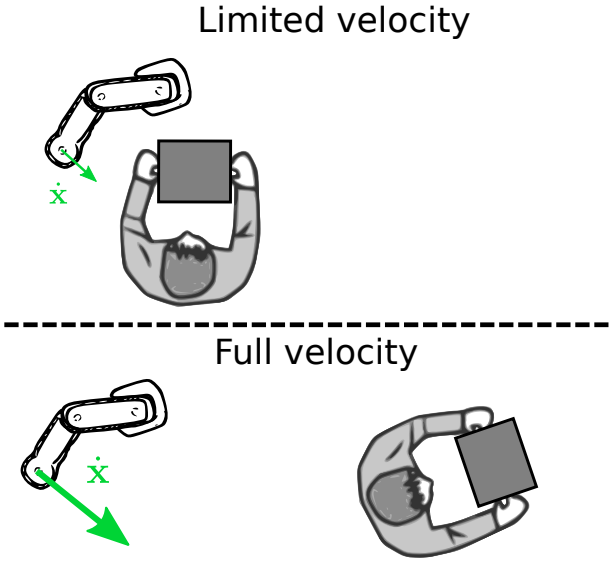


Figure 3.4 – Velocity limitation depending on separation distance.

described in Sect. A.2 to allow a smooth adaptation of the limits depending on the distance d_{min} to the closest operator or to any other object to be avoided. As an example, we can use (A.15) to smoothly interpolate the velocity limit in (3.33) so that: the robot will stop if an operator is at less than 50 cm, and moves at up to 0.25 m.s^{-1} if the workspace if free within a 2 m radius:

$$V_{max} = f_{int}(d_{min}, 0.5, 2, 0, 0.25). \tag{3.48}$$

3.5 Software implementation

The controller, constraints and various inputs described in this chapter are available through the OpenPHRI library, distributed online[1] free of charge under the GNU LGPL license[2]. The library is written in C++ to maximize the efficiency in terms of computations and memory footprint and also to be easily embedded in already existing projects. Python bindings are also provided, since this language is largely used in the robotics community and it allows quick prototyping, while keeping low computational times since most computations are performed in machine language. An interface for the robotics simulator V-REP[3] is provided and interfaces to other simulators and robots can be easily added.

[1] `https://github.com/BenjaminNavarro/OpenPHRI`
[2] `https://www.gnu.org/licenses/lgpl-3.0.en.html`
[3] `http://www.coppeliarobotics.com`

3.5.1 Project organization

Here is the detailed hierarchy of the project:

Table 3.1 – Detailed project hierarchy.

Hierarchy	Content
src	Source files for the libraries
• OpenPHRI	C++ implementation of the controller and of the robot data structure
- constraints	Constraints implementation
- force_generators	Task space force inputs
- velocity_generators	Task space velocity inputs
- torque_generators	Joint space force inputs
- joint_velocity_generators	Joint space velocity inputs
- utilities	Various utilities such as clock, data logger, intergrator/derivator
• pyOpenPHRI	Python bindings for OpenPHRI, developed with Boost.Python[a]
• vrep_remote_api	API[b] for external V-REP control
• vrep_driver	OpenPHRI to V-REP interface
include	Header files for the libraries, that follow the same structure as src
tests	Unit tests for various parts of the OpenPHRI library
apps	Example and demonstrations, to help getting started with OpenPHRI
share	Robot models and scenes for V-REP
build	Build directory

3.5.2 Example

Listing 3.1 presents a short but meaningful example of OpenPHRI usage. In less than 25 lines of code (comments excluded) one can set up a V-REP scenario where a serial manipulator robot Kuka LWR4+ is moved with an external force applied, while limiting its velocity, reading sensory input, and sending joint commands to the simulator. It can be seen (at lines 10 and 18) that smart pointers[4] are used instead of raw pointers to pass data through the library. This has the advantage of releasing automatically the associated memory when it is no longer referenced in the program, avoiding memory leaks. Also, using pointers instead of values allows the user to change some parameters (e.g. maximum velocity) online very easily.

Since the example is self-explanatory thanks to the comments, we only highlight few key elements of the library. First, a *Robot* object is required. This is a data structure containing all the information regarding its current state (e.g., joint positions, external force, kinematics) and control parameters (e.g., velocity bounds, damping factor). Next, the controller itself, called *SafetyController*, is created and paired to the robot to control.

[4]Shared pointers from the standard C++ library.

A generic *add* method can be used to add constraints, velocity and force inputs to the controller. The name given as first parameter to the *add* method can be used to retrieve or remove the associated constraint or input from the controller. Then, to run the controller, the call operator (line 36) is used.

```
1  #include <OpenPHRI/OpenPHRI.h>
2  #include <vrep_driver/vrep_driver.h>
3
4  // Use namespaces to shorten the types
5  using namespace phri;
6  using namespace std;
7
8  int main(int argc, char* argv[]) {
9    // Create a robot with a name (used by the V-REP driver) and a joint
       count
10   auto robot = make_shared<Robot>("LBR4p", 7);
11   // Set task space damping values to 100
12   *robot->controlPointDampingMatrix() *= 100.;
13
14   // Create a controller for the robot
15   auto safety_controller = SafetyController(robot);
16
17   // Create a pointer to store the maximum velocity, here 0.1m/s
18   auto max_vel = make_shared<double>(0.1);
19   // Add this to the controller
20   safety_controller.add("velocity constraint", VelocityConstraint(
       max_vel));
21
22   // Feed the external force to the controller
23   safety_controller.add("external force", ExternalForce(robot));
24
25   // Create a V-REP driver for sending joint positions with 5ms sample
       time
26   vrep::VREPDriver driver(robot, ControlLevel::Joint, 0.005);
27   // Use V-REP synchronous mode.
28   driver.enableSynchronous(true);
29   // Start the simualtion
30   driver.startSimulation();
31
32   while(1) {
33     // Update the robot with the current simulation data
34     driver.getSimulationData();
35     // Run the controller
36     safety_controller();
37     // Send the control output
38     driver.sendSimulationData();
39     // Trigger a simulation step
40     driver.nextStep();
41   }
42 }
```

Listing 3.1 – Example of a short OpenPHRI application.

3.5.3 Benchmarks

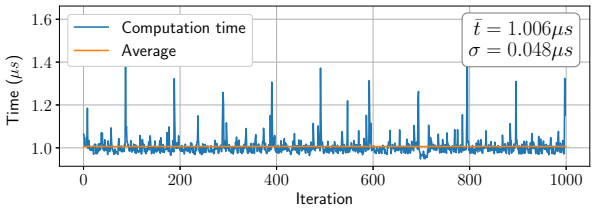
In physical human-robot interaction, in order for the robot to react quickly in the case of an impact or to be as transparent as possible when physically collaborating with a human, its control loop should run at a minimum of 1kHz. It is crucial that the controller presented here and its implementation in OpenPHRI are fast enough to comply with this timing constraint. To assess the performance of this library, some benchmarks have been run on a computer equipped with an Intel i7-6700HQ @ 2.6GHz running Linux 4.11 and the results are present below.

In Fig. 3.5, we present the results of the benchmarks for the controller associated with different constraints and force or velocity inputs running on a 7 degrees of freedom manipulator. At each iteration, the controller is run 10000 times and the average computation time is retained. On Fig. 3.5a-3.5e, the average computation time $\bar{t}$ and the standard deviation σ over 1000 iterations are indicated. It should be noted that the computation of the forward and inverse kinematics is not included in these results to only focus on the control computation time overhead. Also, the current controller implementation is single-threaded but, given the very low computational times observed in the benchmarks ($\bar{t} < 4\mu s$ in the most complex scenario presented, i.e., 3.5e), a multi-threaded version does not seem to be required (although it would be possible to develop one). From Fig. 3.5f, it can be seen that the memory usage[1] stays very low, with a peak at 186 KiB. The abscissa of this graph represents snapshots taken regularly during execution.

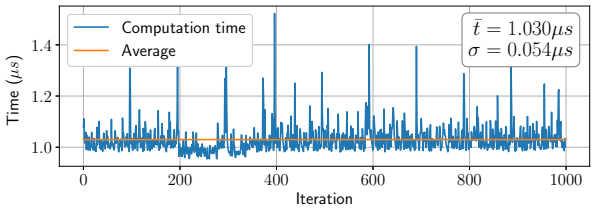
3.5.4 Sum up

Regarding the benchmark results, the presented controller can be executed at very high rates (>1kHz) or on low end machines, while still achieving good performances. Moreover, the OpenPHRI library is easy to use, and programs can be written in a very concise way, while retaining high readability. The open source nature of the project allows its users to add new features and to make improvements, avoiding it to become outdated on a long term.

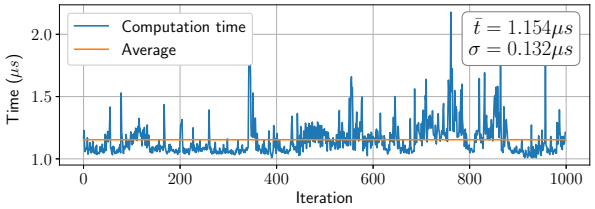
[1]Measured using the Massif tool from the Valgrind profiling software.



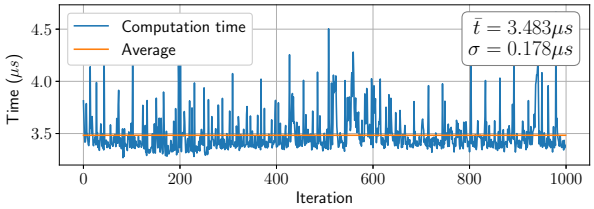
(a) Controller with no constraints and no inputs. ($\mathcal{C} = \mathcal{V} = \mathcal{F} = \omega = \Gamma = \{\}$)



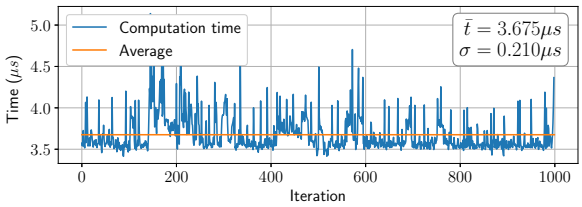
(b) Controller with a velocity constraint ($\mathcal{C} = \{\mathcal{C}_{x,vel}\}$, $\mathcal{V} = \mathcal{F} = \omega = \Gamma = \{\}$).



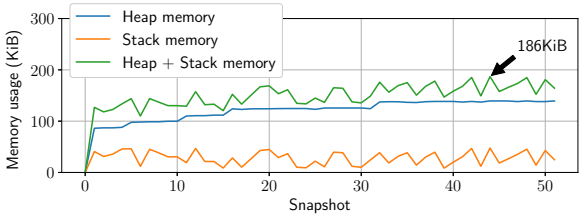
(c) Controller with velocity and power constraints ($\mathcal{C} = \{\mathcal{C}_{x,vel}, \mathcal{C}_{pow}\}$, $\mathcal{V} = \mathcal{F} = \omega = \Gamma = \{\}$).



(d) Controller with velocity, power and kinetic energy constraints ($\mathcal{C} = \{\mathcal{C}_{x,vel}, \mathcal{C}_{pow}, \mathcal{C}_{E_k}\}$, $\mathcal{V} = \mathcal{F} = \omega = \Gamma = \{\}$).



(e) Controller with velocity, power and kinetic energy constraints and with a potential field, a virtual stiffness and a force controller in the task space ($\mathcal{C} = \{\mathcal{C}_{x,vel}, \mathcal{C}_{pow}, \mathcal{C}_{E_k}\}$, $\mathcal{V} = \{\dot{\mathbf{x}}_{fc}\}$, $\mathcal{F} = \{\mathbf{f}_{x,stiff}, \mathbf{f}_{rep}\}$, $\omega = \Gamma = \{\}$).



(f) Memory usage while executing the controller benchmarks ((a)-(e)) sequentially.

Figure 3.5 – Benchmarks of the controller running on a 7 degrees of freedom manipulator.

3.6 Experiment

Let us now present the results of a full-featured experiment using the framework described in this chapter.

The experiment is split in two phases, a *teaching-by-demonstration phase*, and a *replay phase*, where the robot operates autonomously, in the presence of an obstacle and near the human operator. Figure 3.6 shows the setup, consisting in a Kuka LWR4+ arm, with external force $\mathbf{f}_{ext}$ estimated through the FRI interface[1]. All the code was written in C++ using the OpenPHRI library and integrated inside the Knowbotics Framework, currently under development at LIRMM, to interface with the hardware. The FRI library was used to communicate with the Kuka arm. The controller sample time was T=1 ms. To manage the robot behavior, we designed in OpenPHRI the finite state machine shown in Fig. 3.7.

It is important to note that our framework is used continuously throughout both the teaching and replay phases. An equivalent application using the V-REP simulator is

[1] `http://cs.stanford.edu/people/tkr/fri/html/`

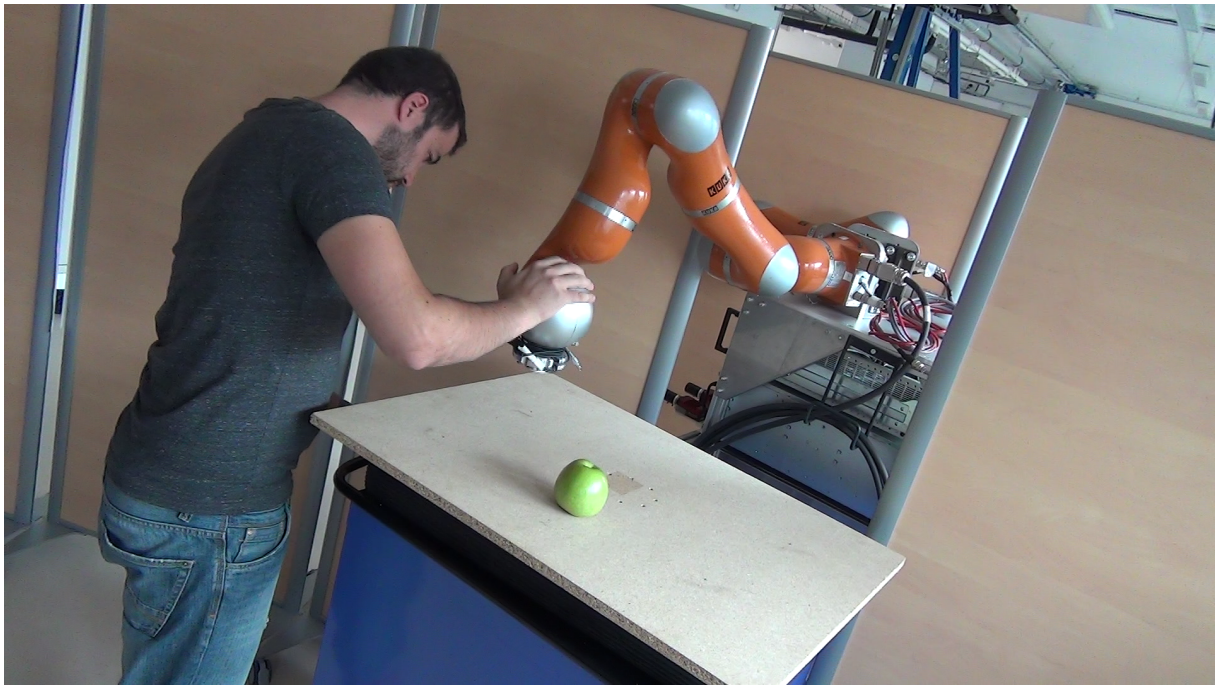


Figure 3.6 – Setup for the experiment.

available in the OpenPHRI repository under "apps/demo"[1]. The whole application has less than 600 lines of code: 125 for the main file and 440 for the finite state machine (header + source).

The *teaching phase* consists in manually guiding the robot, by applying $\mathbf{f}_{ext} \in \mathcal{F}$, to teach it the waypoints where it should later realize a force control task (apply $f_r = 30N$ for 2 s perpendicularly to the end-effector). The number of waypoints is not known a priori. A waypoint is recorded when no motion is detected for 3 s and the teaching phase ends if the robot remains still for 3 more seconds.

Once the operator has specified all the desired points, the *replay phase* is triggered. The trajectory generator is used to output the control point (end effector) reference velocity ($\dot{\mathbf{x}}_r \in \mathcal{V}$) for reaching each waypoint. When a waypoint is reached, the task space force controller is activated ($\dot{\mathbf{x}}_{fc} \in \mathcal{V}$). After the force has been correctly applied, the robot moves to the next waypoint. Once all the force control tasks have been performed, the robot returns to its original position using the trajectory generator. During the replay phase, while moving between waypoints, the external force at the control point is monitored to trigger an emergency stop if its norm exceeds 10 N, as explained in Sect. 3.4.1. Motion is resumed only when the external force is lower than 1 N. Additionally, potential fields ($\mathbf{f}_{rep} \in \mathcal{F}$) are used to avoid a known object (here, an apple) in the environment.

Throughout the experiment, the joint velocities sent to the robot are output by (3.19), with scaling factor α computed with the constraints in (3.18). The task space damping matrix is set to $\mathbf{B}_t = \text{diag}(250, \ldots, 250)$, while joint space damping $\mathbf{B}_j$ is not used. During force control tasks execution, an acceleration limit ($\mathcal{C}_{x,acc} \in \mathcal{C}$) of $A_{max} = 0.5$ m/s^2 is applied, to avoid abrupt motions. During the replay phase, a virtual stiffness $\mathbf{K}_t = \text{diag}(1000, \ldots, 1000)$, described in Sect. 3.2.2, is added to compensate deviations from the

[1] `https://github.com/BenjaminNavarro/OpenPHRI/tree/master/apps/demo`

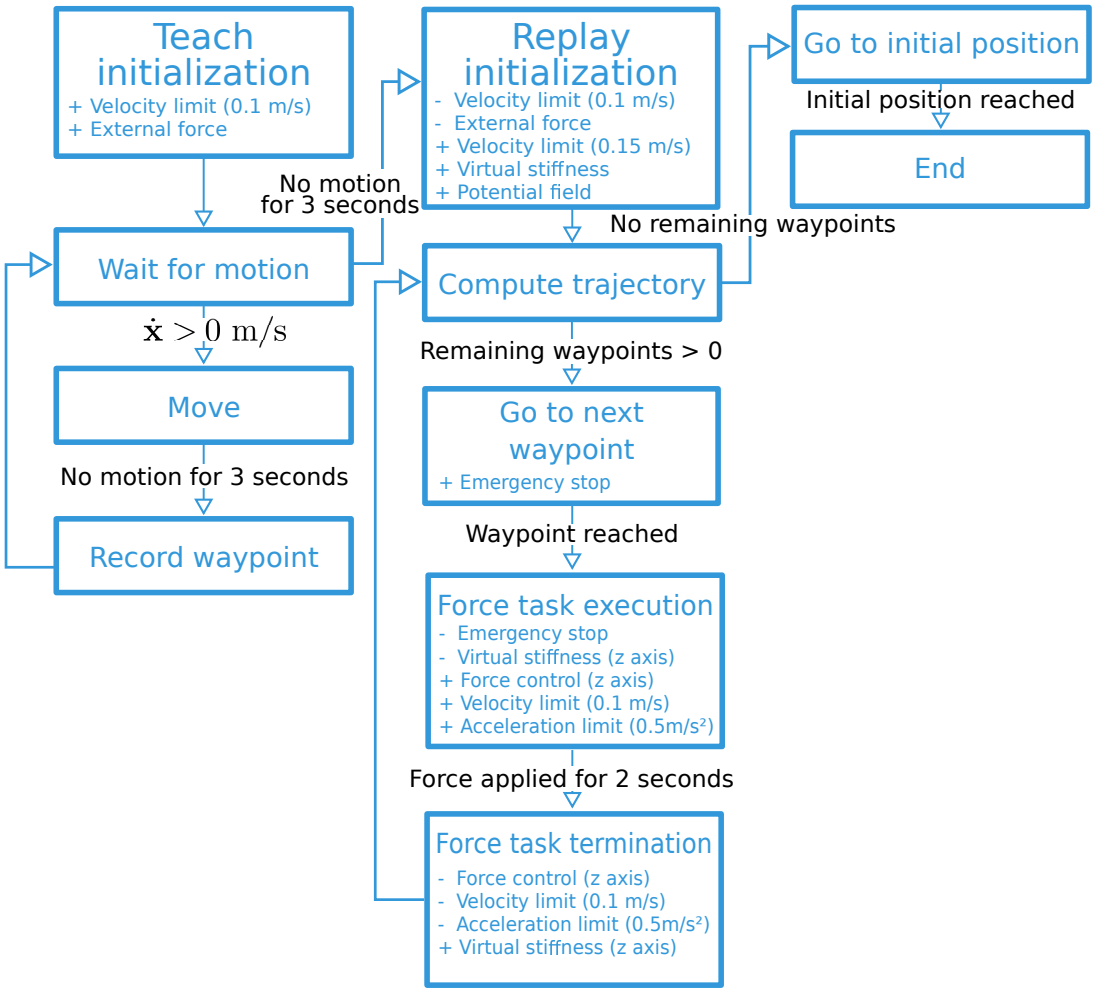


Figure 3.7 – Finite state machine used for the experiment. A + sign indicates an addition to the controller (new constraint or new input) while a - indicates a removal.

trajectory. Both $\mathbf{B}_t$ and $\mathbf{K}_t$ were tuned experimentally to provide a suitable behavior. The potential fields for obstacle avoidance are activated when the distance from the obstacle is below 0.2 m. Throughout the experiment the velocity is limited, to $V_{max} = 0.1$ m/s during teaching, and to $V_{max} = 0.15$ m/s during replaying.

Snapshots of the experiment are displayed in Fig. 3.8 while the results are shown in Fig. 3.9. A video of the experiment is joint to this manuscript[1]. The teaching phase takes place during the first 36 s. Then, the replay phase starts.

In Fig. 3.9a, the external force applied to the end-effector is displayed. The forces applied by the operator during the teaching phase (snapshots 3.8b-3.8c) can be observed from 2 to 30s. Then, the four force control tasks on the z axis (30N, snapshot 3.8e) as well as the collision on the y axis (70s, snapshot 3.8g) can be recognized.

The components of the total control point velocity $\dot{\mathbf{x}}_{tot}$ are given in Fig. 3.9b. During the first 30s, the relationship with the forces given in Fig. 3.9a is clear. Then, velocities are generated by the trajectory generator (snapshots 3.8d,3.8h) and the force control law.

The influence of the constraints can be observed in Fig. 3.9c-3.9d, where the scaling

[1] Also available at `https://youtu.be/Kt7u4p2Xz1g`

(a) The robot is waiting in its initial position.

(b) The operator teaches the first waypoint.

(c) The operator teaches the second waypoint.

(d) The robots goes to the first waypoint.

(e) Force control is performed at the first waypoint.

(f) The robot avoids the obstacle by using repulsive potential fields.

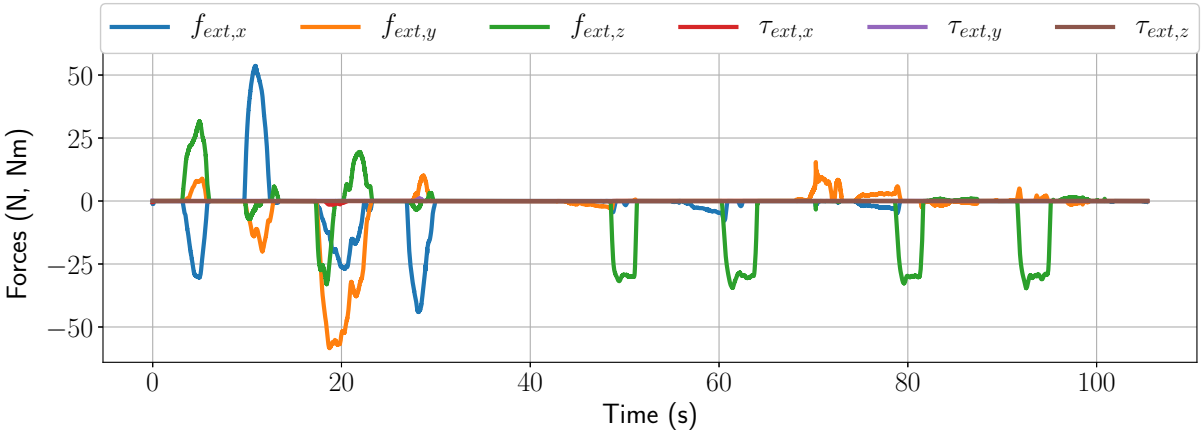
(g) The operator stops the robot to access the workspace.

(h) The robot returns to the initial pose.

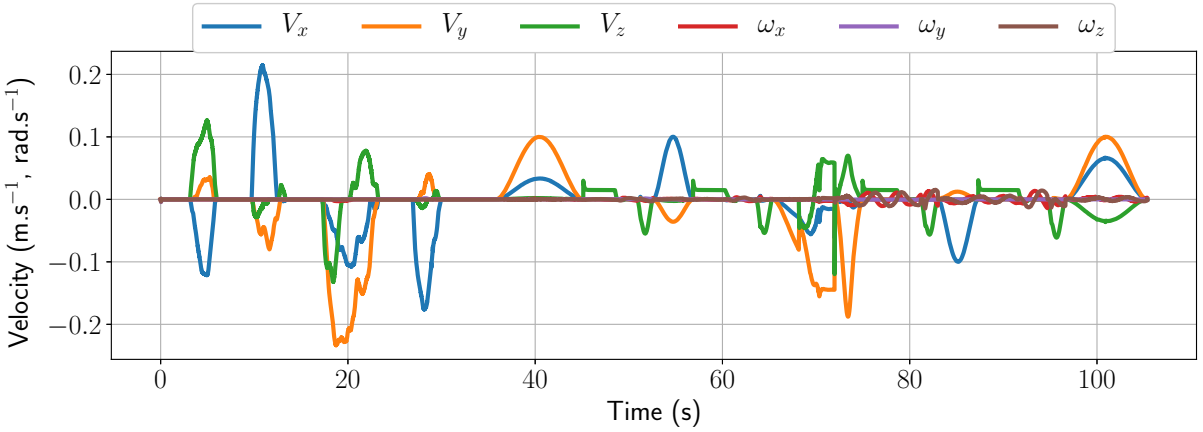
Figure 3.8 – Snapshots of the experiment: teaching (a-c) and replay (d-h) phases.

factor α is plotted on the first one and where the control point velocity after the velocity reduction is given in the second one. We can notice that the velocity has to be reduced several times during the experiment in order to comply with the constraints and maintain the operator's safety.

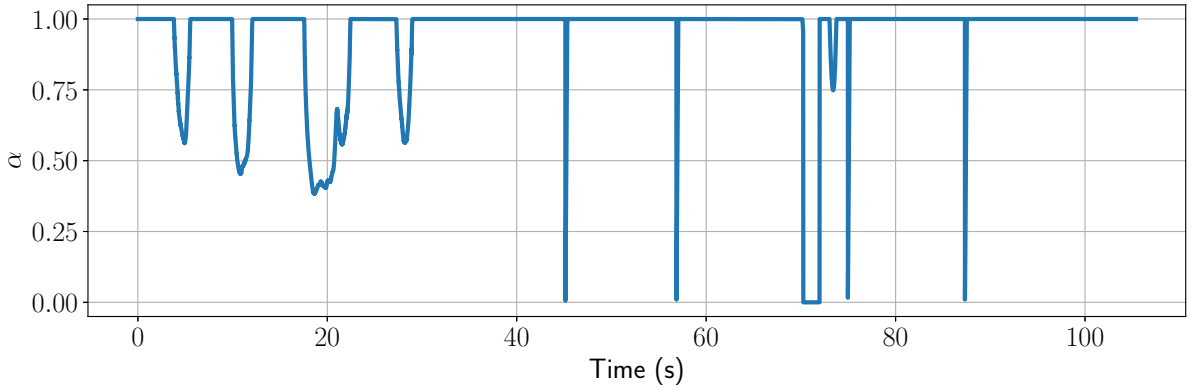
Finally, in Fig. 3.9e, the current velocity limitation and the norms of the control point translational velocities, before and after the velocity reduction, are given. We can see that during the teaching phase, the forces applied by the operator led to a velocity above the 0.1 m/s limit that was correctly reduced to meet the constraint. At the end of the teaching phase, the velocity limit is increased to 0.15 m/s since the operator should not be in the robot's workspace anymore. During the replay phase, the velocity is reduced while avoiding the obstacle at around t =70 s (snapshot 3.8f). We can also notice that during this avoidance motion, the velocity is brought to zero since the operator makes an unpredicted contact with the robot (snapshot 3.8g).



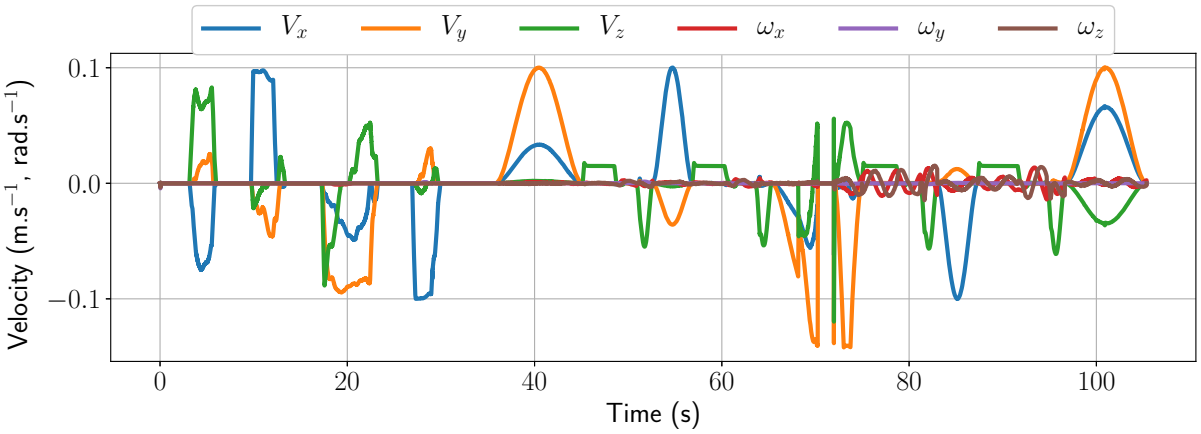
(a) Components of the external force $\mathbf{f}_{ext}$, applied by the human for teaching or upon collision (at $t = 70$ s), then by the robot during the four force control tasks.



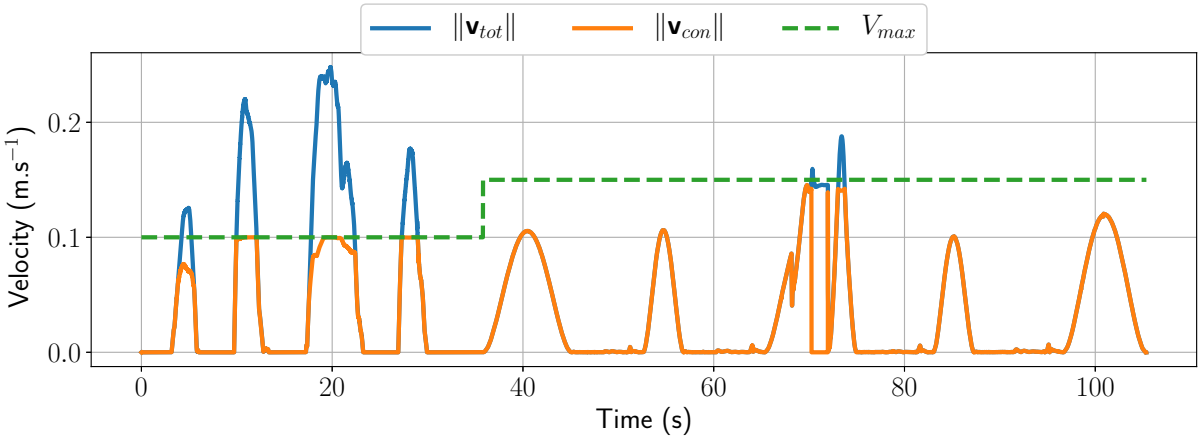
(b) Components of the control point total velocity $\dot{\mathbf{x}}_{tot}$.



(c) The scaling factor α, diminishing whenever the constraints are active.



(d) Components of the control point velocity applied after velocity reduction $\dot{\mathbf{x}}_{con} = \mathbf{J}\dot{\mathbf{q}}_{con}$.



(e) Comparison between the current velocity limit V_{max} and the total and applied translational velocity norms ($\|\mathbf{v}_{tot}\|$ and $\|\mathbf{v}_{con}\|$).

Figure 3.9 – Relevant variables during the experiment.

3.7 Conclusion

We proposed a new control framework to design safe human-robot interactive and collaborative applications. By adopting a two-layer approach, we can use, at the same time, constraints and inputs (velocity and force) expressed in both joint and task spaces, allowing to fit a wide range of scenarios. It has been made as easy to use as possible so that it can be quickly adopted and extended by other users. The associated open source library, OpenPHRI, aims to its adoption even more. The method as been proven effective in a mock up scenario with an operator guiding the robot to all the locations where a given force needs to be applied, then letting the robot perform the tasks autonomously. Velocity limitation has been applied during the whole task with the addition, during the autonomous phase, of collision avoidance and a protective stop in the case of unpredicted collisions. Benchmarks run on the controller showed that its performance is suitable for control loops of 1kHz or more, as often required for collaborative tasks.

In this framework, we considered a fixed manipulator but, of course, many different robot structures can be found and be useful in collaborative scenarios. In the next chapter, we will discuss the extension of this work to omnidirectional mobile manipulators and robotic hands.

Chapter 4

Extending to other robots

In the previous chapter, we presented a framework for the control of robotic manipulators physically interacting with humans. In this chapter, we will investigate how this collaborative framework can be extended to include different robot types, including mobile manipulators and robotic hands.

4.1 Mobile comanipulation framework

In this section, we consider a serial manipulator, as described in Chap. 3, mounted on an omnidirectional mobile platform with $\dot{\mathbf{x}}_{base} \in \mathbb{SE}(3)$ its cartesian control input. In this case, both $\mathbf{x}$ and $\dot{\mathbf{x}}$ are expressed in the robot base frame, attached to the center of the mobile base. Velocities at the end-effector will be generated using the two-layer safe damping control framework presented in Sect. 3.1. The work in this section focuses on the redundancy that arises from having both a manipulator and a mobile platform sharing some of the task space degrees of freedom and on how to provide a solution that emphasizes intuitiveness during physical collaboration.

4.1.1 End-effector control

As explained above, the end-effector is driven using the framework presented in Sect. 3.1. However, since the focus is made on physical collaborations, where the operator manually guides the robot, we configure (3.19) to have the external wrench being the only input. Also, for the sake of clarity, no safety constraints are imposed (although these can of course be added). This translates to:

$$\mathcal{V} = \{\} \tag{4.1}$$
$$\mathcal{F} = \{\mathbf{f}_{ext}\} \tag{4.2}$$
$$\omega = \{\} \tag{4.3}$$
$$\Gamma = \{\} \tag{4.4}$$
$$\mathcal{C} = \{\} \tag{4.5}$$

Then, the end-effector velocity is obtained using:

$$\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}_{con} \tag{4.6}$$

4.1.2 Whole body control strategy

Because of redundancy, mobile manipulators inherently share some mobility of the Cartesian space between the manipulator and the mobile base. In order to solve for such redundancy, we adopt the following strategy:

$$\dot{\mathbf{x}}_{arm} = \mathbf{A}\dot{\mathbf{x}}, \tag{4.7}$$
$$\dot{\mathbf{x}}_{base} = (\mathbf{I} - \mathbf{A})\dot{\mathbf{x}}, \tag{4.8}$$
$$\mathbf{A} = \text{diag}\{a_{v_x}, a_{v_y}, a_{v_z}, a_{\omega_x}, a_{\omega_y}, a_{\omega_z}\} \in \mathbb{R}^{6\times 6}. \tag{4.9}$$

In these equations, $\dot{\mathbf{x}}_{arm}$ is the velocity command for the arm and all six $a \in [0, 1]$. The derivation of the a values will be explained in section 4.1.3.

Inverse kinematics is used on the manipulator to map task space to joint space velocities:

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}(\mathbf{q})\dot{\mathbf{x}}_{arm}. \tag{4.10}$$

Since different kinematic structures exist to allow omnidirectional motion (Swedish or spherical wheels, legs, flying base, etc.) the actuation of the mobile base dof is not considered in this work and we assume that the mapping from $\dot{\mathbf{x}}_{base}$ to the base joint values exists and is known.

4.1.3 Constraints

In this section, we explain how the a values are calculated, in order to satisfy a set of constraints: distance to singularities, minimal manipulability, distance to objects and angular deviation. For each constraint, the goal is to move only the arm, while the system is far from that constraint. This is a major difference with classic whole body control, where all joints are actuated to perform a task. Our choice arises from the fact that for an operator manipulating the robot, it is more intuitive that the base is fixed when working locally, and moves only when a distant target needs to be reached. This behavior is more intuitive since it mimics the human one.

In order to get a smooth evolution of the a values, we use the interpolation f_{int} function described in appendix A.2 and illustrated in Fig. A.1.

4.1.4 Distance to singularities

Clearly, when the arm reaches a singular configuration, the mobile base has to take over. To this aim, we use λ (see (3.7)) as a measure of the distance to singular poses to derive the a values:

$$s = 1 - \lambda^2/\lambda_{max}^2 \tag{4.11}$$
$$a_{s,i} = f_{int}(s, 0, 1, 0, 1), \forall i \in \{v_x, v_y, ..., \omega_z\}. \tag{4.12}$$

Using (4.11) leads to s varying from 1 in non-sigular poses ($\lambda = 0$) to 0 at singularity ($\lambda = \lambda_{max}$).

4.1.5 Manipulability

The manipulability index defined by Yohikawa [64] is a largely used metric in mobile manipulation to solve the redundancy since it is related to the ability, for the arm, to produce velocities in the task space. It can be computed as:

$$\mu = \sqrt{det(\mathbf{J}\mathbf{J}^\top)} = \prod_{i=1}^{M} \sigma_i \tag{4.13}$$

where σ_i is the i-th singular value of $\mathbf{J}$. This measure can be weighted with a factor decreasing near joint limits, as in [65]. We propose the following penalization cost:

$$\beta(\mathbf{q}) = \prod_{i=1}^{j} \left[1 - \left[\frac{2q_i - (q_i^+ + q_i^-)}{q_i^+ - q_i^-} \right]^2 \right] \in [0, 1]\,, \tag{4.14}$$

where q_i^+ and q_i^- are the upper and lower limits of the i-th joint. We can then merge the two measures using:

$$m = [\beta(\mathbf{q})\varphi + (1 - \varphi)]\,\mu, \tag{4.15}$$

with φ a scalar value that can be adjusted from 0 (no penalization) to 1 (full penalization) depending on the desired effect of β on the manipulability.

To account for manipulability, we set the a values to:

$$a_{m,i} = f_{int}(m, m_{min}, m_{th}, 0, 1), \forall i \in \{v_x, v_y, ..., \omega_z\}, \tag{4.16}$$

m_{min} being the smallest manipulability allowed and m_{th} the manipulability threshold at which velocities start to be transfered to the mobile base.

4.1.6 Distance to objects

The manipulator workspace can be limited by real (e.g. the mobile base body, to avoid self-collisions) or virtual (e.g. a virtual wall limiting the arm motion) physical constraints. We consider them all as geometric objects (e.g., planes, spheres, etc). We attach to the end effector a virtual sphere, with a radius large enough to contain any tool the robot may be carrying. Then, we define the set of n physical constraints (objects) limiting the arm workspace. Finally, we compute the distance between each object and the sphere, using the GJK algorithm [66]. This algorithm outputs the pair of closest points, p_{obj} and p_{sphere}, that respectively belong to the surface of the object and to that of the sphere. To compute the a values, we first evaluate for each object k a distance vector $d^k = [d_x^k\ d_y^k\ d_z^k]^\top$ using algorithm 1.

In this algorithm, d_{min} and d_{th} represent the minimum and threshold distances used by the interpolator. We also impose $d_{min} > 0$ so that $|\Delta p_i| = 0$ is true only when the i-th axis is unconstrained. Finally, we can compute the a values realizing the distance constraint with:

$$a_{d,v_i} = \prod_{k=1}^{n} d_i^k, \forall i \in \{x, y, z\}. \tag{4.17}$$

for $k \leftarrow 1$ **to** n **do**
  $\Delta p^k = p^k_{obj} - p_{sphere}$
  for $i \in \{x, y, z\}$ **do**
    if $|\Delta p^k_i| > 0$ **then**
      $d^k_i = f_{int}(|\Delta p^k_i|, d_{min}, d_{th}, 0, 1)$
    else
      $d^k_i = 1$
    end
  end
end

Algorithm 1: Workspace distance computation.

4.1.7 Angular deviation

In order to let the operator rotate the mobile base when needed, we constrain the angular deviation to the reference orientation $\boldsymbol{\theta}^*$. To do so, we define:

$$a_{d,\omega_i} = f_{int}(\Delta\theta_i, \Delta\theta_{th}, \Delta\theta_{max}, 1, 0), \forall i \in \{x, y, z\} \tag{4.18}$$

where $\Delta\theta_i = |\theta^*_i - \theta_i|$, $\Delta\theta_{th}$ is the angular activation threshold and $\Delta\theta_{max}$ is the maximum angular error. In (4.18), the a values vary from 1 at the angular threshold to 0 at the maximum deviation.

4.1.8 Constraint deactivation

Since all the constraint values depend solely on the current robot state, the arm can be locked in one or more task space directions if the corresponding a values approach zero (e.g., if the operator stretches it to the singular configuration, s/he cannot move it afterwards). To solve this problem, we propose a general deactivation strategy that allows the manipulator to move again if the generated velocity $\dot{\mathbf{x}}$ tends to move the robot away from the constraint. For this, we define a virtual manipulator with joint values $\mathbf{q}^v \in \mathbb{R}^j$, end effector pose $\mathbf{x}^v \in \mathbb{SE}(3)$ and associated Jacobian $\mathbf{J}^v$. Then, we execute the end-effector velocity $\dot{\mathbf{x}}$ on the virtual arm over a sampling period T_s, starting from the real robot's current configuration. This gives us two manipulator configurations to compare. At each sample time, we update the virtual robot with:

$$\mathbf{q}^v = T_s \mathbf{J}^\dagger(\mathbf{q})\dot{\mathbf{x}} + \mathbf{q}, \tag{4.19}$$

$$\mathbf{x}^v_{arm} = f_x(\mathbf{q}^v), \tag{4.20}$$

$$\mathbf{J}^v = f_J(\mathbf{q}^v). \tag{4.21}$$

In these equations, f_x and f_J are the forward kinematics algorithms for extracting the manipulator's pose and Jacobian, respectively. Then, constraints (4.12), (4.16), (4.17) and (4.18) are computed for both the real and virtual arm. For each pair of constraints, if the virtual arm constraint value is greater than the real arm one, the deactivation mechanism is triggered. This results in:

$$a = \begin{cases} a^r & \text{if } a^v < a^r \text{ or } t > t_{end}, \\ f_{int}(t, t_{start}, t_{end}, a_{i,start}, 1) & \text{otherwise,} \end{cases} \tag{4.22}$$

with a^r and a^v the constraint values for the real and virtual arm respectively, t the current time, t_{start} the time at which the mechanism was triggered and its associated value $a_{i,start}$ and t_{end} (initialized to 0) the time at which the a value will reach 1. With this technique, we ensure a smooth transfer of the velocities from the mobile base to the arm.

4.1.9 Merging the constraints

In order to use different constraints at the same time, we propose to multiply them to derive the value to inject in (4.9). This translates to:

$$a = \prod_{i \in C} a_i, \tag{4.23}$$

with C being the set of constraints to include.

Instead of the product, the minimal value could also have been used. However, this would not allow the operator to feel that a new constraint is approaching, and react to it if the effect is not the desired one.

4.1.10 Experiments

In this section, we present the experiments assessing the correct behavior of the proposed framework. In Sect. 4.1.11, we present simulations for a given reference trajectory $\dot{\mathbf{x}}^*$ with each constraint taken separately; then, in Sect. 4.1.12, we introduce the setup for real robot validation, and we comment the results in Sect. 4.1.13.

4.1.11 Validation

Distance to singularity

Figure 4.1 presents simulation results when only the singularity constraint is activated. The velocity command $\dot{\mathbf{x}}$ extends the arm to reach a singular configuration, where the mobile base starts moving, then retracts it to a non-singular pose. The relevant parameters are the following: $\epsilon = 0.1$, $\lambda_{max} = 0.1$ and $C = \{s\}$. At $t = 6.65s$, the singularity constraint is activated and velocities are progressively transferred from the arm to the mobile base, until $t = 9.75s$, when the mechanism is deactivated. From this instant, the a_s values increase up to 1 where the mobile base is stopped and only the arm moves.

Manipulability

Results for the manipulability constraint test are displayed in Fig. 4.2. The same velocity command $\dot{\mathbf{x}}$ as in 4.1.11 is used. The parameters for this test are $m_{th} = 0.06$, $m_{min} = 0.03$, $\varphi = 0.2$ and $C = \{m\}$. As expected, the arm follows the velocity trajectory until the manipulability measure drops below the threshold m_{th} (at t=5.8s) and stops when m_{min} is reached (in this example, the manipulability stays just above 0.03 so the arm is not at a complete rest, but almost disabled). At this point, the mobile base fully tracks $\dot{\mathbf{x}}$ and the arm is at rest. At $t = 10s$, the arm starts moving fatser again.

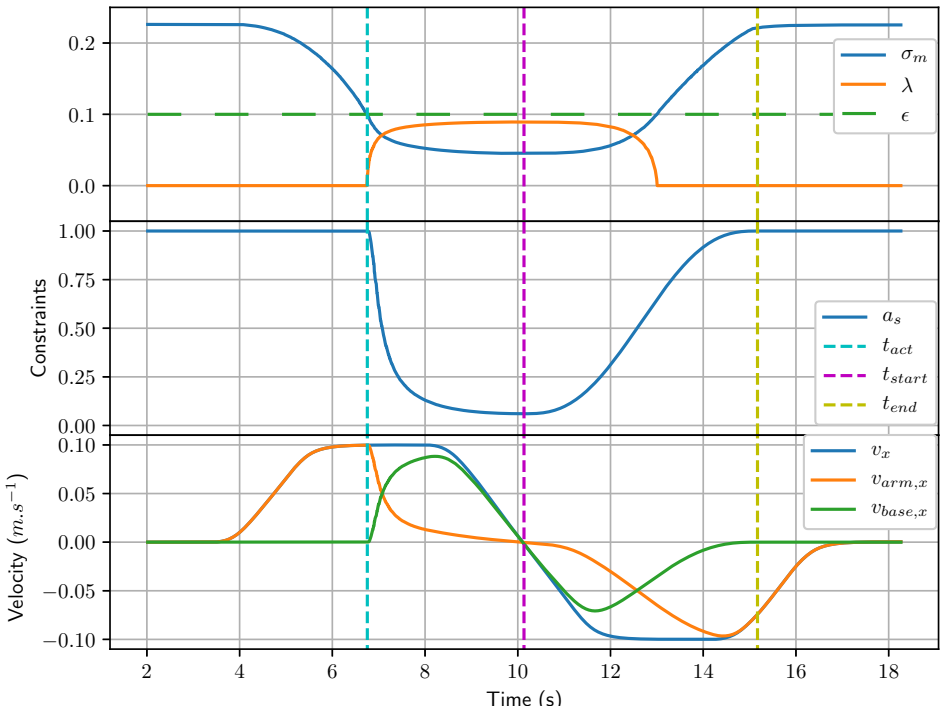


Figure 4.1 – Distance to singularity simulation. Top: smallest singular value σ_m and damping factor λ, middle: singularity constraints $a_s = a_{s,v_x} = a_{s,v_y} = a_{s,\omega_z}$, bottom: velocity commands along the x axis. t_{act} is the time at which the constraint gets activated.

Distance to objects

To assess the behavior of the workspace constraint, we use only one object ($n = 1$), a $1 \times 1 \times 2$ m box, centered at the base frame origin. The same velocity profile as in Sect. 4.1.11 is used but with opposite sign to first send the TCP against the mobile base and then move away from it. The TCP virtual sphere radius is set to 15cm and we use $d_{th} = 0.05$, $d_{min} = 0.001$ and $C = \{d\}$. Results are presented in Fig. 4.3. As in the previous experiments, the arm tracks the velocity profile until the threshold distance is reached. Then, velocities are progressively transferred to the mobile base. It is only when $\dot{\mathbf{x}}$ becomes positive (to send the TCP away from the mobile base) that the arm starts moving again and the base starts to decelerate and finally stop.

Angular deviation

For the angular deviation test, we use a reference rotational velocity ω_z^* that rotates the TCP above the maximum allowed orientation error $\Delta\theta = 1$ rad. We use $C = \{a\}$. It can be seen from Fig. 4.4 that the constraint gets activated when $\Delta\theta_z$ crosses $\Delta\theta_{th} = 0.5$ rad and that the velocities are correctly transfered from the arm to the mobile base. As in the previous examples, deactivation occurs when the velocity sign changes.

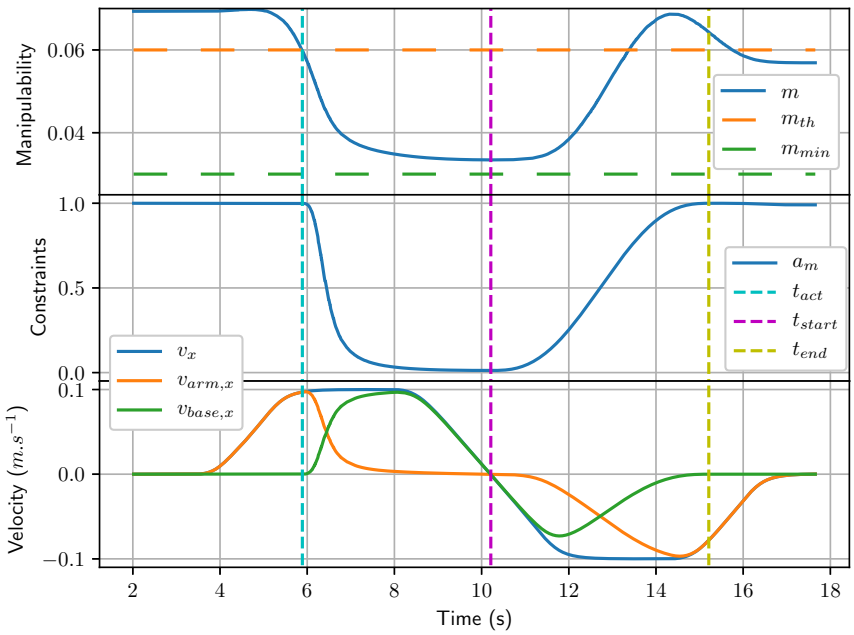


Figure 4.2 – Manipulability simulation. Top: manipulability measure m, middle: manipulability constraints $a_m = a_{m,v_x} = a_{m,v_y} = a_{m,\omega_z}$, bottom: Velocity commands along the x axis. t_{act} is the time at which the constraint gets activated.

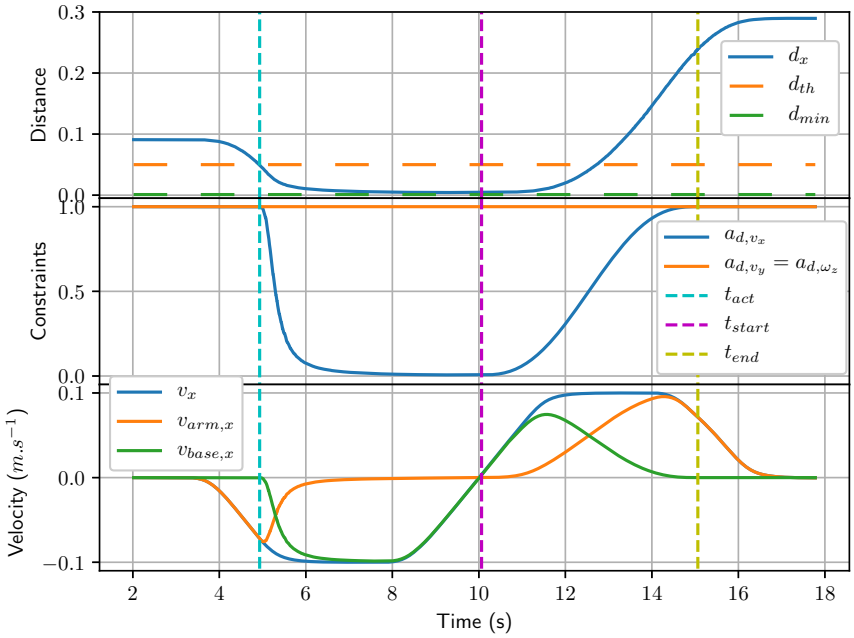


Figure 4.3 – Distance to objects simulation. Top: Minimal distance d_x, middle: workspace constraints, bottom: Velocity commands along the x axis; t_{act} is the time at which the constraint gets activated.

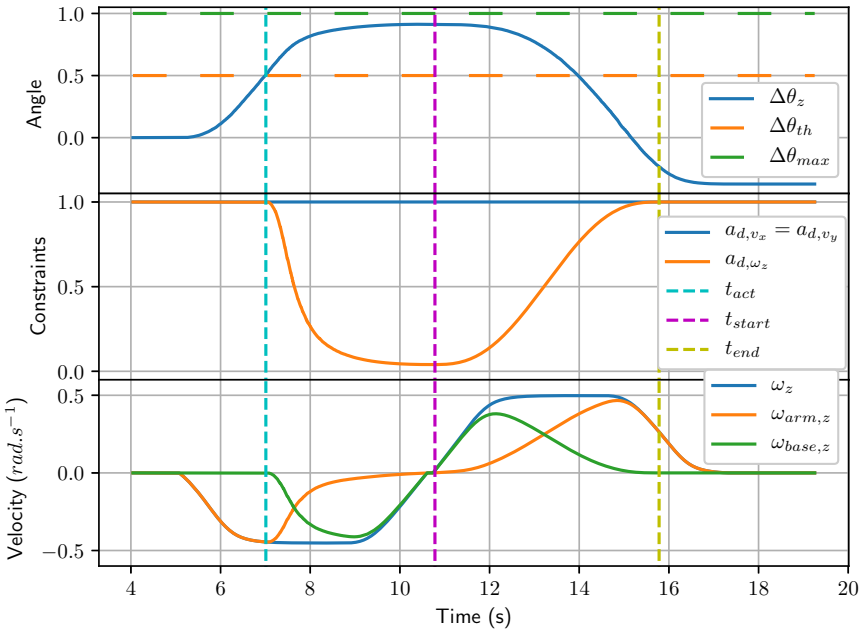


Figure 4.4 – Angular deviation simulation. Top: Angular error $\Delta\theta_z$, middle: workspace constraints, bottom: Velocity commands (z axis); t_{act} is the time at which the constraint gets activated.

4.1.12 Real experimental setup

To validate the proposed approach, we set up an application where an operator needs to move the TCP, attached to the arm's end effector, to a distant area, outside the manipulator's workspace. To this end, we used the LIRMM Bazar[1] platform, shown in Fig. 4.5. Bazar is composed of a Neobotix MPO700 mobile base, two Kuka LWR4 arms, two Shadow Hands and cameras. Only the right arm and the mobile base were used for this experiment. The external wrench $\mathbf{h}_{ext}$ is estimated through the FRI interface[2]. The implementation has been realized on a computer with an i7-6700HQ processor running Linux with the *Realtime Preemption patch*[3]. All the code was written in C++ using the Knowbotics Framework, currently under development at LIRMM. The FRI library was used to communicate with the Kuka arm while a UDP bridge was set up to send cartesian velocities to the computer embedded in the mobile base. The controller sample time was T=1 ms and the average computation time was 0.15 ms. Due to technical limitations in the mobile base low level controller, this was only updated every 25 ms.

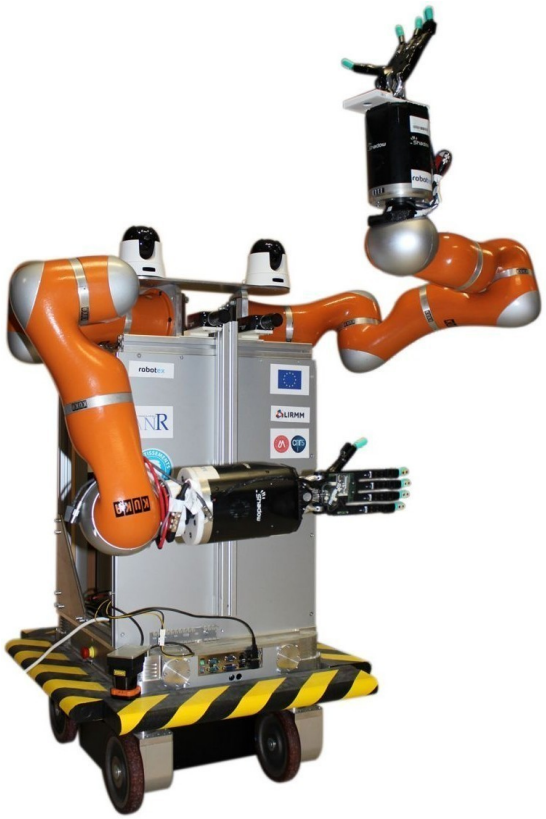


Figure 4.5 – The Bazar mobile manipulator.

[1]Bimanual Agile Zany Anthorpomorphic Robot.
[2]`http://cs.stanford.edu/people/tkr/fri/html`
[3]`https://rt.wiki.kernel.org/index.php/Main_Page`

(a) Base is fixed, arm is moving. $t < 12s$

(b) Base is moving, arm is fixed. $12s < t < 30s$

(c) Base is fixed, arm is moving. $30s < t < 38s$

(d) Both base and arm are moving. $38s < t < 47s$

(e) Base is fixed, arm is moving. $t > 47s$

Figure 4.6 – Snapshots of the experiment.

4.1.13 Results

For this experiment, all three constraints were used $C = \{s, m, d, a\}$, with the following parameters:

- distance to singularity constraint: $\epsilon = 0.1$, $\lambda_{max} = 0.1$,
- manipulability constraint: $m_{th} = 0.04$, $m_{min} = 0.02$, $\varphi = 0.2$,
- allowed workspace constraint: $d_{th} = 0.05$ m, $d_{min} = 0.001$ m, $\theta_{max} = 1$ rad, $\theta_{th} = 0.5$ rad,
- deactivation mechanism: $t_{end} - t_{start} = 2$ s.

Results from this experiment are shown in Figures 4.6, 4.7 and in the video attached to this manuscript[1]. During the first 12 seconds the operator moves the arm freely and the mobile base stays fixed (Fig. 4.6a). Then, since the TCP approaches a singular configuration, the manipulability and singularity constraints are activated to transfer the velocities to the mobile platform. This allows the operator to move the robot to another location. During this phase, both translations and rotations of the mobile base are performed in order to reach the desired configuration (Fig. 4.6b). At $t = 30s$, the deactivation mechanism is triggered and the operator can again control the manipulator (Fig. 4.6c). At $t = 38s$, the end effector is pushed toward the mobile base, activating the workspace constraint. Until $t = 47s$, the mobile platform moves backward but the arm is still allowed to move in the unconstrained directions (Fig. 4.6d). Finally, the deactivation mechanism is enabled a second time to stop the mobile base and unconstrain the arm motion (Fig. 4.6e).

[1]Also available at `https://youtu.be/zEp2ycuUvcU`

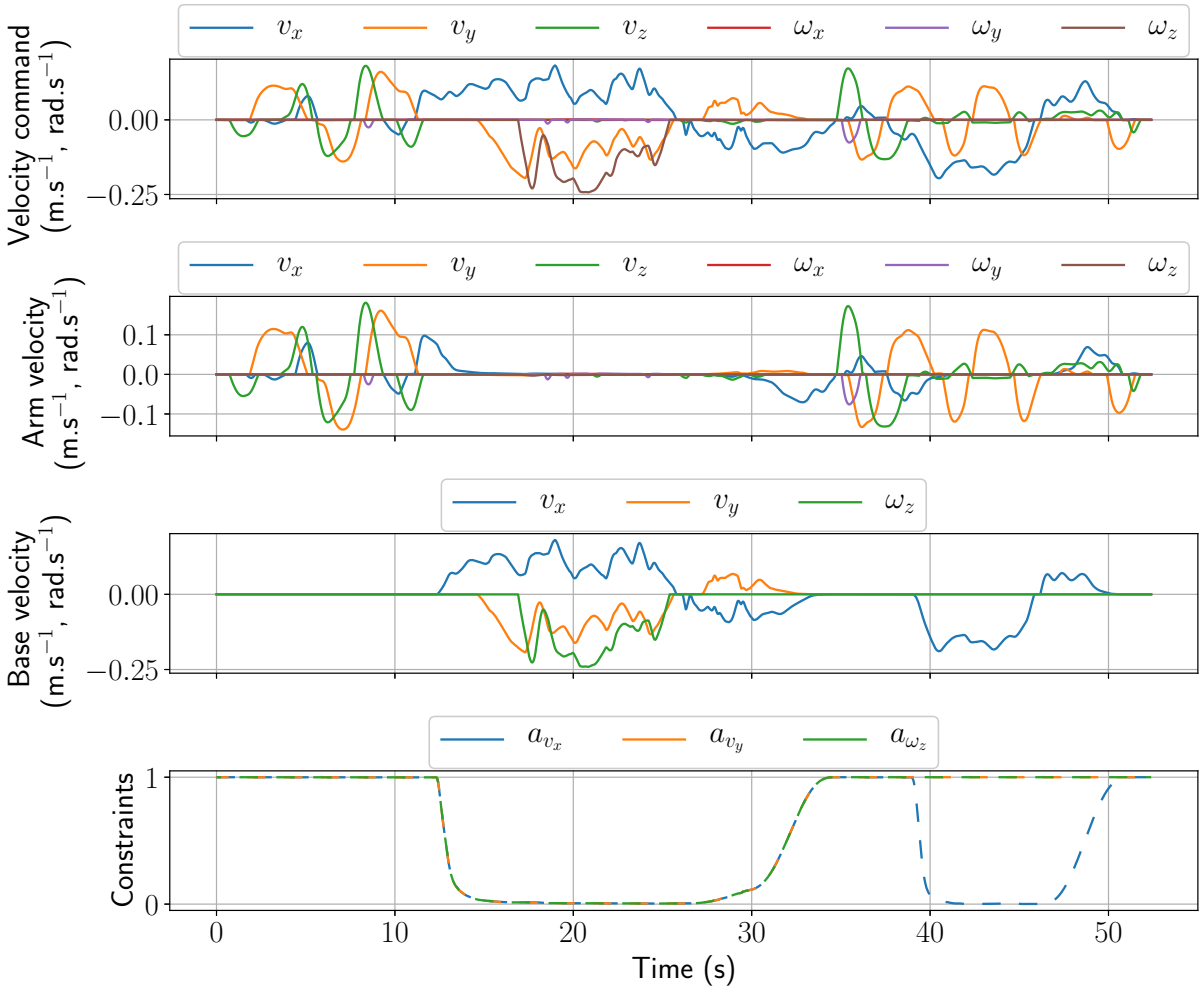


Figure 4.7 – Experimental results. From top to bottom: velocity command $\dot{\mathbf{x}}$ (m/s, rad/s), arm velocity command $\dot{\mathbf{x}}_{arm}$ (m/s, rad/s), base velocity command $\dot{\mathbf{x}}_{base}$ (m/s, rad/s) and constraint values a_{v_x}, a_{v_y} and a_{ω_z}.

4.1.14 Conclusion on mobile comanipulation

The key point of this section is a new redundancy solution for mobile manipulators to enhance physical human-robot collaboration. The tool velocity can be modified through a reference trajectory or by interaction forces applied by a human operator. Four constraints have been proposed to exploit redundancy: distance to singularity, minimum of manipulability, distance to objects and angular deviations. The framework has been validated in both simulated and real environments. In future work, we will study different scenarios where new constraints may be needed. These include navigation in cluttered environments (e.g., for assistance to disabled people or to workers in factories).

4.2 Hand control

In this section, we will investigate how a robotic hand (the Shadow Dexterous Hand with five fingers) equipped with tactile sensors at the fingertips can be used in pHRI scenarios. In Sect. 4.2.1, we will see how tactile data can be processed in order to extract meaningful information, such as the point of contact and the applied force, or can be utilized as a way for the operator to communicate with the robotic system. In Sect. 4.2.2, we expose how the framework described in Sect. 3.1 can be used to grasp objects. Finally, two different experiments are presented in Sect. 4.2.3.

4.2.1 Tactile sensing

Before any control or tactile communication can take place, fingertip sensors must be calibrated in order to extract meaningful information. In this work, we focused on the Syntouch[1] BioTac sensors. They are equipped with a pressure sensor to measure static pressure and vibrations, a temperature sensor for absolute temperature and heat flow measurements and an array of 19 electrodes that, thanks to a conductive fluid present between the electrodes and the flexible skin, relates to the skin's deformation. A BioTac sensor is represented, with its attached frame of reference, in figure 4.8.

Figure 4.8 – BioTac with its attached reference frame.

In order to enable force control at the fingertips and touch-based communication, we need to extract force information from the sensor. To do so, two steps are required. The first one is the calibration of the pressure sensor to establish a mapping between the raw static pressure data provided by the sensor and the contact force magnitude, which can be expressed as:

$$F_c = K_{PF}(P - P_0), \tag{4.24}$$

with P the measured pressure, P_0 the pressure offset and K_{PF} a scaling gain. Then, after a calibration, the electrode array can be used to reconstruct the point of contact. With with point of contact, we can estimate a three-dimensional force by scaling the surface normal vector at this point by the previously estimated force magnitude. This is done under the assumption that the force is applied in the direction of the surface normal vector (otherwise it would be impossible to reconstruct a three-dimensional force). The contact point is computed, as in [67], using:

$$(x_c, y_x, z_c) = \frac{\sum_{i=1}^{19} |e_i^*|^2 (x_i, y_i, z_i)}{\sum_{i=1}^{19} |e_i^*|^2}, \tag{4.25}$$

[1] `https://www.syntouchinc.com/`

with $e_i^* = e_i - e_{i,0}$ being the i-th electrode value with its offset removed and (x_i, y_i, z_i) its position in the BioTac frame. An illustration of a BioTac sensor with its electrodes visible is given in Fig. 4.9.

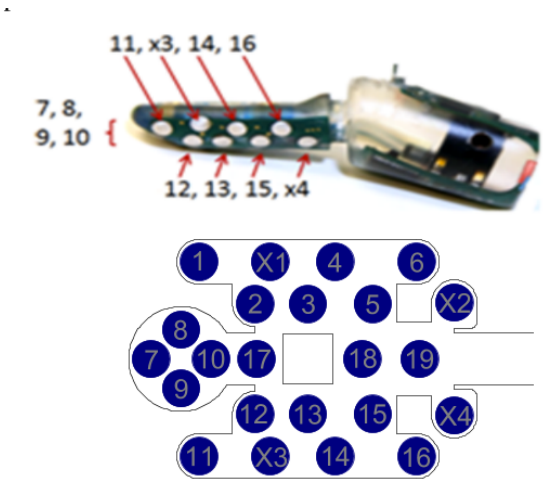


Figure 4.9 – Electrodes on a BioTac sensor.

The BioTac shape is composed of a cylindrical part for $x < 0$ and a spherical part for $x \geq 0$. Knowing this, we can compute the normal vector at the contact point:

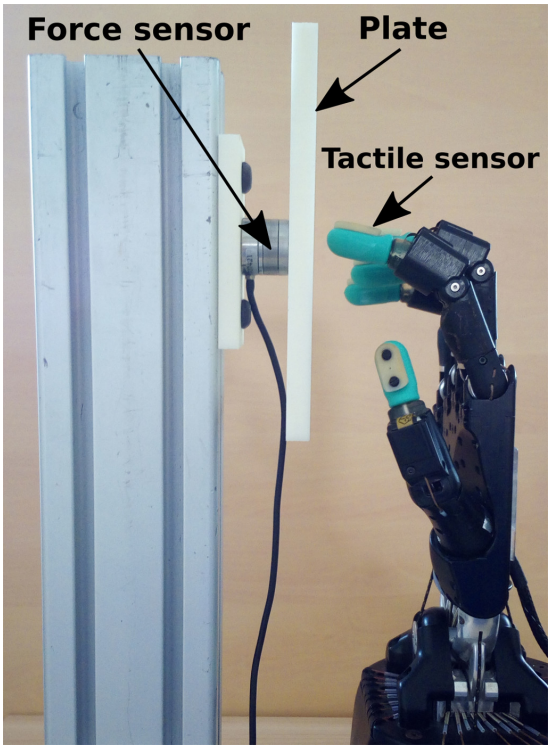
$$\boldsymbol{\eta}_c = \begin{cases} \begin{pmatrix} 0 \\ \cos(\theta_{cyl}) \\ -\sin(\theta_{cyl}) \end{pmatrix} & \text{if } x < 0 \\ \begin{pmatrix} \sin(\theta_{sph}) * \cos(\phi_{sph}) \\ \sin(\theta_{sph}) * \sin(\phi_{sph}) \\ \cos(\theta_{sph}) \end{pmatrix} & \text{otherwise.} \end{cases}, \tag{4.26}$$

with $\theta_{cyl} = \arctan(-z_c/y_c)$, $\theta_{sph} = \arctan(\sqrt{x_c^2 + y_c^2})$ and $\phi_{sph} = \arctan(y_c/x_c)$. Knowing both $\boldsymbol{\eta}_c$ and F_c allows us to estimate the three-dimensional force applied to the sensor:

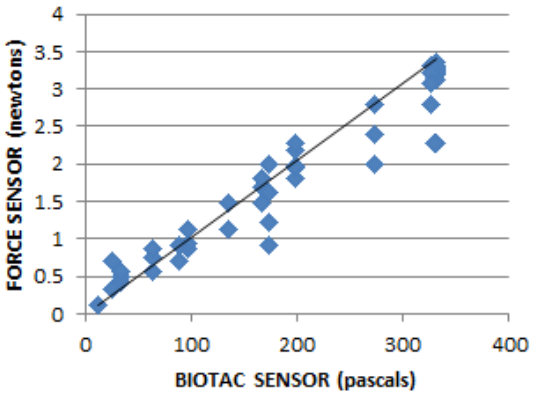
$$\mathbf{f}_c = F_c \boldsymbol{\eta}_c. \tag{4.27}$$

In order to use equations (4.24)-(4.27) we need to identify P_0, K_{PF} and $\mathbf{e}_0$, i.e., a total 21 parameters for each BioTac sensor. To this end, we set up an experiment where a force plate is placed in front of a robotic hand equipped with a BioTac sensor on each fingertip (see Fig. 4.10a). Then, during the experiment, the hand presses the plate which each finger sequentially. Three sequences, each with a distinct level of pressure, are performed to establish the relationship between the BioTac pressure data and the force measured by force sensor $\mathbf{f}_{fs}$. The offset values P_0 and $\mathbf{e}_0$ are obtained by averaging the sensor values when the hand is still and not in contact with the plate. Plotting $\mathbf{f}_{fs}$ against $P - P_0$ clearly shows a linear mapping between the two, as seen in figure Fig. 4.10b. K_{PF} can then be derived using linear regression. More detailed information can be found in [1].

To enable touch-based communication, the tactile sensors can be used as buttons to trigger some events (e.g., to start to grasp an object). To do so, we implement a



(a) BioTac pressure sensor calibration setup.



(b) Pressure to force relationship for a BioTac sensor.

Figure 4.10 – BioTac sensor calibration.

comparator with hysteresis to detect if the sensor has been pressed:

$$C(t) = \begin{cases} 1 & \text{if } F > F_H \\ 0 & \text{if } F < F_L \\ C(t - T_s) & \text{otherwise,} \end{cases} \tag{4.28}$$

where F_H and F_L ($F_H > F_L > 0$) are the pre-tuned high and low thresholds at which the state changes. With this triggering system, the operator can command the robot during interaction without the need of an external, sophisticated interface.

4.2.2 Grasp motion and force control

To successfully grasp an object, we first need to generate a trajectory to make contact with it, then apply sufficient force to avoid any slippage. To do so, we apply the framework described in 3.1 to each finger separately, to generate the grasping motion and to regulate the fingertips contact forces. Hence, for each finger, we apply Eq. (3.19) with $\dot{\mathbf{x}}_{fc} \in \mathcal{V}$ and $\dot{\mathbf{q}}_r \in \omega$. The choice of having joint space trajectories instead of task space ones is due to the fact that the default configuration ${}^o\mathbf{q}^*$ for a hand (open with finger extended) presents kinematic singularities. With the proposed approach, only the force control is expressed in the task space and, since this happens when the fingers are in a closed configuration ${}^c\mathbf{q}^*$, the kinematic singularities do not cause any problem. A simple finite state machine (as the one shown in Fig. 4.11) can be designed to handle the grasping.

The first step consists in opening the hand, allowing the object to be handed over to the hand. Then, the fingers are driven to a configuration known to be close to contact with the object. Once that configuration is reached, a non-zero target force $\mathbf{f}_{grasp}$ is set, forcing the hand to make contact with the object and to apply a force sufficient to hold the object. When the object has to be released, $\mathbf{f}_{grasp}$ is set to zero before the hand is brought back to its open configuration. The close configuration ${}^{c}\mathbf{q}^{*}$ and the amount of force needed to grasp the object $\mathbf{f}_{grasp}$ has to be determined experimentally.

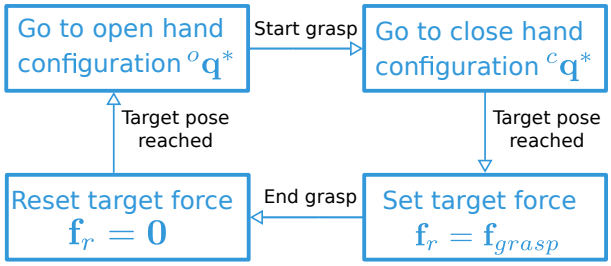


Figure 4.11 – Grasping FSM

4.2.3 Test cases

We will detail two applications based on the previously exposed tactile sensing and hand control. The first one consists in controlling a robotic hand via electromyography (EMG) on tetraplegic patients as a training before performing electromyostimulation (EMS) on their own hand. The second experiment consists of a robotic hand mounted at the end-effector of a serial manipulator to grasp and actuate an electric screwdriver during a physical collaboration task. In both applications, the system is composed of a Shadow Dexterous Hand[1] equipped with Syntouch BioTacs sensors at the fingertips. A ROS interface[2] was used to control the Shadow Hand and get the BioTacs measurements. Kinematic computations were performed using the Robotics Library[3].

EMG hand control

The goal of this experiment, depicted in figure Fig. 4.12, was to determine if patients suffering from tetraplegia could use some of their still functioning upper body muscles[4] to trigger EMS on their own hand to restore functional motion. The complete results were published in [4]. Here, we will only discuss the part that is relevant to this thesis.

When a muscle is stimulated (electrically or neurologically), its cells generate an electric potential. This potential can be measured using surface or intramuscular electrodes and it reflects the muscle activity: the stronger the contraction, the higher the signal. EMG signals need to be filtered before being used. Here, we are interested in the muscle contraction level, so we need to extract the envelope of the signal. The filtering process consists of a high-pass filter (Butterworth, fourth-order, 20Hz) followed by a low pass filter (Butterworth, fourth-order, 2Hz) on the absolute value of the previous filter's output

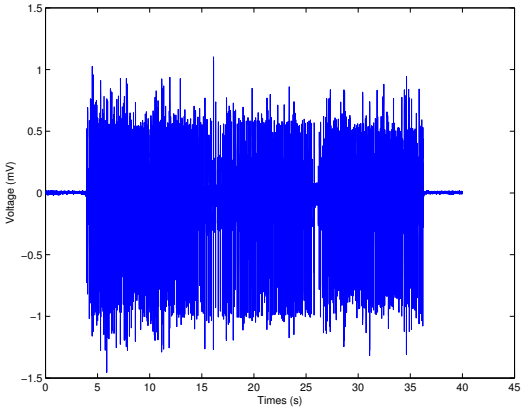
[1] `https://www.shadowrobot.com/products/dexterous-hand/`
[2] `http://wiki.ros.org/shadow_robot`
[3] `http://www.roboticslibrary.org`
[4] Some of them can be contracted but do not induce motion.

Figure 4.12 – EMG controlled robotic hand setup

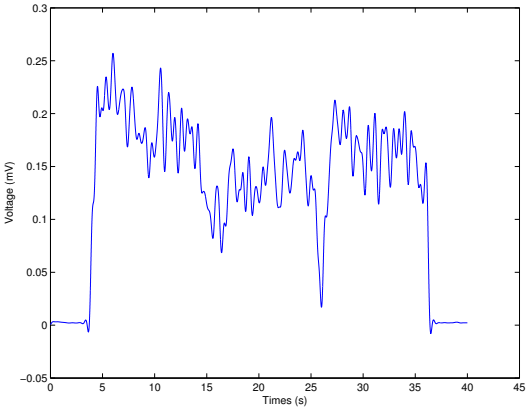
signal. Illustrations of this filtering process on two different EMG signals are given in figure Fig. 4.13. Then, the filtered signal is normalized using the value obtained from a maximum contraction, to ensure homogeneous values between patients. Since valid muscles differ between patients, a first identification phase was necessary. During this phase, electrodes were put on the skin above each muscle and the patient was asked to contract it as much as possible. Then, we determined if the signal was high enough above the noise level to be usable.

In case the muscle was usable, the patient used it to control the closing of the robotic hand. Two different grasps, palmar pinch and key grip, shown in figure Fig. 4.14, were programmed and five different modes of control were investigated, as summed up in table 4.1. For continuous contractions, the contraction must be held to keep the hand closed. In impulse modes, the first contraction triggers the closing of the hand while the second one triggers its opening. For these two modes, a comparator with hysteresis is used to detect if the muscle is contracted or not. The thresholds for the comparator were adjusted on a patient and muscle basis, depending on how easy or difficult it was for the patient to contract a given muscle. In the proportional mode, the stronger the contraction, the more the hand closes. When using two muscles, the patient could choose between the two available grasps. As an illustration, the FSM for mode 5 is given in figure Fig. 4.15. Tactile sensing was also used to stop the grasp once a force threshold was reached, i.e $C(t) = 1$. It allowed the patient to trigger grasps on different objects without the fear of damaging them. A video of the experiment is attached to this document[1]. This experiment was run on two different patients. Later on, a simplified version using only one control mode has been conducted on 10 patients over three months to assess their muscles functioning and to train them before performing EMS on their own hands. All these experiments were conducted at the Propora clinic in Montpellier, France.

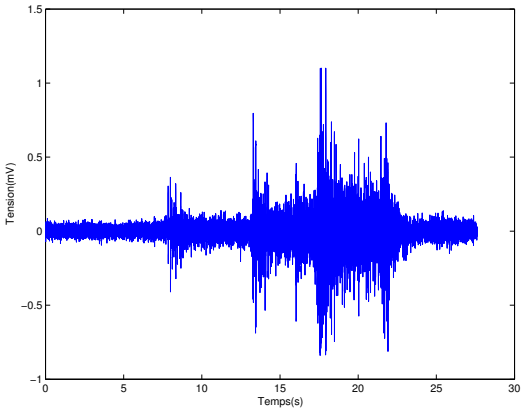
[1]Also available at `https://youtu.be/nohOt01bW0Q`



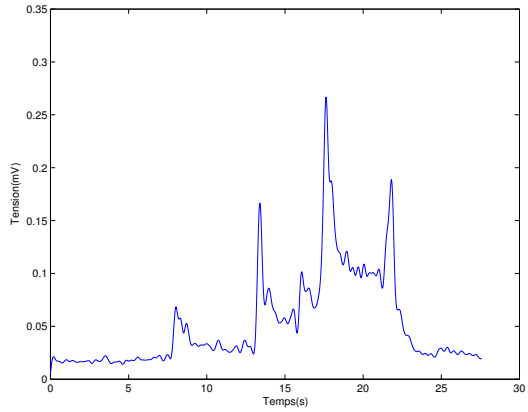
(a) Biceps raw EMG signal.



(b) Biceps filtered EMG signal.



(c) Trapezius raw EMG signal.



(d) Trapezius filtered EMG signal.

Figure 4.13 – Raw and filtered EMG signals.

Mode	Description
1	Continuous contraction, one muscle
2	Impulse, one muscle
3	Proportional, one muscle
4	Impulse, two muscle
5	Continuous contraction, two muscle

Table 4.1 – EMG-based control modes.

Figure 4.14 – Hand configurations. From left to right: open hand, palmar pinch and key-grip.

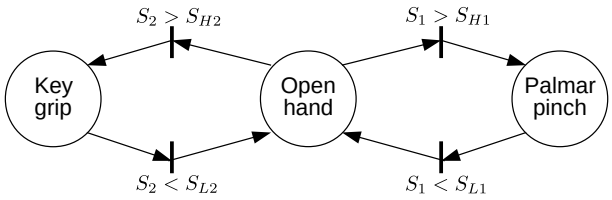


Figure 4.15 – EMG based control mode 5

Collaborative screwing

The second experiment aimed at developing a robotic solution to provide assistance to an operator during a screwing task. The system was composed of a Kuka LWR4+ arm with a Shadow Dexterous Hand mounted at the end-effector.

The arm was under admittance control using (3.19), with the external forces ($\mathbf{f}_{ext} \in \mathcal{F}$) used to move the tool to the screwing positions and a reference velocity ($\dot{\mathbf{x}}_r \in \mathcal{V}$) needed to bring the robot back to its starting position at the end of the task. Virtual stiffness ($\mathbf{K}_t \in \mathcal{F}$) was also used at specific moments, as will be explained later.

As specified by the ISO10218 standard, velocity (3.33), power (3.39) and force (3.40-3.41) constraints were also applied to guarantee the operator's safety. The task space damping matrix was set to $\mathbf{B}_t = \text{diag}\{250, 250, 250, 20, 20, 20\}$ except in state (F) where it is incresed along the non-screwing axes, the stiffness is set to
$\mathbf{K}_t = \text{diag}\{500, 500, 500, 50, 50, 50\}$ through states (A) to (D) and to $\mathbf{K}_t = \mathbf{0}$ in the other ones. The hand was driven using the methodology described in 4.2.2. For the contact detection, the thresholds were set to $F_H = 2.1$ N and $F_L = 1.9$ N. A finite state machine was designed to handle the various steps required to perform the task. Most of the state changes are driven by the touch interface signal C (4.28), either on its rising edge $\uparrow C$ or on its falling edge $\downarrow C$. We detail below the steps involved for this task and represent their transitions in Fig. 4.17:

(A) The tool moves to an initial fixed pose $\mathbf{x}^*$, while the hand is still.

(B) The tool is still while the fingers are reset to their default configuration (${}^o\mathbf{q}^*$).

(C) The tool is still while the hand prepares for grasping (fingers driven to an intermediate ${}^p\mathbf{q}^*$). The operator hands the screwdriver to the robot before triggering the next step.

(D) The tool is still while the hand grasps the screwdriver (fingers driven to ${}^c\mathbf{q}^*$ before force control is activated).

(E) The operator can translate and reorient the tool using (3.11), with $\dot{\mathbf{x}}_r = \mathbf{0}$. When the desired screwing position is reached, the operator triggers the next step.

(F) Tool motion is tolerated only along the screwdriver direction (z axis of the tool frame), by setting $\mathbf{B}_t = \text{diag}\{10^5, 10^5, 250, 10^5, 10^5, 10^5\}$.

(G) When the operator presses the thumb, the middle finger moves to power the screwdriver. Since the middle finger's BioTac does not touch the screwdriver, the finger is controlled in open-loop. When the thumb is released, we move to state (H).

(H) The tool is stopped. If the previous press was long, we go back to (F) to continue the screwing on the same axis, otherwise the FSM moves to (I).

(I) The operator can choose (via the thumb pressure duration) between pursuing screwing at another location (E) or making the tool return to its initial position (A).

In Fig. 4.17, the force applied to the thumb tactile sensor, the short and long press signals as well as the FSM states are displayed. The bottom graph is a close up between 30

and 33 seconds to highlight the trigger signals changes. We can see that all the changes in the FSM are correctly performed using the thumb presses and that such a simple interface can be used even in moderately complex scenarios. Fig. 4.16 shows the evolution of the little finger's contact force before and after the tool is grasped. A large overshoot can be observed before the force is regulated to its target value. This is due to the embedded joint position controller that cannot accurately track the joint position command and react fast enough to the sudden changes in the external force, mainly because of frictions and flexibilities inherent to cable-driven robots. A video of the experiment is joint to this manuscript[1].

4.2.4 Conclusion on hand control

In this section, we proposed an extension to the collaborative framework presented in chapter 3 for a robotic hand, allowing object grasping with contact force regulation but also tactile communication. Both aspects rely on the presence of tactile sensors at the fingertips. A methodology to extract meaningful data from such sensors has been detailed, allowing the estimation of both the direction and the amplitude of the contact force as well as the derivation of basic human intentions through tactile presses. These aspects have been applied in two different scenarios, for the control of the hand via EMG signals by tetraplegic patients and for collaborative screwing task.

[1] Also available at `https://youtu.be/l8llthn_B4I`

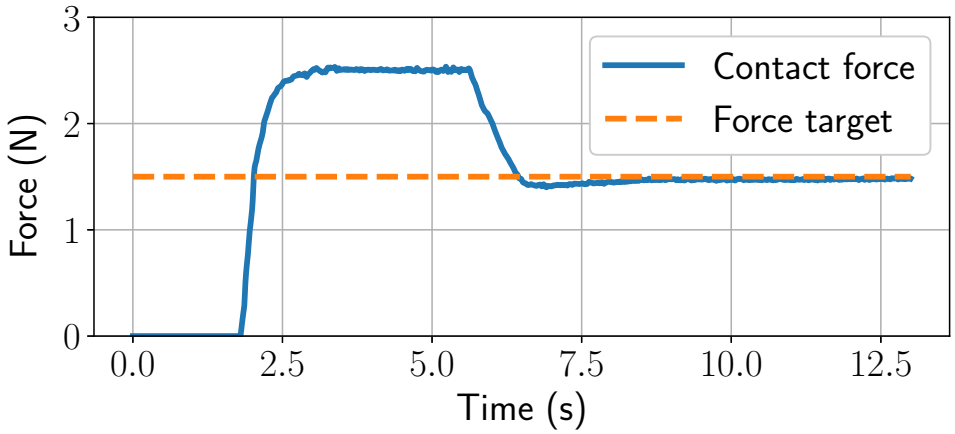


Figure 4.16 – Force regulation on the little finger (Z axis).

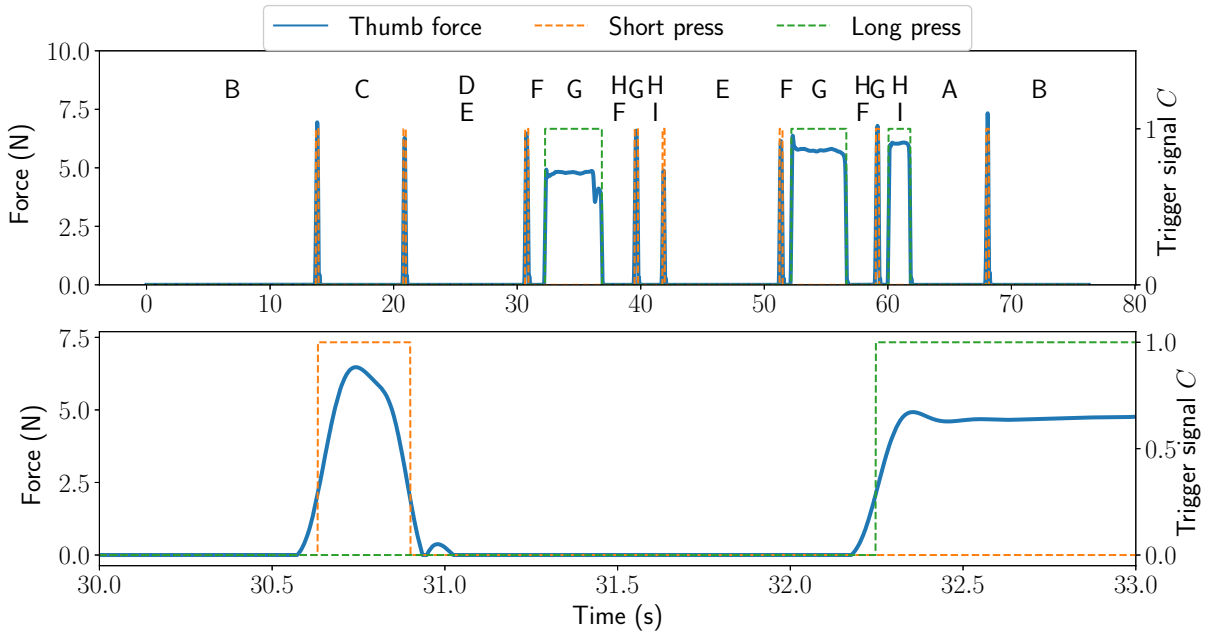


Figure 4.17 – FSM states during the collaborative screwing experiment. The bottom figure is a close up between 30 and 33s to highlight the thumb presses detection.

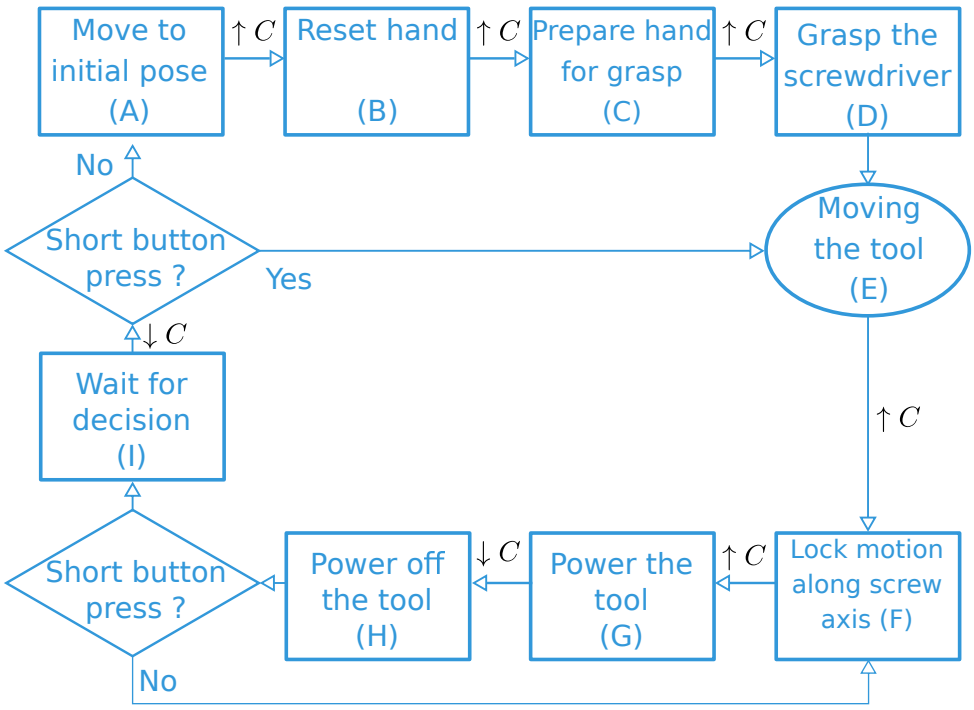


Figure 4.18 – FSM for the collaborative screwing experiment.

4.3 Conclusion

We proposed two extensions of our safe damping controller presented in Chapter 3 to enable the control of mobile manipulators and robotic hands in addition to serial manipulators. For the mobile manipulators with an omnidirectional base, the proposed solution splits the realization of the end-effector task between the arm and the base based on a set of criteria to mimic a human behavior. Experiments, both in simulation and with the real platform, have been conducted to prove the effectiveness of our solution. To achieve the control of a dexterous robotic hand, we first explained how tactile sensors at the fingertips can, once properly calibrated, be used to reconstruct a three-dimensional contact force. Then, we proposed a finite state machine that uses both our safe damping controller and the newly estimated force to perform grasps on an object. This work has been showcased in two distinct experiments, one where the robotic hand was used as part of study aiming to restore functional hand motion for tetraplegic subjects and a second one where the hand, mounted at a manipulator end-effector, is used to grasp and actuate an electric screwdriver and to detect human intentions using the force applied to the thumb tactile sensor.

Conclusion

In this thesis, we explored physical human-robot interaction and collaboration and how control can bring safety to these scenarios.

We first recalled the importance of a dynamic model for implementing low level torque control, estimating the interaction forces and deriving some interesting safety criteria, such as the reflected inertia. We showed that, when using a computed torque method to achieve position control, the Coulomb friction model can hardly be used to describe the joint dry friction of real robots, such as the Kuka LWR4+. The proposed solution is a PD based model-free compensator with a very limited output range, that allows a quick compensation of the errors induced by the dry frictions, without altering the desired external perturbation behavior, and instead allowing accurate non-stiff joint positioning.

Then, we exploited damping control, a special case of admittance control, to build a framework to design collaborative tasks while ensuring safety criteria. In this framework, collaborative tasks can be described in either the joint or task space, using input velocities and forces/torques. Safety constraints are introduced through velocity reduction, allowing emergency stops, limitation of the joint and/or tool velocity, acceleration and power. It is also possible to limit the force and the kinetic energy at the end-effector. Separation distance monitoring, when available, allows to tune these limitations online, to adopt a safer behavior when a human is close and to relax the limitations when no one is present in the workspace. As mentioned earlier in this thesis, it seems that no single solution capable of dealing with the four collaborative operations defined by the ISO15066 exists to this day. To remedy this issue, we developed an open source library, called OpenPHRI, that implements everything presented in this chapter and provides an interface for the V-REP simulation software. The library was proven to be very efficient in terms of computation and memory footprint, allowing execution at 1kHz or more.

In the last chapter, we extended this control framework to both omnidirectional mobile manipulators and robotic hands. For mobile manipulators, the goal was to mimic a human-like behavior by leaving the base fixed when the arm can execute the task by its own and moving it only when necessary, to reach a distant object or to avoid self collision for example. To cope with this and the task space redundancy introduced by the mobile base, we proposed to split the velocity generated at the end-effector between the arm and the base according to a set of constraints. These constraints are based on the arm's kinematic singularities and on the manipulability index, workspace limitations and orientation deviation. The default behavior is to move only the arm but, when a constraint is approaching, velocities start to be transferred to the mobile base. When any of the given constraints are reached, the arm is completely stopped and only the mobile base is used to perform the task. Also, a generic constraint deactivation strategy has been designed to give the control back to the arm when the operator is moving

the robot away from a constraint. In the second part of this chapter, we looked at the estimation of external forces using tactile sensors mounted on the fingertips of a robotic hand and at how to provide a grasping strategy and contact forces regulation, based on the work presented in chapter 3. This work was showcased in two different applications, a collaborative screwing task and EMG-based control for tetraplegic patients.

Even if the solutions provided in this thesis aim for better physical human-robot collaborations, many things need to be done before such interactions can be fully considered for real world applications. First, hardware must improve while reducing the costs. Collaborative robots with flexible skin and/or joints are yet to be found in robot manufacturers' catalogs but are a necessary step to make robot intrinsically safer to work or interact with. Better sensing capabilities would benefit a lot to collaborative applications, with, for example, a tactile skin to sense the location and intensity of a contact force anywhere on the robot or even capacitive sensing to detect the direct proximity of humans without relying on cameras or laser scanners. All this technology already exists to some extent, but has not reached the market yet, limiting its spread.

Then, considering control, a huge amount of solutions have been proposed to solve specific problems, but we are still missing a common framework to tackle any collaborative task, slowing down the progress in the area. Moreover, the lack of open source software makes the integration of previously published work harder and can quickly become very time consuming.

OpenPHRI is a first attempt to bridge this gap but it is not perfect. Being solely based on kinematics it can not include dynamics-based constraints, such as torque limitations, or incorporate works from others that rely on a dynamic model and on torque control. These issues will be investigated for future versions of OpenPHRI, to make the library usable in more cases.

While physical human-robot interactions are still mainly focused on serial manipulators, it is important to consider other robot structures to be able to bring a larger set of innovative solutions to current robotics problems. We investigated the use of mobile manipulators with an omnidirectional base during physical collaborations but non-omnidirectional bases must also be considered since differential drive is common in currently available robots, thanks to its simplicity and low cost. For non-omnidirectional bases, non-holonomic constraints have to be accounted for. This will lead eventually to a more complex solution that the one we proposed here. Regarding robotic hands, a device fully covered with a tactile skin would help to have a better sense of the grasped object and would make tactile signing possible, allowing more detailed intention communication to the robot. Miniaturization of the motorization would make light[1] and dexterous hand design possible, allowing these to be incorporated in other robots, typically humanoid.

Although improvements on control, software and hardware are required before we can have access to truly collaborative robots, we still believe that this thesis contributed to make robots easier to integrate, when designing collaborative applications and safer to work with. We also hope that this work will contribute positively to the future of collaborative robots.

[1]The Shadow Dexterous Hand weighs around 4.5kg.

Appendix A

Polynomial interpolation and trajectory generation

This appendix will present how fifth-order polynomials are used in this thesis for smooth interpolation and trajectory generation under velocity and acceleration constraints.

A.1 Fith-order polynomials

The order of a polynomial used for interpolation or trajectory generation is given by the number of constraints to satisfy. In our case, we need to impose initial and final values as well as their first and second derivatives. This leads to six constraints that can be satisfied by a polynomial composed of six parameters, hence the use of fifth-order polynomials.

In this section we will recall how such functions are described and how their parameters can be computed to satisfy the given constraints. The equations of a fifth-order polynomial and its two first derivatives are given in (A.1)-(A.3):

$$\mathcal{P}(t, \mathbf{c}) = [t^5 \ t^4 \ t^3 \ t^2 \ t \ 1] \ \mathbf{c} \tag{A.1}$$
$$\dot{\mathcal{P}}(t, \mathbf{c}) = [5t^4 \ 4t^3 \ 3t^2 \ 2t \ 1 \ 0] \ \mathbf{c} \tag{A.2}$$
$$\ddot{\mathcal{P}}(t, \mathbf{c}) = [20t^3 \ 12t^2 \ 6t \ 2 \ 0 \ 0] \ \mathbf{c}, \tag{A.3}$$

where $t \in \mathbb{R}$ and $\mathbf{c} = [a \ b \ c \ d \ e \ f]^\top \in \mathbb{R}^6$ are the vector of the polynomial coefficients. Then, we define the initial and final constraints as:

$$\mathcal{P}(0) = \mathcal{P}_i \qquad \mathcal{P}(T) = \mathcal{P}_f \tag{A.4}$$
$$\dot{\mathcal{P}}(0) = \dot{\mathcal{P}}_i \qquad \dot{\mathcal{P}}(T) = \dot{\mathcal{P}}_f \tag{A.5}$$
$$\ddot{\mathcal{P}}(0) = \ddot{\mathcal{P}}_i \qquad \ddot{\mathcal{P}}(T) = \ddot{\mathcal{P}}_f, \tag{A.6}$$

with $T \in \mathbb{R}_{>0}$. Using $t \in [0, T]$ instead of the more general form $t \in [T_1, T_2]$ allows simpler expressions and solutions as well as faster computations. To compute the polynomial

coefficients, we put the problem in matrix form:

$$\begin{bmatrix} \mathcal{P}_i \\ \dot{\mathcal{P}}_i \\ \ddot{\mathcal{P}}_i \\ \mathcal{P}_f \\ \dot{\mathcal{P}}_f \\ \ddot{\mathcal{P}}_f \end{bmatrix} = \mathbf{A}\mathbf{c} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ T^5 & T^4 & T^3 & T^2 & T & 1 \\ 5T^4 & 4T^3 & 3T^2 & 2T & 1 & 0 \\ 20T^3 & 12T^2 & 6T & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}, \tag{A.7}$$

that can then be solved with:

$$\mathbf{c}(T, \mathcal{P}_i, \dot{\mathcal{P}}_i, \ddot{\mathcal{P}}_i, \mathcal{P}_f, \dot{\mathcal{P}}_f, \ddot{\mathcal{P}}_f) = \mathbf{A}^{-1} \begin{bmatrix} \mathcal{P}_i \\ \dot{\mathcal{P}}_i \\ \ddot{\mathcal{P}}_i \\ \mathcal{P}_f \\ \dot{\mathcal{P}}_f \\ \ddot{\mathcal{P}}_f \end{bmatrix}. \tag{A.8}$$

It can be found that the determinant of $\mathbf{A} = -4T^9$, leading to the matrix always being invertible since T is strictly positive. Once the coefficients have been computed, the evaluation of the polynomial and its first and second derivatives can be obtained using (A.1)-(A.3). When generating multiple polynomials at the same time, the output vectors can be obtained using:

$$\boldsymbol{\mathcal{P}}(t, \mathbf{c}_{traj}) = \begin{bmatrix} \mathbf{c}_0^\top \\ \mathbf{c}_1^\top \\ \vdots \\ \mathbf{c}_p^\top \end{bmatrix} \begin{bmatrix} t^5 \\ t^4 \\ \vdots \\ 1 \end{bmatrix} \tag{A.9}$$

$$\dot{\boldsymbol{\mathcal{P}}}(t, \mathbf{c}_{traj}) = \begin{bmatrix} \mathbf{c}_0^\top \\ \mathbf{c}_1^\top \\ \vdots \\ \mathbf{c}_p^\top \end{bmatrix} \begin{bmatrix} 5t^4 \\ 4t^3 \\ \vdots \\ 0 \end{bmatrix} \tag{A.10}$$

$$\ddot{\boldsymbol{\mathcal{P}}}(t, \mathbf{c}_{traj}) = \begin{bmatrix} \mathbf{c}_0^\top \\ \mathbf{c}_1^\top \\ \vdots \\ \mathbf{c}_p^\top \end{bmatrix} \begin{bmatrix} 20t^3 \\ 12t^2 \\ \vdots \\ 0 \end{bmatrix}, \tag{A.11}$$

with $p \in \mathbb{N}$ the number of polynomials.

Trajectories should often be described using multiple waypoints. To deal with this, we can split the trajectory into segments, each represented by a polynomial. This can be translated to:

$$\mathcal{P}(t, \mathbf{c}_s) = [t^5 \ t^4 \ t^3 \ t^2 \ t \ 1] \begin{cases} \mathbf{c}^0 & \text{if } 0 \leq t \leq T_0 \\ \mathbf{c}^1 & \text{if } T_0 < t \leq T_1 \\ \vdots & \\ \mathbf{c}^q & \text{if } T_{q-1} < t \leq T_q \end{cases}, \tag{A.12}$$

with $q \in \mathbb{N}$ the number of segments. To avoid discontinuities in the trajectory, we impose the following condition:

$$\dot{\mathcal{P}}_i^k = \dot{\mathcal{P}}_f^{k-1} \tag{A.13}$$

$$\ddot{\mathcal{P}}_i^k = \ddot{\mathcal{P}}_f^{k-1}, \tag{A.14}$$

$\forall k \in [1, q]$.

A.2 Interpolation

To perform smooth interpolations, we define the following function based on the polynomial described in Sect. A.1:

$$f_{int}(x, x^-, x^+, y^-, y^+) = \begin{cases} y^- & \text{if } x \leq x^-, \\ y^+ & \text{if } x \geq x^+, \\ \mathcal{P}(t, \mathbf{c}(T, y^-, 0, 0, y^+, 0, 0)) & \text{otherwise.} \end{cases} \tag{A.15}$$

with $t = x - x^-$ and $T = x^+ - x^-$. Imposing $\dot{\mathcal{P}}_i = \ddot{\mathcal{P}}_i = \dot{\mathcal{P}}_f = \ddot{\mathcal{P}}_f = 0$ gives us a smooth function for all real value x, as depicted in Fig. A.1, where the evolution of $f_{int}(x, 0.5, 2, 0, 0.25)$ for $x \in [0, 2.5]$ is given. It can be seen that the constraints are respected and that f_{int} can be used when smooth interpolation is needed.

A.3 Trajectory generation

We will detail four complementary methods used in this work to generate task or joint space trajectories.

The first method, presented in Sect. A.3.1, allows to generate polynomial-based trajectories under velocity and acceleration constraints with arbitrary initial and final position, velocity and acceleration. Similar solutions, such as the Reflexxes Motion Library [68], are already available, with the main differences being bang-bang acceleration profiles, no notion of waypoints to build complex trajectories and that they are only usable with rotations described by Euler angles. Bang-bang acceleration profiles will produce the shortest trajectories between two points, but are very demanding on the robot actuators and can even cause them some damage due to their discontinuous nature. Also, smooth trajectories are preferable during human-robot interaction, since they provide a more natural motion. Using fifth-order polynomials results in smooth and acceleration-continuous trajectories, at the cost of a longer completion time.

The second method allows multiple trajectories, possibly each composed of several segments, to be synchronized. It will be detailed in Sect. A.3.2.

In Sect. A.3.3, a method to generate trajectories in task space using unit quaternions while maintaining translational and rotational velocities and accelerations under a given limit will be presented.

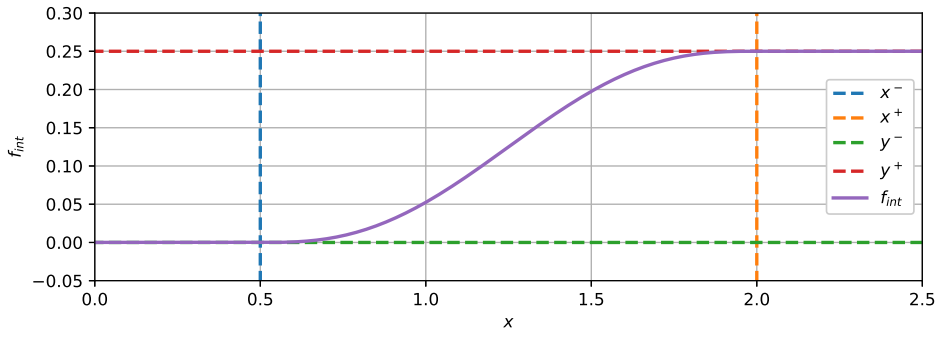


Figure A.1 – Interpolation function f_{int} for $x^- = 0.5$, $x^+ = 2$, $y^- = 0$ and $y^+ = 0.25$.

The last method monitors the tracking error to pause the trajectory generation when the error becomes too large, and to resume it when the robot is close enough to the target pose. This avoids the robot from "trying to catch up" with the trajectory if it has been stopped (Sect. 3.4.1), slowed down (Sections 3.4.2, 3.4.4, 3.4.6) or pushed away (Sections 3.2.1, 3.2.3).

A.3.1 Constrained trajectory generation

If a trajectory segment has to be completed in a given time, one can just use (A.8) to compute the coefficients of the polynomial. Instead, if the time is not constrained, but velocity and acceleration limits are given, T is a parameter to be determined.

Let us first consider the specific case of null initial and final velocities and accelerations:

$$\mathbf{c}(T, \mathcal{P}_i, 0, 0, \mathcal{P}_f, 0, 0) = \begin{bmatrix} \frac{6\Delta\mathcal{P}}{T^5} & -\frac{15\Delta\mathcal{P}}{T^4} & \frac{10\Delta\mathcal{P}}{T^3} & 0 & 0 & 0 \end{bmatrix}^\top, \tag{A.16}$$

with $\Delta\mathcal{P} = \mathcal{P}_f - \mathcal{P}_i$. Solving $\ddot{\mathcal{P}}(t_{v_{max}}, \mathbf{c}) = 0$ gives us the time at which the velocity is maximal, which is $t_{v_{max}} = \frac{T}{2}$. The maximum velocity is then determined by:

$$\max[\dot{\mathcal{P}}(\mathbf{c})] = \frac{30\Delta\mathcal{P}}{16T}, \tag{A.17}$$

which leads to the minimum time required to satisfy the velocity limit V_{max}:

$$T_{min,v} = \frac{30\Delta\mathcal{P}}{V_{max}}, \tag{A.18}$$

with $V_{max} \in \mathbb{R}_{>0}$. The same reasoning can be applied to the acceleration limit A_{max} resulting in:

$$T_{min,a} = \sqrt{\frac{10\sqrt{3}\Delta\mathcal{P}}{3A_{max}}}, \tag{A.19}$$

with $A_{max} \in \mathbb{R}_{>0}$. The minimum duration required for a segment to satisfy both constraints is:

$$T = \max(T_{min,v}, T_{min,a}). \tag{A.20}$$

When the initial and final velocities and accelerations are non zero, no trivial solution can be found for T. To solve this problem we can use Alg.2 that, given an initial value for T (set to 1 here, but could be any $T \in \mathbb{R}_{>0}$), will converge to the minimum time necessary to comply with both constraints. The first part will solve T for the velocity limit V_{max} and the second one for the acceleration constraint A_{max}. At the end, (A.20) is used to obtain the segment duration. In this algorithm, the minimum durations updates $(*)$ and $(**)$ for $T_{min,v}$ and $T_{min,a}$ are exact in the specific case described above and are assumed to be a good approximation in the general case. The maximum velocity $\max[\dot{\mathcal{P}}(\mathbf{c})]$ and acceleration $\max[\ddot{\mathcal{P}}(\mathbf{c})]$ for a given polynomial can be found using a zero search algorithm, such as [69]. Benchmarks of Alg.2 are presented in A.3.5.

$T_{min,v} = T_{min,a} = 1$
repeat
  $v_{max} = \left|\max[\dot{\mathcal{P}}(\mathbf{c}(T_{min,v}, \mathcal{P}_i, \dot{\mathcal{P}}_i, \ddot{\mathcal{P}}_i, \mathcal{P}_f, \dot{\mathcal{P}}_f, \ddot{\mathcal{P}}_f))]\right|$
  $\Delta_v = v_{max} - V_{max}$
  $T_{min,v} = T_{min,v} \ \frac{v_{max}}{V_{max}}$ $(*)$
until $|\Delta_v| < \varepsilon_v$;
repeat
  $a_{max} = \left|\max[\ddot{\mathcal{P}}(\mathbf{c}(T_{min,a}, \mathcal{P}_i, \dot{\mathcal{P}}_i, \ddot{\mathcal{P}}_i, \mathcal{P}_f, \dot{\mathcal{P}}_f, \ddot{\mathcal{P}}_f))]\right|$
  $\Delta_a = a_{max} - A_{max}$
  $T_{min,a} = T_{min,a} \ \sqrt{\frac{a_{max}}{A_{max}}}$ $(**)$
until $\Delta_a < \varepsilon_a$;
$T = \max(T_{min,v}, T_{min,a})$.

Algorithm 2: Segment minimum time computation.

A.3.2 Synchronization

When multiple trajectories must be generated simultaneously, some synchronization mechanisms should be used. This is illustrated in Fig. A.2, where two trajectories $\mathcal{P}_0$ and $\mathcal{P}_1$, each composed of two segments, are represented using no synchronization (top), waypoint synchronization (middle) and trajectory synchronization (bottom). For the the waypoint

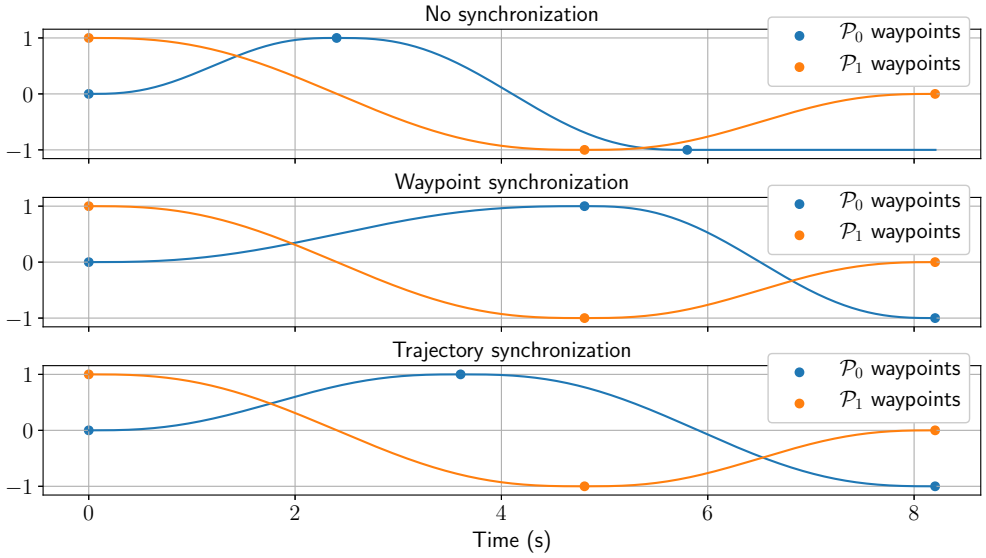


Figure A.2 – Comparison of the synchronization mechanisms.

synchronization, the duration of the shortest segments (the ones of $\mathcal{P}_0$ in the example) is increased to match with the longest one ($\mathcal{P}_1$ segments here). For the trajectory synchronization, each segment duration of the shortest trajectories ($\mathcal{P}_0$) is increased so that their

total duration matches the longest one ($\mathcal{P}_1$). This can be translated to:

$$T_j^k = \begin{cases} T_{min,j}^k & \text{if no synchronization} \\ \max(T^k) & \text{for waypoint synchronization} \\ T_{min,j}^k + \frac{\max(T_j) - T_{min,j}}{N} & \text{for trajectory synchronization,} \end{cases} \tag{A.21}$$

where T_j^k denotes the duration of the k-th segment of the j-th trajectory, $T_{min,j}^k$ is the minimum desired duration for the segment, $T_{min,j} = \sum_{k=1}^{N} T_{min,j}^k$ the minimum trajectory duration, $\max(T^k)$ the longest of the k-th segments, $\max(T_j)$ the longest of the trajectories and $N \in \mathbb{N}$ is the number of waypoints. When $T_{min,j}^k$ is computed to respect velocity and acceleration constraints (i.e., output by Alg.2), increasing it for synchronization purposes does not violate the initial limits since the segment maximum velocity and acceleration can only be lowered.

A.3.3 The case of orientations

Using unit quaternions to describe orientation in three dimensional space has several advantages over both Euler angles (simpler composition, no gimbal lock) and rotation matrices (more compact and numerically stable). Nevertheless, using unit quaternions over Euler angles has the disadvantage that their interpolation under velocity and acceleration (or higher derivative) constraints is not trivial. To overcome this, we propose a method to convert the quaternion interpolation problem in a form that allows the trajectory generator described above to be used. We define the orientation quaternion as:

$$\boldsymbol{q} = \mathcal{Q}\left(\mathbf{q}_v, q_w\right) \tag{A.22}$$

with $\mathbf{q}_v$ being the vector part and q_w the scalar part. The target orientation is denoted by $\boldsymbol{q}_r$. Then, we can compute an angular error vector $\Delta\boldsymbol{\theta}$ between $\boldsymbol{q}_r$ and $\boldsymbol{q}$ vector using:

$$\Delta\boldsymbol{q}(t) = \boldsymbol{q}_r(t)\overline{\boldsymbol{q}}(0) \tag{A.23}$$

$$\phi = \begin{cases} 2\cos^{-1}(\Delta q_w) & \text{if } \Delta q_w \geq 0 \\ 2\cos^{-1}(\Delta q_w) - 2\pi & \text{otherwise} \end{cases} \tag{A.24}$$

$$\Delta\boldsymbol{\theta}(t) = \phi\frac{\Delta\boldsymbol{q}_v(t)}{\|\Delta\boldsymbol{q}_v(t)\|}. \tag{A.25}$$

Using (A.24) gives $\phi \in]-\pi, \pi]$, allowing the rotation to be kept at its minimum (e.g., a rotation of $-\pi$ instead of $\frac{3\pi}{4}$). The trajectory generator described earlier can then be configured to output a trajectory going from $\mathbf{0}^3 = [0\ 0\ 0]^\top$ to $\Delta\boldsymbol{\theta}(t)$ under the desired velocity and acceleration constraints $\boldsymbol{\omega}_{max}$ and $\dot{\boldsymbol{\omega}}_{max}$:

$$\Delta\boldsymbol{\theta}^*(t) = \boldsymbol{\mathcal{P}}(t, \mathbf{c}') \tag{A.26}$$

$$\boldsymbol{\omega}^*(t) = \dot{\boldsymbol{\mathcal{P}}}(t, \mathbf{c}') \tag{A.27}$$

$$\dot{\boldsymbol{\omega}}^*(t) = \ddot{\boldsymbol{\mathcal{P}}}(t, \mathbf{c}'). \tag{A.28}$$

For pose tracking, the orientation quaternion $\boldsymbol{q}^*$ to be tracked can be computed after each interpolation as follow:

$$\boldsymbol{q}_\Delta(t) = \mathcal{Q}\left(\frac{\Delta\boldsymbol{\theta}^*(t)}{2}, 0\right) \tag{A.29}$$

$$\boldsymbol{q}^*(t) = e^{\boldsymbol{q}_\Delta(t)}\boldsymbol{q}(0) \tag{A.30}$$

A.3.4 Path tracking

When the robot is paused or slowed down due to α in (3.18) being lower than 1 while following a trajectory, the tracking error will grow, resulting in abrupt motions when it becomes unconstrained. In fact, the robot will try to reach the current desired position using the shortest path, which may be different from the initially planned one. To avoid such problems, a path following approach can be used. The goal here is to monitor the tracking error and to stop the trajectory generation when the error becomes too large. This translates to:

$$t_{TG}(0) = 0 \tag{A.31}$$

$$t_{TG}(t) = \begin{cases} t_{TG}(t - T_s) + T_s & \text{if } \|\mathbf{x}_r - \mathbf{x}\| < \varepsilon_{TG} \\ t_{TG}(t - T_s) & \text{otherwise} \end{cases} \tag{A.32}$$

$$\mathbf{x}^* = \mathcal{P}(\mathbf{c}', t_{TG}(t)). \tag{A.33}$$

Here, we can see that the time given to the trajectory generator is increased only when the tracking error stays below the threshold ε_{TG}. When the error becomes too large, t_{TG} remains constant so that the target pose will no longer be updated. The generation of the trajectory will be resumed only when the last computed target pose $\mathbf{x}^*$ is reached within ε_{TG}.

A.3.5 Benchmarks

Here, we only assess the good performance of algorithm 2. Indeed, the actual evaluation of the polynomial is very fast since it only requires a limited number of additions and multiplications and thus does not requires benchmarking. We defined three trajectories (one for each task space translation), each composed of six segments using three different waypoints. Details of the trajectories are presented in tables A.1 and A.2. It can be seen that the trajectory on the x axis always has null velocity and acceleration. Thus, its waypoints fall in the specific case described in A.3.1. The y axis trajectory is a bit more complicated since the velocity and acceleration at the second waypoint are non-null. Finally, the z axis trajectory is the most complicated one, since non-null velocity and acceleration are imposed to all the waypoints.

All the benchmarks have been run on a computer equipped with an Intel i7-6700HQ @ 2.6GHz running Linux 4.11. Figure A.3 gives the trajectory parameters computation time using Alg. 2 for each individual trajectory and for the three together, using different synchronization methods. The velocity and acceleration thresholds are set to $\varepsilon_v = 10^{-6}$ m/s and $\varepsilon_a = 10^{-6}$ m/s^2. Figure A.3 shows that the synchronization method has no noticeable impact on the computation time. Moreover, the average computation time in the worse

case scenario (generating x, y and z trajectories) is around $40\mu s$, which is suitable even for online use. Table A.3 gives the number of iterations needed to converge to the solution for $\varepsilon_{v,a} = 10^{-6}$ (high precision) and $\varepsilon_{v,a} = 10^{-3}$ (lower precision). It can be seen that for the x axis trajectory, the number of iterations per segment is 4, which is minimal since two iterations are needed to compute both $T_{min,v}$ and $T_{min,a}$ (one to update the minimum duration and a second to check that the solution has been found). Reducing the precision has a great impact on the convergence in non-ideal cases (y and z trajectories) since the number of iterations is approximatively divided by two in this example.

Table A.1 – Trajectories' waypoints.

Axis	P1			P2			P3		
	$\mathcal{P}$	$\dot{\mathcal{P}}$	$\ddot{\mathcal{P}}$	$\mathcal{P}$	$\dot{\mathcal{P}}$	$\ddot{\mathcal{P}}$	$\mathcal{P}$	$\dot{\mathcal{P}}$	$\ddot{\mathcal{P}}$
x	0	0	0	0.1	0	0	0.3	0	0
y	0	0	0	0.2	0.1	0.05	0.3	0	0
z	0	0.1	0.1	0.2	0.05	-0.05	0.1	-0.1	0.1

Table A.2 – Trajectories.

Axis	P1→P2		P2→P3		P3→P1		P1→P3		P3 → P2		P2→P1	
	$\dot{\mathcal{P}}_{max}$	$\ddot{\mathcal{P}}_{max}$	$\dot{\mathcal{P}}_{max}$	$\ddot{\mathcal{P}}_{max}$	$\dot{\mathcal{P}}_{max}$	$\ddot{\mathcal{P}}_{max}$	$\dot{\mathcal{P}}_{max}$	$\ddot{\mathcal{P}}_{max}$	$\dot{\mathcal{P}}_{max}$	$\ddot{\mathcal{P}}_{max}$	$\dot{\mathcal{P}}_{max}$	$\ddot{\mathcal{P}}_{max}$
x	0.05	0.01	0.1	0.02	0.05	0.01	0.1	0.02	0.05	0.01	0.05	0.01
y	0.15	0.1	0.25	0.2	0.15	0.1	0.25	0.2	0.15	0.2	0.25	0.2
z	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2

Table A.3 – Number of iterations to compute the trajectories.

	x	y	z
Total iterations ($\varepsilon_{v,a} = 10^{-6}$)	24	87	136
Total iterations ($\varepsilon_{v,a} = 10^{-3}$)	24	47	63
Iterations per segment (average, $\varepsilon_{v,a} = 10^{-6}$)	4	14.5	22.6
Iterations per segment (average, $\varepsilon_{v,a} = 10^{-3}$)	4	7.8	10.5

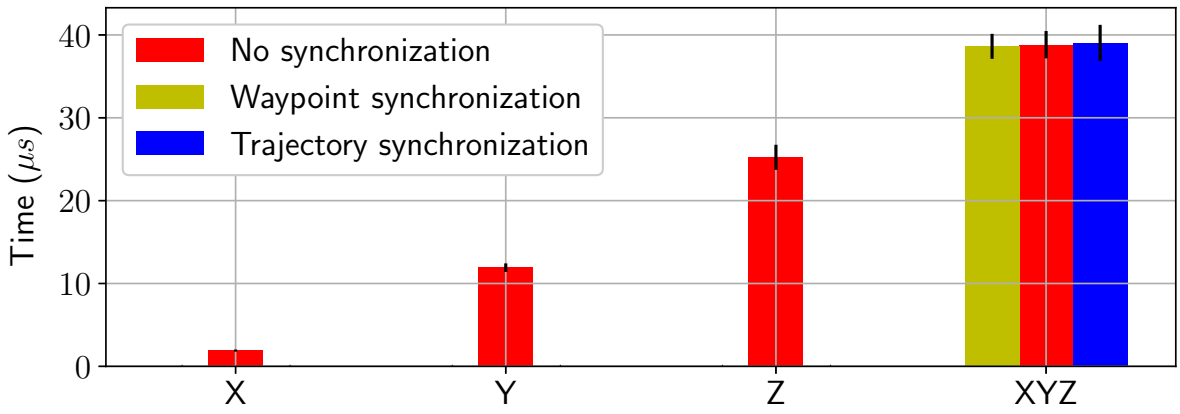


Figure A.3 – Trajectories' coefficients computation benchmark.

Bibliography

[1] B. Navarro, P. Kumar, A. Fonte, P. Fraisse, G. Poisson, and A. Cherubini. Active calibration of tactile sensors mounted on a robotic hand. In *IEEE/RSJ IROS Workshop on Multimodal sensor-based robot control for HRI and soft manipulation*, 2015.

[2] B. Navarro, A. Cherubini, A. Fonte, R. Passama, G. Poisson, and P. Fraisse. An ISO10218-compliant adaptive damping controller for safe physical human-robot interaction. In *IEEE International Conference on Robotics and Automation*, pages 3043–3048, May 2016.

[3] B. Navarro, A. Cherubini, A. Fonte, G. Poisson, and P. Fraisse. A framework for intuitive collaboration with a mobile manipulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3043–3048, May 2017.

[4] Wafa Tigra, Benjamin Navarro, Andrea Cherubini, Xavier Gorron, Anthony Gelis, Charles Fattal, David Guiraud, and Christine Azevedo-Coste. A novel EMG interface for individuals with tetraplegia to pilot robot hand grasping. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2016.

[5] R. Alami, et al. Safe and dependable physical human-robot interaction in anthropic domains: State of the art and challenges. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[6] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger. Requirements for safe robots: Measurements, analysis and new insights. *International Journal of Robotics Research*, 2009.

[7] A. De Luca and F. Flacco. Integrated control for pHRI: collision avoidance, detection, reaction and collaboration. In *IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*, 2012.

[8] Nathanaël Jarrassé, Themistoklis Charalambous, and Etienne Burdet. A framework to describe, analyze and generate interactive motor behaviors. *PLOS ONE*, 7(11):1–13, 11 2012.

[9] R. Stiefelhagen, C. Fugen, R. Gieselmann, H. Holzapfel, K. Nickel, and A. Waibel. Natural human-robot interaction using speech, head pose and gestures. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2422–2427 vol.3, Sept 2004.

[10] B. Burger, F. Lerasle, I. Ferrane, and A. Clodic. Mutual assistance between speech and vision for human-robot interaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4011–4016, Sept 2008.

[11] Eri Sato, Toru Yamaguchi, and Fumio Harashima. Natural interface using pointing behavior for human-robot gestural interaction. *IEEE Transactions on Industrial Electronics*, 54(2):1105–1112, apr 2007.

[12] Charles Rich, Brett Ponsler, Aaron Holroyd, and Candace L. Sidner. Recognizing engagement in human-robot interaction. In *5th ACM/IEEE International Conference on Human-Robot Interaction*. IEEE, mar 2010.

[13] Mohammad Rabiei and Alessandro Gasparetto. System and method for recognizing human emotion state based on analysis of speech and facial feature extraction; applications to human-robot interaction. In *4th International Conference on Robotics and Mechatronics*. IEEE, oct 2016.

[14] Ntombikayise Banda, Andries Engelbrecht, and Peter Robinson. Feature reduction for dimensional emotion recognition in human-robot interaction. In *IEEE Symposium Series on Computational Intelligence*. IEEE, dec 2015.

[15] A. Bruce, I. Nourbakhsh, and R. Simmons. The role of expressiveness and attention in human-robot interaction. In *Proceedings IEEE International Conference on Robotics and Automation*. IEEE, 2002.

[16] T. Fincannon, L.E. Barnes, R.R. Murphy, and D.L. Riddle. Evidence of the need for social intelligence in rescue robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2004.

[17] Michiel Joosse, Manja Lohse, Jorge Gallego Perez, and Vanessa Evers. What you do is who you are: The role of task context in perceived social robot personality. In *IEEE International Conference on Robotics and Automation*. IEEE, may 2013.

[18] Lee M. Ssanderson, James W. Collins, and James D. McGlothlin. Robot-related fatality involving a u.s. manufacturing plant employee: Case report and recommendations. *Journal of Occupational Accidents*, 8(1-2):13–23, jun 1986.

[19] J. Krüger, T.K. Lien, and A. Verl. Cooperation of human and machines in assembly lines. *CIRP Annals - Manufacturing Technology*, 58(2):628 – 646, 2009.

[20] Wansoo Kim, Jinoh Lee, Luka Peternel, Nikos Tsagarakis, and Arash Ajoudani. Anticipatory robot assistance for the prevention of human static joint overloading in human–robot collaboration. *IEEE Robotics and Automation Letters*, 3(1):68–75, jan 2018.

[21] Luka Peternel, Nikos Tsagarakis, Darwin Caldwell, and Arash Ajoudani. Adaptation of robot physical behaviour to human fatigue in human-robot co-manipulation. In *IEEE-RAS 16th International Conference on Humanoid Robots*. IEEE, nov 2016.

[22] Sami Haddadin, Simon Haddadin, Augusto Khoury, Tim Rokahr, Sven Parusel, Rainer Burgkart, Antonio Bicchi, and Alin Albu-Schaffer. A truly safely moving robot has to know what injury it may cause. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, oct 2012.

[23] Gerd Hirzinger and Alin Albu-Schaeffer. Light-weight robots. *Scholarpedia*, 3(4):3889, 2008.

[24] M. Zinn, O. Khatib, B. Roth, and J.K. Salisbury. Playing it safe. *IEEE Robotics and Automation Magazine*, 11(2):12–21, 2004.

[25] A. Albu-Schaffer et al. Soft robotics. *IEEE Robotics and Automation Magazine*, 15(3):20–30, 2008.

[26] R. Schiavi, G. Grioli, S. Sen, and A. Bicchi. VSA-II: A novel prototype of variable stiffness actuator for safe and performing robots interacting with humans. In *IEEE International Conference on Robotics and Automation*, 2008.

[27] Antonio Bicchi, Giovanni Tonietti, Michele Bavaro, and Marco Piccigallo. Variable stiffness actuators for fast and safe motion control. In *Springer Tracts in Advanced Robotics*, pages 527–536. Springer Berlin Heidelberg, 2005.

[28] J.C. Dean and A.D. Kuo. Elastic coupling of limb joints enables faster bipedal walking. *Journal of The Royal Society Interface*, 6(35):561–573, oct 2008.

[29] Sami Haddadin, Tim Laue, Udo Frese, Sebastian Wolf, Alin Albu-Schäffer, and Gerd Hirzinger. Kick it with elasticity: Safety and performance in human–robot soccer. *Robotics and Autonomous Systems*, 57(8):761–775, jul 2009.

[30] Raphaël Furnémont, Glenn Mathijssen, Tom Verstraten, Dirk Lefeber, and Bram Vanderborght. Bi-directional series-parallel elastic actuator and overlap of the actuation layers. *Bioinspiration & Biomimetics*, 11(1):016005, jan 2016.

[31] ISO 10218-1:2011 Robot for industrial environments - Safety requirements - Part 1 : Robot. Technical report, International Organization for Standardization, Geneva, Switzerland, 2006.

[32] ISO/TS 15066:2016 robots and robotic devices – collaborative robots. Technical report, International Organization for Standardization, Geneva, Switzerland, 2016.

[33] Martin J. Rosenstrauch and Jorg Kruger. Safe human-robot-collaboration-introduction and experiment using ISO/TS 15066. In *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*. IEEE, apr 2017.

[34] Robotiq. Collaborative robot ebook - http://blog.robotiq.com.

[35] Antoine de Rengerve, Julien Hirel, Pierre Andry, Mathias Quoy, and Philippe Gaussier. On-line learning and planning in a pick-and-place task demonstrated through body manipulation. In *IEEE International Conference on Development and Learning*. IEEE, aug 2011.

[36] Luka Peternel, Nikos Tsagarakis, and Arash Ajoudani. Towards multi-modal intention interfaces for human-robot co-manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, oct 2016.

[37] Jeremy A. Marvel and Rick Norcross. Implementing speed and separation monitoring in collaborative robot workcells. *Robotics and Computer-Integrated Manufacturing*, 44:144–155, apr 2017.

[38] A Vick, D Surdilovic, and J. Krüger. Safe physical human-robot interaction with industrial dual-arm robots. In *9th IEEE Workshop on Robot Motion and Control*, pages 264–269, 2013.

[39] H. G. Tanner, S. G. Loizou, and K. J. Kyriakopoulos. Nonholonomic navigation and control of cooperating mobile manipulators. *IEEE Transactions on Robotics and Automation*, 19(1):53–64, Feb 2003.

[40] A. De Luca, G. Oriolo, and P. R. Giordano. Kinematic modeling and redundancy resolution for nonholonomic mobile manipulators. In *IEEE International Conference on Robotics and Automation*, pages 1867–1873, May 2006.

[41] A. De Luca, G. Oriolo, and P. Robuffo Giordano. Kinematic control of nonholonomic mobile manipulators in the presence of steering wheels. In *IEEE International Conference on Robotics and Automation*, pages 1792–1798, May 2010.

[42] J. Vaillant, K. Bouyarmane, and A. Kheddar. Multi-character physical and behavioral interactions controller. *IEEE Transactions on Visualization and Computer Graphics*, PP(99):1–1, 2016.

[43] D. J. Agravante, A. Sherikov, P. B. Wieber, A. Cherubini, and A. Kheddar. Walking pattern generators designed for physical collaboration. In *IEEE International Conference on Robotics and Automation*, pages 1573–1578, May 2016.

[44] H. Adachi, N. Koyachi, T. Arai, and K. I. Nishimura. Control of a manipulator mounted on a quadruped. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 883–888 vol.2, Nov 1996.

[45] A. E. Jimenez-Cano, J. Martin, G. Heredia, A. Ollero, and R. Cano. Control of an aerial robot with multi-link arm for assembly tasks. In *IEEE International Conference on Robotics and Automation*, pages 4916–4921, May 2013.

[46] J. U. Álvarez-Muñoz, Nicolas Marchand, Fermi Guerrero-Castellanos, Sylvain Durand, and A. E. Lopez-Luna. Improving control of quadrotors carrying a manipulator arm. In *XVI Congreso Latinoamericano de Control Automático*, page 6, -, Mexico, October 2014.

[47] K. S. Kim, A. S. Kwok, G. C. Thomas, and L. Sentis. Fully omnidirectional compliance in mobile robots via drive-torque sensor feedback. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4757–4763, Sept 2014.

[48] Y. Jia, H. Wang, P. Stürmer, and N. Xi. Human/robot interaction for human support system by using a mobile manipulator. In *IEEE International Conference on Robotics and Biomimetics*, pages 190–195, Dec 2010.

[49] Q. Leboutet, E. Dean-León, and G. Cheng. Tactile-based compliance with hierarchical force propagation for omnidirectional mobile manipulators. In *IEEE-RAS 16th International Conference on Humanoid Robots*, pages 926–931, Nov 2016.

[50] Fei Chao, Zhengshuai Wang, Changjing Shang, Qinggang Meng, Min Jiang, Changle Zhou, and Qiang Shen. A developmental approach to robotic pointing via human–robot interaction. *Information Sciences*, 283:288–303, nov 2014.

[51] Jim Mainprice, Mamoun Gharbi, Thierry Simeon, and Rachid Alami. Sharing effort in planning human-robot handover tasks. In *IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, sep 2012.

[52] Huajin Tang, Boon Hwa Tan, and Rui Yan. Robot-to-human handover with obstacle avoidance via continuous time recurrent neural network. In *IEEE Congress on Evolutionary Computation*. IEEE, jul 2016.

[53] Dimitrios Papageorgiou and Zoe Doulgeri. A kinematic controller for human-robot handshaking using internal motion adaptation. In *IEEE International Conference on Robotics and Automation*. IEEE, may 2015.

[54] M. Jindai and T. Watanabe. A handshake robot system based on a shake-motion leading model. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, sep 2008.

[55] O. Khatib. Inertial properties in robotic manipulation: An object-level framework. *The International Journal of Robotics Research*, 14(1):19–36, feb 1995.

[56] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008.

[57] Łukasz Wolinski and Paweł Malczyk. Dynamic modeling and analysis of a lightweight robotic manipulator in joint space. *Archive of Mechanical Engineering*, 62(2), jan 2015.

[58] N. Hogan. Impedance control: An approach to manipulation: Part II-Implementation. *Journal of Dynamic Systems, Measurement, and Control*, 107(1):8–16, 1985.

[59] S. Chiaverini, O. Egeland, and R. K. Kanestrom. Achieving user-defined accuracy with damped least-squares inverse kinematics. In *5th International Conference Advanced Robotics*, pages 672–677 vol.1, June 1991.

[60] F. Almeida, A. Lopes, and P. Abreu. Force-impedance control: A new control strategy of robotic manipulators. In *Recent Advances in Mechatronics, Springer, Singapore*, pages 126–137, 1999.

[61] S. Jung, T.C. Hsia, and R.G. Bonitz. Force tracking impedance control of robot manipulators under unknown environment. *IEEE Transactions on Control Systems Technology*, 12(3):474–483, May 2004.

[62] G. Muscio, F. Pierri, and J. Trinkle. A hand/arm controller that simultaneously regulates internal grasp forces and the impedance of contacts with the environment. In *IEEE International Conference on Robotics and Automation*, pages 895–900, May 2014.

[63] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, mar 1986.

[64] T. Yoshikawa. Manipulability and redundancy control of robotic mechanisms. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1004–1009, Mar 1985.

[65] Bong-Huan Jun, Pan-Mook Lee, and Jihong Lee. Manipulability analysis of underwater robotic arms on rov and application to task-oriented joint configuration. In *MTTS/IEEE TECHNO-OCEAN*, volume 3, pages 1548–1553 Vol.3, Nov 2004.

[66] Adrien Escande, Sylvain Miossec, Mehdi Benallegue, and Abderrahmane Kheddar. A strictly convex hull for computing proximity distances with continuous gradients. *IEEE Transactions on Robotics*, 30(3):666–678, June 2014.

[67] V. Ciobanu, D. Popescu, and A. Petrescu. Point of contact location and normal force estimation using biomimetical tactile sensors. In *8th International Conference on Complex Intelligent and Software Intensive Systems*, pages 373–378, 2014.

[68] Torsten Kroger. Opening the door to new sensor-based robot applications - the reflexxes motion libraries. In *IEEE International Conference on Robotics and Automation*. IEEE, may 2011.

[69] M. A. Jenkins and J. F. Traub. A three-stage algorithm for real polynomials using quadratic iteration. *SIAM Journal on Numerical Analysis*, 7(4):545–566, 1970.

[Benjamin NAVARRO]

[Solutions pour une collaboration humain-robot sûre]

L'interaction physique humain-robot devient cruciale dans un nombre grandissant d'applications, de la santé à l'industrie, impliquant différents types de machines robotiques : manipulateurs sériels, plates-formes mobiles, manipulateurs mobiles ou humanoïdes. Dans les phases d'interactions, le robot ne doit assurément causer aucune blessure à l'opérateur humain. C'est dans ce but que des efforts de standardisation ont étés réalisés (ISO 10218, ISO/TS 15066) afin de fournir des guides lors de développements de tâches collaboratives entre un humain et un robot. Cependant, à l'heure actuelle, aucune solution globalement acceptée n'est en passe de répondre à ces problématiques. Les solutions disponibles ne se concentrent que sur un nombre limité de fonctionnalités et ne peuvent pas encore être utilisées seules pour répondre à la grande variété de scénarios collaboratifs. Dans cette thèse, nous proposons un «framework »de contrôle sûr en amortissement à deux niveaux, conjointement à son implémentation libre sur la plateforme ouverte OpenPHRI. Celui-ci permet de décrire une grande variété de tâches tout en offrant la capacité de respecter plusieurs critères de sécurité. Ce respect de la sécurité impose par ailleurs une maîtrise de différents niveaux matériels. Nous nous intéresserons ainsi au contrôle bas niveau en couple de robots présentant des frottements secs non modélisables, à l'estimation des forces et couples externes, mais aussi à des extensions de notre framework aux manipulateurs mobiles, grâce à une gestion de la redondance inspirée du comportement humain. Nous nous intéresserons enfin aux mains robotiques équipées de capteurs tactiles permettant l'estimation des forces de contact et la détection des intentions humaines simples.

Mots clés : robotique, interaction, collaboration, contrôle, sécurité

[Solutions for safe human-robot collaboration]

Physical human-robot interaction is becoming crucial in an increasing number of applications, from health care to industrial processes, with many types of robots: serial manipulators, mobile platforms, mobile manipulators or humanoids. During the interactions phases, the robot must certainly not cause any harm to the human operator. To this end, some standardization efforts have been realized to provide guidelines (ISO 10218, ISO/TS 15066) when developing human-robot collaboration tasks but no generally accepted solution have been proposed so far to fulfill these requirements. Present day solutions focus on a limited set of features and so can't be used, alone, to accommodate with a large variety of scenarios. In this thesis, we propose a two-layer safe damping control framework, alongside with its open source implementation OpenPHRI, that allows to describe a wide range of tasks while being able to respect multiple safety constraints. This respect for safety also imposes a good knowledge of the devices behavior. So, we also take a look at low level torque control in the presence of non-modelisable static frictions, the estimation of external torques and forces, as well as extensions to our framework to mobile manipulators, using a novel redundancy solution that mimics a human behavior. We finally study robotic hands equipped with tactile sensors allowing the extraction of contact forces and the detection of basic human intentions.

Keywords : robotics, interaction, collaboration, control, safety

Centre Val de Loire UNIVERSITÉ Spirit of the Loire Valley

PRISME, Université d'Orléans, Château de la Source, Avenue du Parc Floral, BP 6749, 45067 Orléans Cédex 2

UNIVERSITE D'ORLEANS