An energetic approach to safety in robotic manipulation

Lucas Joseph







THÈSE

présentée pour obtenir

le titre de docteur délivré par Sorbonne Université

École doctorale: Sciences Mécaniques, Acoustique, Électronique et Robotique de Paris

par

Lucas JOSEPH

An energetic approach to safety in robotic manipulation

Soutenue publiquement le 7 Décembre 2018 devant le jury composé de:

Pascal MORIN	Professeur des Universités à Sorbonne Université	Président du Jury
Andrea CHERUBINI	Maître de Conférences au LIRMM - HDR	Rapporteur
Bernard BAYLE	Professeur des Université à Télécom Physique Strasbourg	Rapporteur
Guillaume MOREL	Professeur des Universités à Sorbonne Université	Directeur de thèse
Vincent PADOIS	Maître de Conférences à Sorbonne Université - HDR	Encadrant
Serge MULLER	Chief Scientist Women's Health, GE Healthcare	Encadrant

Institut des Systèmes Intelligents et de Robotique (ISIR)
Pyramide Tour 55, 4 place Jussieu
UMR CNRS 7222, Paris, France

Abstract

Collaborative robots offer new possibilities to use robots in workspaces shared with humans. These robots can interact with their environment and assist human beings in their task in a safer way compared to standard industrial ones. They are required to be fast, precise, and efficient during the accomplishment of their tasks. However, their strength can make them dangerous tools around people. Therefore, to ensure safety they are often used in a sub-optimal way.

The aim of this work is to ensure the safety of a human interacting with a robot performing a set of tasks. It is more specifically focused on the case of undesired contact with the robot. An undesired contact can either come from an impact between the robot and an obstacle or from the robot pushing the obstacle against a fixed object. This work shows that in both cases the dangerousness of the robot can be linked to the robot variation of kinetic energy. These robots are also submitted to a set of constraints coming either from their intrinsic design or from the environment. When a human enters in the robot workspace, the robot environment becomes partially unknown. The robot should be able to adapt to its environment and reactively compute a safe control solution satisfying its constraints while performing its tasks in an optimal way.

To that aim, this work formulates the control problem as a constrained optimization one and solves it using Linear Quadratic Programming. The robot variation of kinetic energy is constrained within the quadratic programming problem. This constraint prevents the robot from reaching a dangerous amount of kinetic energy. It also indirectly prevents the robot from exerting too much forces when in contact with an obstacle. This work also shows that it is possible to use the robot redundancy to minimize its perceived mass in the direction of an obstacle. This indirectly reduces its kinetic energy for a given task.

The proposed controller is implemented on a 7 dof serial robotic manipulator. This work, realised in collaboration with General Electric Healthcare, features several experiments with this controller in the context of medical imaging. Using external sensors it is shown that it is possible to realise tasks in an optimal way while limiting the robot dangerousness in case of an undesired contact.

Keywords: Human/Robot Interaction, Safety, Linear Quadratic Programming, Redundant Robots, Kinetic Energy, Undesired Contact

Contents

List of Figures

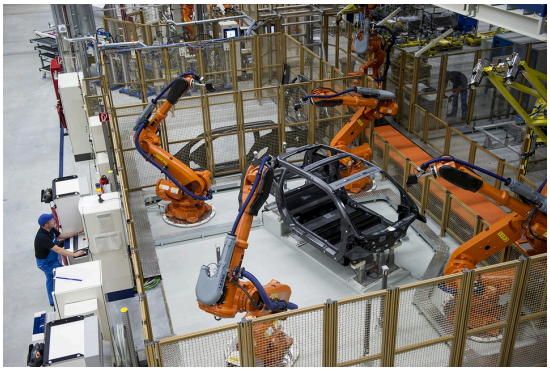
Chapter 1

Introduction and applicative context

Standard industrial robots are used to realise specific tasks requiring high repeatability and good accuracy. Untiredness and payload capacity are examples of the many advantages that such systems exhibit as compared to humans. The most standard and efficient way to obtain precise positioning and trajectory tracking is to implement stiff position control. Indeed, the high control gains used in such an approach provide robustness to disturbances. When a stiff robot enters in contact with its environment, most of its mechanical energy is transmitted to the environment. This can lead to potential danger for a human operator working nearby and not anticipating the movement of the robot. In 1987, a study by B. Jiang and C. Gainer analysed the causes of robot injuries in several developed countries [Jiang and Gainer, 1987]. The results of the study showed that most accidents could have been avoided if proper safety measures had been enforced during the robot implementation.

A common safety measure to prevent contact is to impose a strict separation between the robot and the human. This separation can be done by putting the robot in a fence or using any sensor able to detect the intrusion of humans in the robot workspace. Figure 1.1A depicts a typical situation where robots working in an assembly line are put in a cage, preventing any contact with humans. The opening of the cage triggers a complete shut-down of the robot, and power is restored in the actuators only when the robot workspace is freed or when the cage is closed.

When a robot is placed in a cage, its surrounding environment can be perfectly determined. Hence, it is possible to compute off-line trajectories that achieve tasks optimally while avoiding obstacles around the robot [Hart et al., 1968]. This problem



Source: BMW Launches i3 Electric Car Production. Getty images

(A)

Source: Day on the Job: Operating Room. the Official United States Air Force Website

(B)

FIGURE 1.1: (A) Robots in an assembly line working in a workspace delimited by cages. The environment is perfectly known and every motion can be determined off-line. (B) An imaging robot evolving in an operating room. Motions are restrained and humans are constantly moving around the robot. If the robot has to update its trajectories on-line it must account for the environment.

has seen many contributions over the years [Lavalle, 1998], [Fox et al., 1997] and is commonly used in factories.

This separation is an efficient way to ensure safety as it prevents any possible contact with the robot. However, there are applications which can benefit from a close collaboration between a human and a robot. Indeed, on one hand, robots physical capacities, *i.e.* their precision, untiredness and capacity to lift heavy loads are far superior to the ones of humans. On the other hand, humans are able to adapt to unknown situations and realise abstract tasks. Extensive work in the field of collaborative robotics has been conducted to combine the strengths of robots to the ones of humans. Dedicated actuation technologies [Albu-Schäffer et al., 2007], [Bicchi and Tonietti, 2004] and state-of-the-art control solutions [De Luca et al., 2006] are employed to allow interactions with these robots. However, opening the robot workspace to humans and having an environment constantly evolving brings new control challenges, especially in terms of safety.

By opening the workspace, the environment surrounding the robot becomes partially unknown and off-line trajectory planning techniques can no longer be used alone to ensure safety. An example of such a situation is depicted in Figure 1.1B where an imaging robot is evolving in an operating room. The robot evolves in a cluttered environment with moving humans and medical equipment and should avoid entering in contact with them. This implicitly limits the robot motions to account for the surrounding environment. Removing the separation between robots and humans requires to reactively[1] consider

[1] In this document, a robot reactivity refers to is ability to respond in real-time to a change in the environment.

the environment surrounding the robot to decide future motions. This can be to avoid obstacles, such as humans or objects or to adapt its trajectories to new inputs in the scene (new goal, new user input, *etc.*). However, the updated desired motions should not push the robot towards a state that it cannot reach. It should also avoid pushing the robot towards a state from which it will inevitably end-up in such an unreachable state [Rubrecht et al., 2012]. While this statement can seem evident, in practice, this is not easy to implement. It requires to reactively account for the robot dynamics and for its actuation limits when computing its future motions. These limits can be regarded as constraints to be satisfied at all time. Constraints are expressed to avoid pushing the robot towards a non-achievable state or to forbid some robotic behaviours.

Overall, when controlling a robot in an open environment, three key points should be considered:

Reactivity: A robot should reactively account for new information provided by the environment to adapt its behaviour.

Performance: A robot should optimally achieve its tasks.

Constraint compliance: A robot should always be able to satisfy any set of explicit constraints.

Safety: A robot should not be dangerous towards its environment.

Table 1.1 details different robot contexts and analyses their characteristics in terms of performance, reactivity, constraint compliance and safety. Three main contexts arise: robots operating in a workspace that is physically separated from the human, robots being hand guided and robots operating autonomously in a workspace shared with a human with possible contact (desired or not). The robot performances are divided into the following three categories:

- Tracking, which is the ability to correctly follow a desired trajectory.
- Transparency, which is the ability not to resist a human voluntarily moving the robot.
- Flexibility, which is the simplicity to reuse a robot to perform a different task in a different context.

As detailed previously, for robots working in a separated workspace, safety is ensured by fences placed around the robot. The environment is supposed to be perfectly known

Characteristics / Control context	Reactivity Necessity to adapt on-line to its environment?	Constraint compliance Is the robot satisfying its constraints?	Performance			
			Tracking	Transparency	Flexibility	
Separated workspace	$--$ Fixed environment Pre-planned trajectory	$++$ Validated off-line	$++$ Optimal	$--$ Stiffly actuated	$-$	
Hand guided	$\varnothing$ Intrinsic	$+$ Only position limits. Slow motions far from the dynamic limits	$\varnothing$ Follows human motions	$++$ Intrinsic	$+$	Er
Shared workspace	$++$	**?** Constraints should be verified reactively	**?** As good as possible	$+$	$+$	

$+$: Condition satisfied with some restrictions $++$: Condition satisfied

$-$: Condition satisfied with some difficulties $--$: Condition not satisfie

? : No concrete answer yet $\varnothing$: Not concerned

TABLE 1.1: Different robot contexts and a quantitative analyse of their characteristics in terms of reactivity, cons and safety.

so that no reactivity is required. As such, constraint compliance and optimal trajectory can be handled off-line. Stiff robots ensure a good trajectory tracking at the cost of transparency.

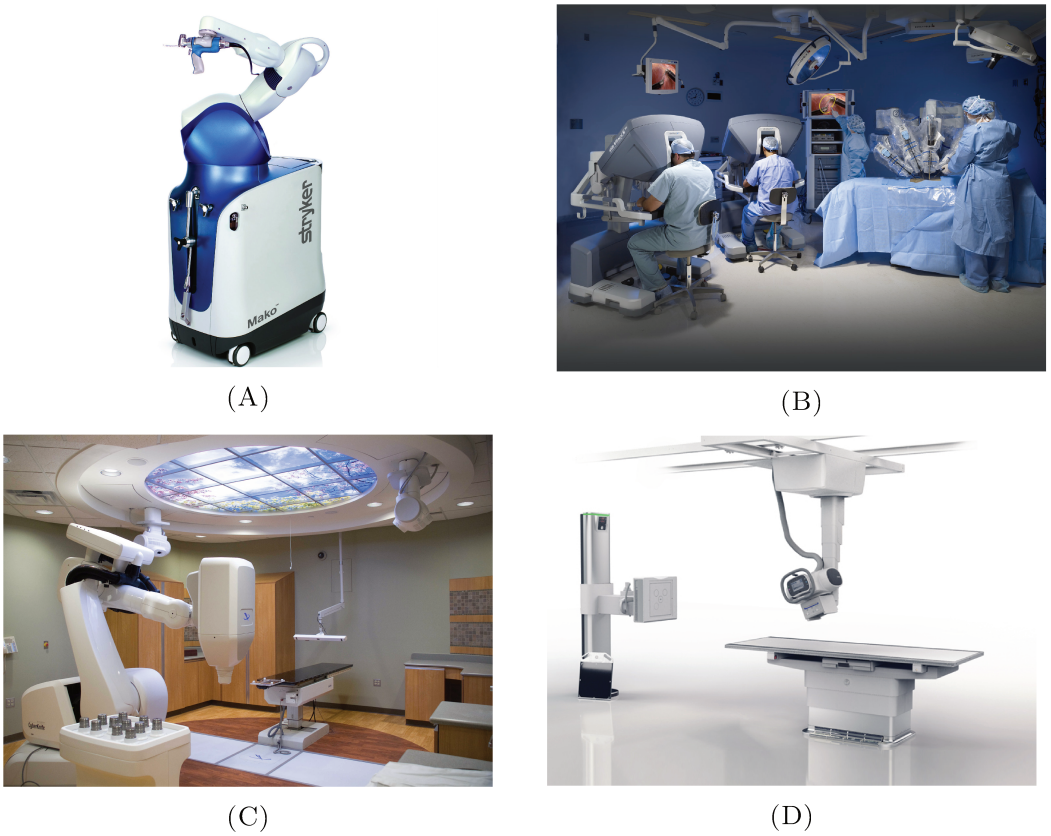
Hand guided robots are physically guided by an operator. They are usually grabbed by their end-effector and follow the motions imposed by the operator in Cartesian space [Poquet et al., 2015]. When a robot has more degrees of freedom than the ones required to perform the task, they are called redundant. Given a desired Cartesian pose, it means that there exists an infinite number of combination of articular configurations leading to a specific pose of the robot. If these inner motions are not monitored, undesired contact with the environment can occur. That is why hand guided robots are intrinsically safe when considering the motion of their end-effector as they only follow the human motions and yet, they can be dangerous because of their inner motions. In the hand guided context, the constraint compliance problem is alleviated by the fact that a robot moved by hand performs slow motions that do not go beyond the limited dynamics of the robot. In practice only the position limits of each joint are monitored.

When a robot can move autonomously in a workspace shared with human operators, it is required to ensure the safety of people around it. Ensuring safety includes avoiding collisions but also ensuring that any kind of contact with a human is not dangerous, regardless of the tasks to achieve. It thus requires to constantly adapt the robot behaviour. Table 1.1 shows that having a robot operating in such a shared workspace requires to optimally realise its tasks, and to re-plan on-line according to its evolving environment, all while satisfying its constraints. The resolution of such problem is still an active topic.

This work is focused on the expression of a generic control architecture to realise tasks defined on-line while satisfying the robot constraints and ensuring safety. This control solution is applied for the specific context of X-ray medical imaging. This chapter first presents different medical applications and relates them to the control contexts exposed in 1.1. It then details the applicative context of this work and presents solutions to ensure safety that can be found in the literature.

1.1 Robots in medical applications

Robots used in medical applications always share their workspace with medical equipment, care takers and, at least, a patient. This section presents different applications where robots are used to improve a medical procedure. It analyses these applications from the perspective of the typology given in table 1.1.



(A) (B)

(C) (D)

FIGURE 1.2: (A) The Mako robot, a co-manipulated robot for surgical interventions. (B) The Da Vinci telemanipulated robot. (C) The Cyberknife, a medical device dedicated to radiotherapy. (D) The GEHC Discovery XR656 that can realise Digital Tomosynthesis procedures.

Hand guided robots are often used in surgical robotics. They hold tools and follow the surgeon motions during the procedure. The Surgicobot [Riwan et al., 2011] and the Mako robot[2] presented in Figure 1.2A are two co-manipulated robots performing bone milling. They create virtual walls to prevent the surgeon from entering in specific areas with its milling tool [Jakopec et al., 2003]. The Da Vinci robot[3], depicted in Figure 1.2B, is a tele-manipulated robot used for mini-invasive surgery. A surgeon operates a multi-arm robot from a remote console. The robot is able to perform precise motions suited for operations that require accurate positioning of a tool. Although these robots perform slow motions at the end-effector level, inner motions of the robotic arms can be problematic. Self-collision between two arms can happen but also collisions with the medical staff. In practice, these motions can be bothersome and should be dealt with. However, this is out of the scope of this work.

The Cyberknife[4] is a robotised system consisting in an X-ray source mounted on

[2] www.stryker.com/us/en/joint-replacement/systems/mako-robotic-arm-assisted-surgery.html

[3] www.intuitivesurgical.com/products/davinci_surgical_system/

[4] www.accuray.com/cyberknife/

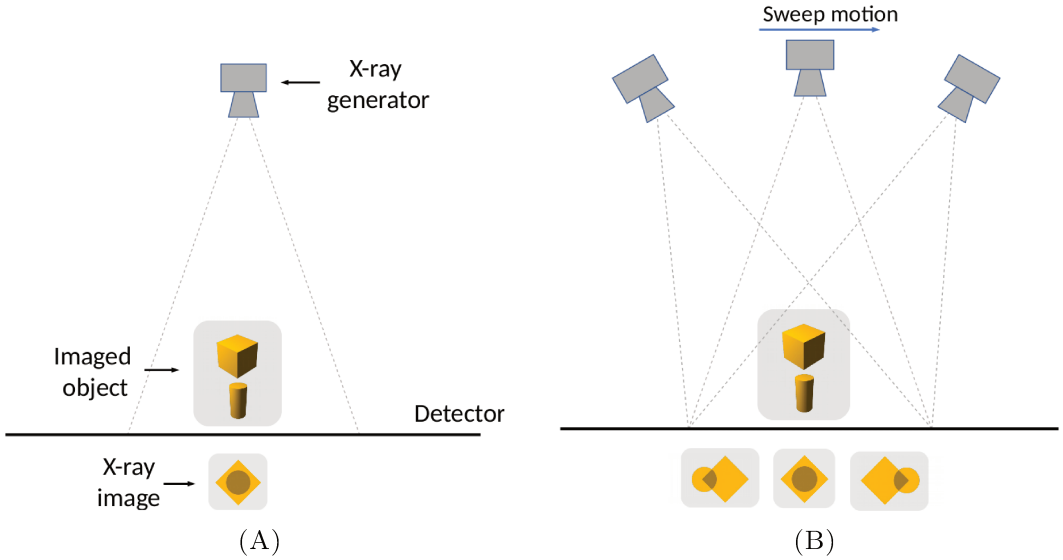


FIGURE 1.3: (A) Scheme of an X-ray scanner and the resulting X-ray image. It is difficult to discriminate overlapping tissues. (B) Representation of Digital Tomosynthesis procedure.

an industrial robotic manipulator and moving around a table on which a patient is located (Figure 1.2C). The robot is used for radiotherapy, a procedure dedicated to the removal of cancerous cells via the emission of X-rays. This procedure irradiates at a high dose cancerous cells while sparing the surrounding healthy ones. To this end, the robot is positioned at different places around the patient to shoot X-rays. All the X-ray beams intersect at the same concurrent point allowing sending high doses of radiation to cancerous cells while avoiding irradiating healthy ones. Radiation therapy requires to precisely position the robot around the patient and hold that position during the radiation. Fiducial gold markers detected by X-ray images associated with infra-red cameras can be used to detect the small motions induced by human breathing [Schweikard et al., 2004]. The radiotherapy X-rays are only shot when the tumour is located in a predefined window. Preoperative data are acquired to locate the tumour and the patient is fixed on the table to restrain any motion. The robot trajectories can thus be computed off-line assuming that the environment is invariant. This allows having robots working in a shared workspace using control strategies similar to the one of robots working in a separated workspace.

X-ray medical imaging procedures also use robots to hold an X-ray source. X-rays consist in an electromagnetic radiation possessing penetration properties that can be used to image the inside of a body. When X-rays cross an anatomical structure, a part of the radiation is absorbed by the tissues and the rest goes through the structure. As depicted in Figure 1.3A, an X-ray detector gathers the remaining light and gives a 2D image of the level of absorption of the different objects crossed by the X-rays. Standard X-ray imaging cannot be used for tissues located deeply in the body, such as brain

cells. Indeed, when tissues overlap, their superposition makes it difficult to differentiate different elements in the resulting X-ray image. To overcome this drawback, some procedures propose to take several images at different positions and use image reconstruction algorithms to obtain a 3D representation of the observed object as depicted in Figure 1.3B. Digital Tomosynthesis (DT) is a procedure relying on such a process. DT uses a robotic arm to move the X-ray generator around the patient. The procedure requires a precise knowledge of the X-ray source position for the reconstruction algorithm. To avoid reconstruction artefacts induced by human motion, the procedure should be as fast as possible. Considering that the environment is still, the X-ray source follows a pre-planned trajectory satisfying the systems constraints. In such a situation, the robot is stiffly actuated, and safety is ensured through collision detection sensors. The GEHC Discovery XR656[5] in Figure 1.2D is a commercially available example of a DT system.

The robots presented in this section are representative of the medical devices commercially available. They are either hand guided robot performing precise surgical procedures, or robotic manipulators carrying heavy tools around a patient. In the latter case, it can be noted that although these robots are used in a context where they share their environment with humans, it is assumed that the environment is fixed and their trajectories are computed off-line. This eases the control problem, particularly in terms of constraint compliance but prevents any reactive behaviour.

1.2 Towards autonomous robots in a shared workspace for medical applications

A robot that can adapt its trajectory on-line according to a higher level decision scheme is of great interest. For instance, Digital Tomosynthesis is an application that could be improved by redefining the position of the X-ray source reactively. It has been shown that image quality could be improved by adapting the trajectory of the X-ray source according to images taken previously [Haque et al., 2013] or [Stayman and Siewerdsen, 2013]. Defining new X-ray source positions on-line requires accounting dynamically for the robot constraints such as its actuation limits but also the constraints related to the patient safety. Indeed, the robot is evolving in an open environment with a patient next to it, and undesired contacts between the two could happen.

This practical example features problems that apply to any autonomous robot sharing its workspace with a human. The ability to define trajectories and generate motions suitable to ensure on-line the safety of the robot and its environment are prerequisites

[5] www.gehealthcare.com/en/products/categories/radiography/fixed_rad_systems/discovery_xr656_plus

to the use of robots in a shared workspace. To define safe robot motions, one must first determine the potential hazards that can occur when a robot is performing its tasks. This is the topic of the next section.

1.3 Characterization of a robot dangerousness

This section presents different indicators of safety that are used in the literature and shows how they can be linked to the robot control variables. A special focus is put on the dangerousness of a robot interacting with a human. A qualitative measure of human injury is first presented and is used in a second part to characterise quantitative indicators of robot dangerousness.

1.3.1 The Abbreviated Injury Scale

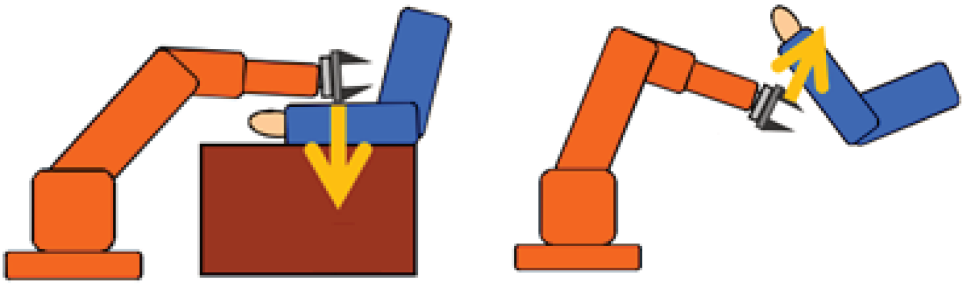
The human pain being highly subjective, it cannot be used to characterise a robot dangerousness. On the other hand, the results of an injury (bruises, open wounds, ...) can be observed on the human body. The Abbreviated Injury Scale (AIS) is a coding system that classifies and describes the severity of an injury based on anatomical observations. It defines the location of the injury (head, neck, thorax, ...), the type of anatomic structure concerned (vessel, nerves, organs, ...) and the level of severity of the injury on a scale between 1 and 6. Each level is associated with a corresponding probability of death.

AIS	Severity	Type of injury	Death prob (%)
1	Minor	Superficial	0
2	Moderate	Recoverable	1-2
3	Serious	Possibly recoverable	8-10
4	Severe	Not fully recoverable without care	5-50
5	Critical	Not fully recoverable with care	5-50
6	Maximum	Fatal	100

TABLE 1.2: AIS Code

Table 1.3.1 depicts the code representing the degree of severity of an injury. An AIS of 1 corresponds to a minor injury with a probability of death of 0% and will result in a superficial injury such as a bruise. An AIS of 6 corresponds to a fatal injury leading to death.

The AIS scale was first introduced in the automotive field to describe the dangerousness of a car crash [States, 1969]. It has been used in robotics to characterize the dangerousness of a collision between a robot and a human [Haddadin et al., 2008b], [Oberer et al., 2006].



Images extracted from [ISO/TS-15066, 2016]

(A) (B)

FIGURE 1.4: (A) Quasi-static contact: the robot crushes a human body part against a fixed object. (B) Transient contact: the robot enters in contact with a human body part for a short amount of time. The body part can recoil from the contact.

1.3.2 Definition of safety indicators

During an unwanted physical interaction between a robot and a human operator, contact can be divided into two types:

- Quasi-static contact: A constrained contact where the operator body is clamped between a moving part of the robot and another fixed or moving part of the environment as depicted in Figure 1.4A. In this situation, the static contact between the robot on the body part can lead to bones and tissues crushing.
- Transient contact: A brief unconstrained contact between the robot and an operator. The operator can recoil or retract from the moving part of the robot as depicted in Figure 1.4B. The high dynamics of the robot motion can cause contusions, internal bleeding and even broken bones in such type of contact.

Considering such undesired physical interaction situations, the literature proposes different indicators to measure the dangerousness of a robot. The head being one of the weakest point of the human anatomy, much attention has been given to measure the dangerousness relatively to this part of the body. The most used criterion is the Head Impact Criterion introduced by J. Versace [Versace, 1971]. Originally used in car crash tests, this indicator for severe head injury measures the acceleration of the head over a time period. The HIC is formulated as

$$HIC = \max_{\Delta t} \left\{ \Delta t \left(\frac{1}{\Delta t} \int_{t_1}^{t_2} \|\bar{\boldsymbol{a}}_{head}\|_2 dt \right)^{\frac{5}{2}} \right\} \leq 650 \ (SI), \tag{1.1}$$

with $\bar{\boldsymbol{a}}_{head}$ the head acceleration, Δt the time lapse between t_1, the starting time of the collision and t_2, the stop time of the measurement. Δt is limited to a specific

value between 3 and 36 ms. S. Haddadin *et al.* show that when the movements of the head are not constrained, standard industrial robotic manipulators (such as the Kuka KR500[6]) do not exceed the HIC limit for severe injury [Haddadin et al., 2008b]. It means that the collision by itself is not life-threatening according to the Head injury criterion. However, the contact can induce other injuries such as bone fractures or internal damages that are not considered by this criterion. Though interesting this measure is not really adapted for the type of contact considered here. Indeed, during car crashes, the head acceleration is due to the sudden stop of the car, while during a collision with a robot, this acceleration is directly linked to the contact between the robot and the head. According to the tool that is being carried, *i.e.* sharp or with round edges, manufactured in soft or stiff materials, the results of this contact will be more or less dangerous.

For the evaluation of the dangerousness of a collision between a robot and a chest, several injury indexes are proposed, such as the Rib Deflection Criteria (RDB), the Compression Criterion [Haddadin et al., 2007] or the Viscous Criteria (VC) [Oberer et al., 2007]. VC relates the chest compression to its velocity of deformation. S. Haddadin *et al.* also performed impacts experiments with an object whose motions are constrained and cannot slip during contact [Haddadin et al., 2008a]. The robot follows a pre-planned trajectory at a velocity of 0.6 m/s and an object is placed along the trajectory. The forces exerted at the moment of impact are recorded and compared to anatomical thresholds, the Compression Criteria and the Viscous Criteria. It is shown that the efforts applied by a robot can be dangerous for the human and might lead to the crushing of body parts.

While all these indicators are representative of the severity of an injury caused by a robot, they use post-collision information and cannot be used in a control loop to ensure safety. To ensure safety at the control level, safety indicators must be defined relatively to physical quantities that can be monitored on-line.

The severity of a human injury caused by a robot depends on the human mass, its velocity, the protections that are used, *etc.* However, few assumptions can be made on a human state before the impact. On the other hand, a robot state can be precisely known at each control instant, from the mass of each link to its velocity and to the type of object that it carries. This information can be used to determine the robot dangerousness.

M. Wassink and S. Stramigioli propose to formalize the human/robot impact in terms of kinetic energy along the normal surface of contact [Wassink and Stramigioli, 2007]. The end-effector kinetic energy in the operational space is

$$e_c = \frac{1}{2}\boldsymbol{v}^T \Lambda(\boldsymbol{q})\boldsymbol{v}, \tag{1.2}$$

[6] www.kuka.com/en-gb/products/robotics-systems/industrial-robots/kr-500-fortec

with $\boldsymbol{v} \in \mathbb{R}^6$, the robot twist expressed at some point of interest and $\Lambda(\boldsymbol{q}) \in \mathbb{R}^{6\times6}$ the robot operational-space inertia matrix, which depends on the robot joint configuration $\boldsymbol{q} \in \mathbb{R}^n$, with n, the number of degrees-of-freedom of the robot. $\Lambda(\boldsymbol{q})$ describes the inertial properties of the end-effector at the operational point. $\Lambda(\boldsymbol{q})$ is related to the joint-space inertia matrix, $M(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ through the relation

$$\Lambda(\boldsymbol{q}) = \left(J(\boldsymbol{q})M^{-1}(\boldsymbol{q})J^T(\boldsymbol{q})\right)^{-1}. \tag{1.3}$$

For the specific task of positioning the robot end-effector in 3D space, only the terms associated with the linear velocity at the end-effector in the Jacobian, $J_{lin}(\boldsymbol{q})$, should be considered. In his work [Khatib, 1995], O. Khatib defines a robot effective mass, $m_u(\boldsymbol{q})$, perceived at the operational point along an arbitrary direction $\boldsymbol{u}$ as

$$m_{\boldsymbol{u}}(\boldsymbol{q}) = \boldsymbol{u}^T \left(J_{lin}(\boldsymbol{q})M^{-1}(\boldsymbol{q})J_{lin}^T(\boldsymbol{q})\right)^{-1} \boldsymbol{u} \tag{1.4}$$

$m_{\boldsymbol{u}}$[7] can also be interpreted as the perceived mass[8] at the end-effector in response to the application of a force along $\boldsymbol{u}$. The kinetic energy along the surface normal to contact, $\boldsymbol{u}$, becomes

$$e_{c,\boldsymbol{u}} = \frac{1}{2} m_{\boldsymbol{u}} v_{\boldsymbol{u}}^2 \tag{1.5}$$

with v_u the end-effector velocity along the direction $\boldsymbol{u}$. $e_{c,\boldsymbol{u}}$ represents the energy that would be transferred from the robot (considered rigid) to the operator in case of collision and can be directly linked to safety.

Another way to characterize the safety of a robotic solution is to measure the contact impulse force along a direction $\boldsymbol{u}$, $\hat{f}_{\boldsymbol{u}}$. This force represents the product of the average force and the time it is exerted. In his work, I.D. Walker [Walker, 1994] defines this impulse force just after the collision as

$$\hat{f}_{\boldsymbol{u}} = -\left(1+e\right) m_{\boldsymbol{u}} v_{\boldsymbol{u}} \tag{1.6}$$

with e, a restitution coefficient equal to 0 when contact is purely plastic, *i.e.* all the energy is transferred, and 1 for a purely elastic collision, *i.e.* there is no transfer of energy.

[7] It can be noted that if $J_{lin}(\boldsymbol{q})M^{-1}(\boldsymbol{q})J_{lin}^T(\boldsymbol{q})$ is not full-rank, *i.e.* if the robot is in a singular configuration, $m_{\boldsymbol{u}}(\boldsymbol{q})$ becomes infinite which can be interpreted as pushing against a wall.

[8] This term can also be referred in the literature as the robot equivalent mass or reflected mass.

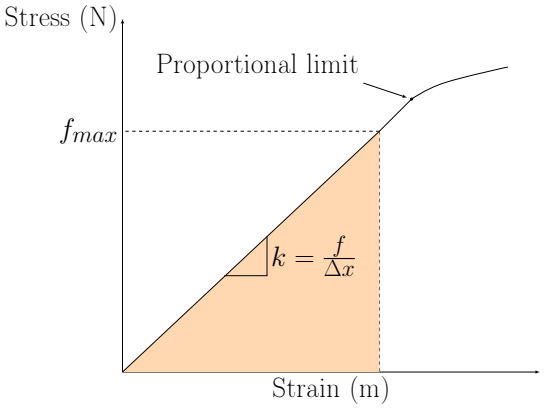
Equations (1.5) and (1.6) explicitly map the robot velocity, its mass and the colliding objects properties to the energy and impact forces during collision. During quasi-static contact, the force exerted by the robot on a human body part is directly linked to the torques generated by the actuators. These torques must also be monitored to limit dangerousness. Therefore, the energy of the robot and the exerted forces are good indicators of its dangerousness. Limits should be associated to these indicators to propose a control architecture ensuring safety.

1.3.3 Defining limits

ISO technical specification 15066 on collaborative robot safety is available to help manufacturers ensuring safety with their equipment [ISO/TS-15066, 2016]. In this specification, safety in collaborative applications can be provided through four methods: safety-rated monitored stop, hand-guiding, speed and separation monitoring, and power and force limiting. The first method consists in stopping the robot when an operator enters its workspace. This method is similar to the one used for robot in a separated workspace as depicted in Table 1.1. It is suited when the robot performs tasks that require few human operations (such as changing the robot tool). The second method corresponds to the one described in Table 1.1 and discussed in the introduction of this chapter. The third one consists in monitoring the robot velocity depending on the distance from the operator. The closer the operator is, the slower the robot moves. The last method consists in limiting the robot power and force during its motion. In this last method, the specification proposes to define a maximum permissible pressure and force that can be applied on different parts of the human body. These forces are determined using the equipment depicted in Figure 1.5A; which applies forces on different parts of a human body. The force limits correspond either to an injury level below AIS 1 [BGIA, 2011] or to "a sensation corresponding to the onset of pain". While this definition is rather vague, it can still be used to define a threshold on the force applicable on different body parts. From this measure it can be noted that the weakest part of the human body is the face and more specifically the masticatory muscle which can only hold a maximal tangential force of 65 N. The thighs and knees are the strongest and can withstand a force of 220 N. Similar quantities are provided for the maximum permissible pressure that can be exerted. These limits can be used to define the robot maximal applicable force/pressure during quasi-static contact.

Based on these maximal applicable forces, it is possible to estimate the maximum energy that can be transferred to a body part. ISO specification 15066 proposes to model human body parts as masses attached to a spring. The spring stiffness, k, is related to the proportion of soft tissues in the concerned body part. It assumes a fully inelastic contact

(A)



(B)

Source: Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung

FIGURE 1.5: (A) Experimental setup designed to apply efforts on a human body. Exerted forces are linked to human pain threshold. (B) Schematic representation of the relation between the maximal force applicable on a human body and its corresponding stiffness.

situation, *i.e.* all the kinetic energy is transferred to the body part. This corresponds to a worst-case situation. The norm assumes that the deformation of the concerned body part is elastic and satisfies Hook's Law, *i.e.* the stress is directly proportional to the strain (see Figure 1.5B). The resulting maximal energy that can be dissipated during a transient contact is determined by:

$$e_c = \frac{f_{max}^2}{2k} \tag{1.7}$$

where f_{max} is the force defined for quasi-static contact. ISO specification 15066 also provides the stiffness of different body parts. The face is still the weakest part of the human body during transient contact and can only withstand 0.11 J while the pelvis can absorb up to 2.6 J.

The robot kinetic energy and the force exerted by the robot are two safety indicators that can be used to control a robot dangerousness. From a design point of view, it is possible to create a robotic architecture reducing the energy dissipated at the impact. From a control point of view, it is also possible to develop special control methods to reduce a robot dangerousness in the pre-collision and in the post-collision phase. These solutions are presented separately in the two following sections.

1.4 Safety by design

Safety being related to the robot energy during transient contact, a key to reduce the dangerousness of a robot is to reduce the mass of its moving parts. This has been a priority on collaborative robot design for the last decade and has led to new robots such as the DLR LightWeight Robot [Albu-Schäffer et al., 2007]. With light materials and Harmonic Drive transmission, this robot has a mass/payload ratio close to 1 similarly to that of a human arm. It can lift loads of up to 7 kg with a repeatability of 0.1 mm. A new version, the KUKA LWR IIWA[9], is now available for industrial applications and features properties similar to the LWR 4+.

Most industrial robots have one actuator per joint. For serial manipulators, it requires placing bulkier actuators at the robot base to be able to move the remaining axes and their respective actuators. Another way to reduce the moving parts mass is to relocate these actuators at the base of the system. The Barrett WAM[10] is an example of a cable-actuated robot proposing such a design. Featuring a good repeatability, these robots have lighter links than standard robots. The actuated cables allow motions that are more silent compared to standard robots. However, they present flexibility that will deteriorate the robot precision depending on the payload that is being carried.

According to the Equation (1.6), another solution to reduce the dangerousness of the contact is to modify its coefficient of restitution, e. To do so, a special skin around the robot can be used to absorb a part of the impact. This has been done by K. Suita *et al.* [Suita et al., 1995] and J. J. Park *et al.* [Park et al., 2011]. These skins can be enhanced by embedding proximity sensors into them. The Aura cobot from Comau[11] or the skin from Fogale Robotics[12] propose such technology. However, these solutions only allow reducing contact dangerousness at a certain level.

Once a collision has occurred and kinetic energy has been transferred, the wrenches applied to the human body shall be limited. To do so, it is possible to improve the compliance of the robot. Passive compliance approaches consist in creating mechanical devices with built-in mechanical compliance. A first approach is to use a remote centre compliance (RCC) device which is attached between the last joint and the end-effector and is composed of 6 springs [Drake, 1977]. This device is composed of springs allowing a certain degree of compliance in the task. Initially intended for manufacturing tasks such as pegging [Watson, 1976], this device has been proposed to detect collisions [Bright and

[9] www.kuka.com/en-us/products/robotics-systems/industrial-robots/lbr-iiwa
[10] www.barrett.com/wam-arm/
[11] www.comau.com/EN/Pages/our_competences/robotics/Automation%20Products/Aura_test.aspx
[12] www.fogale-robotics.com/

Deubler, 1999]. However, these devices do not account for contact with other parts of the robot and are highly dependent on the robot tasks.

These drawbacks led to more sophisticated actuator architectures such as the variable stiffness actuators presented by A. Bicchi and G. Tonietti [Bicchi and Tonietti, 2004]. Such systems use common actuators associated with springs [Vanderborght et al., 2006]. They can be used to mimic the antagonist behaviour of human muscles to realize actuators with variable stiffness [Koganezawa, 2005], [Wolf and Albu-Schäffer, 2013]. G.A. Pratt and M.M. Williamson propose a series-elastic actuator featuring a spring between an axis gear and the output of its actuator [Pratt and Williamson, 1995]. The springs are used to filter out the high-frequency motion of the mechanism, allowing for a more stable force controller robot. They also protect against shock load but increase the joint flexibility. These actuators are used in the Baxter robot[13]. Similarly, in the work of L. Esteveny *et al.* , a one degree of freedom system is actuated using a combination of pulleys, a spring and a linear actuator [Esteveny et al., 2014]. The system is statically balanced by the spring. It is indirectly actuated by acting on the spring elongation using the linear actuator, allowing to have a back-drivable system that can limit the interaction forces between the robot and the environment. It can also accurately track a trajectory while in contact.

D. Shin *et al.* propose a hybrid actuation solution called Distributed Macro-Mini (DM^2) actuation using two actuators for each axis [Shin et al., 2010]. Trajectory tracking or gravity compensation tasks which have low time-varying properties are handled by a heavy actuator located at the robot base and using cable transmission. Smaller actuators are used for disturbance rejection by generating high frequencies and are located on each axis. Such design allows the relocation of actuators and reduces the robot mass but imply a more complex control architecture. A good overview of such enhanced actuators is exposed in [Vanderborght et al., 2013].

Though promising, DM^2 or variable stiffness actuators are still in an active research state and are not yet ready for the market. While these technologies have the potential to reduce the dangerousness of a robot during contact, the variable stiffness actuator solution requires sophisticated mechatronics and complex control schemes [Albu-Schäffer et al., 2010] and will probably not apply soon to applications where cost-effectiveness is an issue.

This work is intended for applications using currently available robots in the market. Since safety in an open environment cannot be entirely solved through design, the next section details different control techniques to improve a robot safety by control.

[13] http://sdk.rethinkrobotics.com/wiki/Arms

1.5 Safety by mean of control

From a control point of view, safety measures can be applied before and after the occurrence of a contact. Pre-collision methods provide solutions to avoid obstacles and compute commands preventing the robot from being dangerous. Post-collision methods propose to detect and react to an undesired interaction with the robot. These two phases are important in ensuring safety when sharing workspace with a robot.

1.5.1 Pre-collision methods

To prevent collision between a human and a robot, obstacle avoidance techniques have been developed. A popular approach, first introduced by O. Khatib [Khatib, 1986], consists in using potential fields. In this method, a virtual repulsive field is created around an obstacle. This field acts on the robot to push it away from the obstacle. This approach is popular as it offers a simple way to realise obstacle avoidance at a low computational cost. It is used in many robotic applications such as in [Flacco et al., 2014]. Usually, the obstacle is surrounded by an envelope that is larger than its real dimensions. This is a safety margin used to take into account any eventual measure imprecision in the obstacle geometry. This envelope restrains the robot motion leading to suboptimal control solution towards the task when the robot is near an obstacle. Furthermore, in cluttered environment this method shows some instabilities [Ren et al., 2006]. Obstacle avoidance techniques are highly dependent on the system detecting the obstacles. In industrial environments, the use of these systems is complex because of the environment (light, dust accumulation, ...) and the disposition of objects in the facility that constantly changes and can cause sensors obstructions. Ideally redundant sensors placed at different locations should be used, but this solution is expansive. Furthermore, in an unknown environment that is constantly evolving such as in Figure 1.1B, it is not possible to guarantee that no contact can happen. Consequently, control solutions ensuring safety in case of contact must be defined.

If a contact cannot be avoided, it is still possible to actuate a robot in a way that complies with ISO specification 15066 on robot safety. B. Matthias and T. Reisinger propose examples on how to implement this specification for different applications [Matthias and Reisinger, 2016]. The proposed approach is to reduce the robot maximal velocity and maximal torque to respect the recommended safety limits. Once again, if the task is known in advance, it is possible to define robot motions satisfying ISO specification 15066 recommendations. However, if the robot tasks are constantly redefined, the only way to comply with ISO 15066 recommendations with this method is to reduce the overall actuation capacities of the robot at the expense of the task tracking performances.

For a more optimal use of the robot capacities, one can use external measuring devices to record the position of an obstacle in the robot environment. The behaviour of the robot is adapted using this information. Safety zones can be implemented as in [Vogel et al., 2013] and used to stop the robot [Cherubini et al., 2013] or to reduce the robot velocity [Lasota et al., 2014]. Similarly, pre-collision safety can be ensured by computing a set of control inputs that respect a threshold on some safety markers. H. Jochen and A. Zelinsky propose to compute a set of torque commands that respect a limit on the impact force as expressed Equation (1.6) [Jochen and Zelinsky, 2003]. In M. Laffranchi *et al.* , [Laffranchi et al., 2009], a single axis system adapts its reference trajectory to prevent the accumulation of energy during contact. These solutions do prevent the robot from being dangerous during an undesired interaction with a human. However, the methods used to lower the robot performances does not guarantee an optimal achievement of the desired tasks. In the work of A. Meguenani *et al.* , the control problem is expressed as an optimization one that is subject to a constraint on the robot kinetic energy [Meguenani et al., 2015]. It finds an optimal solution to the realisation of the robot tasks without generating more kinetic energy than a defined limit.

Once the robot has entered in collision with an obstacle, control solutions should prevent the robot from applying dangerous forces on the obstacle. This is the topic of the next section.

1.5.2 Post-collision methods

Collision detection algorithms have been developed using the robot theoretical torques required to perform a task and comparing them with the applied ones [Yamada et al., 1997]. The difference gives an estimation of the interaction with the environment and even information such as the direction of the collision [De Luca and Ferrajoli, 2008]. To have a good knowledge of the interaction forces applied on the robot, many contributions focus on torque sensing robots [Albu-Schäffer et al., 2007]. N. Briquet-Kerestedjian *et al.* propose to use a Kalman filter to improve collision detection by removing characterized uncertainties coming either from an imperfect robot model or from quantification noises [Briquet-Kerestedjian et al., 2017]. The interested reader can refer to [Haddadin et al., 2017] for a detailed survey on detection, isolation and identification of collisions with a robot. Once a collision is detected, several reactions can be used, from a simple stop, to gravity compensation or motion in the direction opposite to the contact [De Luca et al., 2006].

If the robot must carry on its task after a collision, impedance control can be used to prevent the robot from exerting dangerous forces when a contact occurred. First

introduced by N. Hogan, [Hogan, 1984], impedance control implements a mass-spring-damper behaviour between a target position and the actual position of the robot. This relation is expressed as

$$\boldsymbol{f} = M\Delta\dot{\boldsymbol{v}} + K_p\Delta\boldsymbol{x} + K_d\Delta\boldsymbol{v}, \tag{1.8}$$

where M, K_p and K_d are positive definite matrices representing the controller virtual Cartesian inertia, stiffness and damping. A deviation of the robot current position towards this target position is related to a control force $\boldsymbol{f}$. A stiffly actuated robot can be assimilated to a stiff spring: small errors will generate strong torques to correct the positioning error. On the contrary a soft impedance control will allow large position error before imposing important torques.

Stiffness coefficients must be set according to the task and a trade off must be made between a soft, imprecise but safe robot and a stiff, precise but possibly dangerous robot. A major drawback of this approach is that a lack of stiffness, added to model imperfections (such as dry friction at the joint level), leads to a positioning error affecting the general precision. To overcome such drawbacks, one can increase the controller gains at the expense of safety. Variable impedance control proposes to adapt these coefficients on-line depending on the interaction with the environment. G. Raiola *et al.* propose to adapt these gains reactively to limit the robot power and kinetic energy [Raiola et al., 2018]. However, variable impedance control can lead to instabilities. Indeed, K. Kronander *et al.* show that modifying the robot stiffness on-line can lead to internal energy production leading to a loss of stability [Kronander and Billard, 2016]. Using energy tank techniques, B. Hannaford *et al.* show that it is possible to prevent the production of energy inside the controller and ensure stability [Hannaford and Jee-Hwan, 2000]. This is the approach proposed in [Raiola et al., 2018]. However, a technique able to modify online the impedance gains in an optimal way to be both precise and safe when required is yet to be found.

When a contact is established, energy is virtually accumulated in the robot controller. This energy results in a force pushing in the direction of the desired position. In the case of a spring-damper impedance controller, this energy is conservative and can be assimilated to a potential energy driving the robot towards its desired position. In Meguenani *et al.* , the forces pulling the robot are derived from this potential energy and linked to the robot torques exerted by the actuators [Meguenani et al., 2017]. By constraining this energy, using an optimization control scheme, Meguenani *et al.* optimally restrain the robot from exerting dangerous forces while tracking a trajectory.

All the proposed solutions can improve the overall safety of a robotic solution. However, they only handle contact from either the transient or the quasi-static contact case or induce suboptimal tracking performances. To ensure safety these algorithms rely on some assumptions about the environment (robot in contact or in free motion). As such, these solutions require collision detection algorithms and a switching between two command modes. This switching can introduce discontinuities in the control law inducing unpredictable and potentially dangerous robotic behaviour. Dedicated control solutions such as energy tanks control approaches can be used to ensure stability but lead to sub-optimal tracking solutions [Ferraguti et al., 2013].

1.6 Proposed contribution

When robots can perform autonomous tasks in an open environment, safety has to be dealt with. If contact cannot be avoided, one must find control solutions to prevent dangerous interactions with a human. It has been seen that during quasi-static contact with a human, the wrenches applied by the robot must be below a limit to avoid severe injury. Similarly, during transient contact, one must monitor the robot kinetic energy. A safe control solution should be able to handle both cases of contact. It should also avoid switching between different command modes to avoid discontinuities when the nature of contact changes between a motion in free space and a motion in a constrained space.

The work energy theorem states that the work done by all the forces acting on an object equals the change in kinetic energy of this object. For a system moving from point A to point B:

$$\Delta e_c^{A \to B} = \int_A^B \boldsymbol{f} \boldsymbol{du}(t), \tag{1.9}$$

where Δe_c is the system variation of kinetic energy, $\boldsymbol{du}(t)$ defines the trajectory from point A to point B and f represents the forces acting on the system.

This variation of kinetic energy can be used as a safety indicator. It can be used to characterize the dangerousness of a robot generating too much kinetic energy while in free motion or applying too much efforts while in contact. This indicator should be minimized to ensure that the robot is as safe as possible. It should also be constrained to stay within a safe limit.

If the robot velocity is fixed by the application, the only way to minimize its variation of kinetic energy is to minimize its perceived mass. This perceived mass is linked to the robot configuration as detailed in Section 1.3.2. Using a redundant robot, it is thus

possible to find a configuration inducing the lowest perceived mass in the direction of an obstacle. However, this does not guarantee that this energy stays below a defined safe limit.

Expressing the variation of kinetic energy as a constraint is an efficient way to prevent the robot from developing a dangerous amount of energy during its motion. This requires, from a control point of view, to define a control architecture suitable to optimally achieve the desired task while satisfying the constraints.

This thesis treats these two aspects which can be summarized as:

- Minimize the robot perceived mass to reduce the energy at the impact and limit the impact forces
- Constrain the robot variation of kinetic energy to both ensure safety during transient and quasi-static contact.

This work aims at developing control strategies suitable for any robotic application. To that extent, the proposed contribution is developed for a generic robotic manipulator moving in an environment shared with humans. It proposes contributions to alleviate remaining problems in the third row of Table 1.1. A control architecture that computes control solutions complying with the robot constraints and ensuring an optimal trajectory tracking is developed. A kinetic energy constraint is specifically designed to comply with ISO specification 15066. To this end, the control problem is expressed as an optimization one. The Mathematics used to define the control problem are developed for generic tasks and constraints. Digital Tomosynthesis is used as a concrete application example of the proposed controller. In this application, the robot must perform two main tasks:

- A positioning task ensuring that the robot follows a trajectory in 3D space to perform the DT procedure. This trajectory can be either defined off-line or reactively computed and adapted on-line.
- A pointing task which keeps the robot pointing towards a fixed point related to the centre of the X-ray imaging detector at all time.

The problem formulation is generalized for any fixed base serial redundant robot and control solutions are proposed to advantageously use the degree of freedom left to realise secondary tasks related to safety.

1.7 Structure of this manuscript

This manuscript exposes a control framework that is able to ensure that a robot is not dangerous in case of an undesired contact with a human. The aim of this framework is to realise tasks at best while satisfying a set of constraints. Two common families of problem resolution methods can be used for the realisation of tasks under constraints: *Explicit Inversion Methods* (EIM) and *Quadratic Programming* (QP). They are described in the first part of Chapter 2 and the second method is selected for the rest of this work. The second part of Chapter 2 thoroughly details the expression of tasks and constraints inside the QP formulation. In particular, it details the expression of the kinetic energy variation in Equation (1.9) as a constraint. Quadratic programming features solutions to use the robot redundancy to realise secondary tasks. This secondary task, that must not interfere with the main tasks, can be the definition a desired joint configuration to reach. Using this property, it is possible to find a configuration minimizing the robot perceived mass. Each aspect of the problem formulation is detailed and used in the specific context of Digital Tomosynthesis.

Several experiments are then conducted on a KUKA LWR4+ robot are then described. The experimental setup is described in Chapter 3. External sensors are used to validate the tasks and the correct constraint enforcement. The description of the real-time control architecture is also given.

Chapter 4 presents several experiments realised with the robot and the external sensors. It provides several experiments realised in the context of Digital Tomosynthesis to show the interesting properties of the presented control solution. This chapter also features several experiments showing that by using the robot redundancy it is possible to reduce its perceived mass in the direction of an obstacle and thus reduce its dangerousness towards this obstacle.

Chapter 5 concludes this work by summarizing the contributions that are presented in this document. It then provides some research perspectives that could be carried out in the field of safety using the tools developed in this work.

1.8 Related publications

Part of the work presented in this manuscript has been published in ICRA 2018:

[Joseph *et al.*,2018a] *Joseph, L., Padois, V., and Morel, G. (2018). Towards X-ray medical imaging with robots in the open: safety without compromising performances.*

IEEE International Conference on Robotics and Automation (ICRA). https://hal.archives-ouvertes.fr/hal-01614508/en

Another contribution is in press for ISER 2018:

[Joseph *et al.*,2018b] *Joseph, L., Padois, V., and Morel, G. Experimental validation of an energy constraint for a safer collaboration with robots. International Symposium on Experimental Robotics (ISER). Manuscript in press* https://hal.archives-ouvertes.fr/hal-01883995/en

Chapter 2

Development of a control architecture for safety

Table 1.1 presents different robotic applications requiring the robot to realise all sorts of tasks. The realisation of these tasks requires to send some desired control inputs to the robot actuators. These control inputs are either expressed as desired joint positions, velocities, or force/torques. Independently of the control inputs, a robot is subject to the laws of physics. These laws are summarised by the equation of motion which relates the external forces acting on a robot to its joint acceleration. For a fixed based robot with n degrees of freedom (dof) this equation can be written

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}, ...) = \boldsymbol{\tau} - \sum_{i=1}^{N} J_{C_i}^T \boldsymbol{\omega_i}. \tag{2.1}$$

$\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} \in \mathbb{R}^n$ respectively represent the joint position, velocity and acceleration. $M(\boldsymbol{q}) \in \mathbb{R}^{n \times n}$ represents the joint-space inertia matrix, $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^n$ corresponds to the gravity induced joint torque, $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^n$ represents the joint torque induced by Coriolis/Centrifugal effects and $\boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}, ...) \in \mathbb{R}^n$ represents the joint torque induced by other forces such as dry friction. $\boldsymbol{\tau} \in \mathbb{R}^n$ is the joint actuation torque vector. Considering N contact points with the robot, J_{C_i} represents the contact Jacobian at the point i and $\boldsymbol{\omega}_i$ is the contact wrench exerted by the environment on the robot.

Equation (2.1) only describes the unconstrained robot dynamic. Indeed, the robot is also subject to intrinsic physical constraints including:

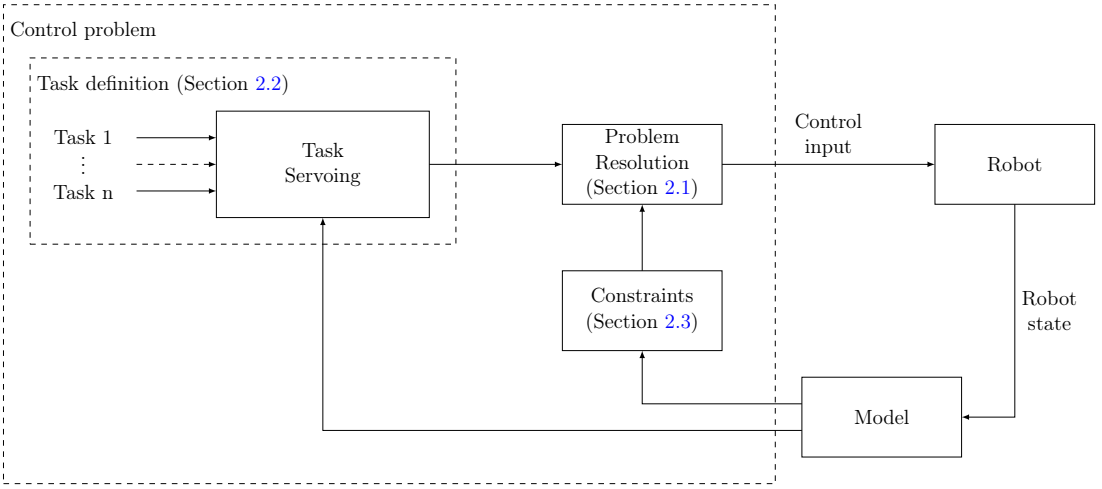


FIGURE 2.1: General control scheme

- joint position limits

$$\boldsymbol{q_{min}} \leq \boldsymbol{q} \leq \boldsymbol{q_{max}}, \tag{2.2}$$

- joint velocity limits

$$\boldsymbol{\dot{q}_{min}} \leq \boldsymbol{\dot{q}} \leq \boldsymbol{\dot{q}_{max}}, \tag{2.3}$$

- joint torque limits

$$\boldsymbol{\tau_{min}} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau_{max}}. \tag{2.4}$$

These limits are usually given by the robot manufacturer. Control inputs leading to a violation of these constraints will not be feasible by the robot and should be forbidden. It must also be forbidden to send control inputs leading to a dangerous robot behaviour. Hence, the safety indicator presented in Chapter 1 should also be considered as a constraint.

A control problem can thus be defined as: find a continuous sequence of control inputs allowing the robot to go from an initial pose to a goal pose, satisfying the robot equation of motion and the constraints. It can be noted that from a trajectory generation point of view, it is possible to find, a priori, a trajectory satisfying this problem. However, due to model uncertainties, it is always necessary to use feedback to reject disturbances induced by modelling errors. The control problem thus requires both the computation of a feasible trajectory and the computation of a suitable control input to track this trajectory. The second operation is called task servoing.

Figure 2.1 depicts a generic control architecture to solve the proposed control problem. This architecture translates tasks either defined off-line or reactively updated into

control inputs satisfying the robot constraints. The model of the robot is used to determine the dynamic terms in the robot equation of motion. These terms are used for the task servoing and the expression of the constraints.

This chapter first details the mathematical tools that can be used to solve the control problem. It then details the expression of the tasks and the constraints for any fixed based serial robot. Through this chapter, special attention is paid to the improvement of the robot safety towards its environment using the described tools.

2.1 Control problem resolution methods

Let's consider a n degrees of freedom robot realising a task in the operational space and controlled at the joint velocity level. Its desired operational twist, expressed at some point of interest (usually the end-effector), $\boldsymbol{v}^*$, is linked to its joint velocity through the Jacobian $J(\boldsymbol{q})$. The control problem can thus be expressed as

$$\text{find } \dot{\boldsymbol{q}} \text{ such that } \boldsymbol{v}^* = J(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2.5}$$

while respecting a set of constraints such as the ones described in Equations (2.2) to (2.3).

Control problem resolution methods are used to determine the optimal control input to Problem (2.5). Two families of methods exist for the resolution of this problem: *Explicit Inversion Methods* and *Constrained optimization methods*. The first class of methods is based on linear algebra and the explicit inversion of the robot Jacobian to deal with constraints in a passive (clamping) or active (avoidance) way using some heuristics. The second expresses the problem as an optimization one.

2.1.1 Explicit inversion methods

Explicit inversion methods propose to inverse the robot Jacobian to determine an adequate control input to solve the problem presented in (2.5). When the task to achieve requires n dof, $J(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ can be inverted with a unique solution if $rank(J(\boldsymbol{q})) = n$. However, if the task requires m dof with $m < n$, the inversion of $J(\boldsymbol{q}) \in \mathbb{R}^{m\times n}$ yields an infinity of solutions. A solution to the problem can be written

$$\dot{\boldsymbol{q}}^c = J(\boldsymbol{q})^+ \boldsymbol{v}^*, \tag{2.6}$$

where $\dot{\boldsymbol{q}}^c \in \mathbb{R}^n$ is the commanded joint velocity and $J(\boldsymbol{q})^+$ is the Moore-Penrose inverse of $J(\boldsymbol{q})$ [Penrose, 1955]. This pseudoinverse gives the minimum norm to $\boldsymbol{v}^* = J(\boldsymbol{q})\dot{\boldsymbol{q}}$ in the least square sense. This solution does not account for the robot constraints yet.

When the robot is redundant relatively to its task ($m < n$), there remains some unused degrees of freedom. A. Liegeois proposes to use advantageously this property to perform secondary tasks [Liégeois, 1977]. Given the defined problem in (2.5), A. Liegeois proposes the following solution

$$\dot{\boldsymbol{q}}^c = J^+\boldsymbol{v}^* + \left(I_n - J^+J\right)\dot{\boldsymbol{q}}_{\boldsymbol{0}} \tag{2.7}$$

where $(I_n - J^+J)$[1] is a projector on the Jacobian kernel, with $I_n \in \mathbb{R}^{n\times n}$ the identity matrix, and $\dot{\boldsymbol{q}}_{\boldsymbol{0}} \in \mathbb{R}^n$, an arbitrary joint velocity vector. By selecting a suitable $\dot{\boldsymbol{q}}_{\boldsymbol{0}}$, it is possible to perform a second task in the null-space of the first one. This way, objectives are strictly hierarchised and a low-level objective is carried out as long as it does not interfere with objectives of higher level. In his work, Liegeois proposes an algorithm to keep the robot away from its joint limits by defining a secondary task expressed as a gradient descent of the joint configuration and projected into the null-space of the main task Jacobian [Liégeois, 1977].

An extension of the null-space projector approach is developed in [Siciliano and Slotine, 1991] and generalizes the idea for multiple tasks. The multitask problem consists in finding the joint velocity, $\dot{\boldsymbol{q}}^c$, satisfying

$$\underbrace{\begin{pmatrix} \boldsymbol{v_1}^* \\ \vdots \\ \boldsymbol{v_i}^* \end{pmatrix}}_{\boldsymbol{v}^*_{(i)}} = \underbrace{\begin{pmatrix} \boldsymbol{J_1} \\ \vdots \\ \boldsymbol{J_i} \end{pmatrix}}_{J_{(i)}} \dot{\boldsymbol{q}} \tag{2.8}$$

for the i^{th} first tasks. $\boldsymbol{v}_i^* \in \mathbb{R}^m$ and $J_i \in \mathbb{R}^{m\times n}$ are respectively the desired operational velocity and the Jacobian associated to the i^{th} task. $J_{(i)} \in \mathbb{R}^{m\times in}$ is an extended Jacobian. By recursively projecting lower level tasks in the Jacobian kernel of the previous ones (with higher priority), a strictly hierarchised control solution can be formulated as

$$\dot{\boldsymbol{q}}_i^c = \dot{\boldsymbol{q}}_{i-1}^c + (J_iP_{i-1})^+\left(\boldsymbol{v}_i - J_i\dot{\boldsymbol{q}}_{i-1}\right) \tag{2.9}$$

[1] For the sake of clarity the dependence to $\boldsymbol{q}$ is omitted.

with $P_i = (I_n - J^+_{(i)} J_{(i)})$ and $\dot{\boldsymbol{q}}^c_1 = J^+_1 \boldsymbol{v}_1$.

The formulation in Equation (2.9) has been used to hierarchise several tasks with a strict priority order. O. Stasse *et al.* use a gradient descent method to define a secondary task to avoid self-collisions on a humanoid robot in real time [Stasse et al., 2008]. Flacco *et al.* propose a control architecture where three tasks are strictly hierarchised and executed by a redundant manipulator [Flacco et al., 2014]. Similarly, O. Khatib uses potential fields [Khatib, 1986] to avoid obstacles or joint boundaries. Potential field methods allow formulating the task and the constraint related tasks at the same level. However, they do not guarantee the respect of the constraints. Another approach, proposed by J. Baillieul, extends the main task Jacobian by adding auxiliary tasks so that the Jacobian becomes a square invertible matrix [Baillieul, J, 1985]. This technique is well suited to avoid singularities but cannot be easily translated to other type of constraints. Some drawbacks of the mentioned explicit inversion methods are that conflict may arise between different tasks and most importantly, not every constraint can be projected into the null-space of the main task Jacobian. Indeed, the number of constraints (joint limits, obstacles to avoid, *etc.*) is potentially higher than the number of available degrees of freedom. In such a situation, the null-space of the previous tasks is potentially empty, and constraints cannot be enforced.

A. Maciejewski and C. Klein propose instead to project the main task into the null-space of the Jacobians of constraints related tasks [Maciejewski and Klein, 1985]. This way, it ensures that constraints are satisfied priorly to the realisation of the main tasks. Again, if the number of constraints, n_c, is too large ($n_c \geq m$), the null-space of the Jacobians of constraints related tasks is empty and the robot won't be able to perform its main tasks. However, not every constraint related task needs to be activated at the same time. Based on this observation, constraints activation techniques have been developed. By only activating the tasks critical to the satisfaction of each constraint, it is possible to leave some degrees of freedom for the main tasks. To avoid conflicts between tasks, one needs to introduce passive avoidance techniques. Rather than trying to move away from constraints, passive avoidance techniques prevent motions towards the constraints.

Based on these two concepts, P. Baercloacher and R. Boulic propose an algorithm to iteratively perform a whole model inversion satisfying joint position constraints [Baerlocher and Boulic, 2004]. If a joint must go beyond its boundary for the correct achievement of the task, it is clamped to its boundary. A new model inversion is then performed without accounting for the clamped joints until a solution that does not require more joint clamping is found. While interesting, this algorithm can lead to sub-optimal solutions (see Chapter 3 in the thesis of S. Rubrecht [Rubrecht, 2011]) and can only be

applied for joint position constraints. In the work of F. Flacco, an algorithm is proposed to account for the robot dynamic and constrains the joint position, velocity and acceleration [Flacco et al., 2012]. The proposed approach scales the task to be performed to satisfy the constraints. This approach has been extended to find an optimal joint velocity command satisfying the constraints in [Flacco, 2013].

A major drawback of these explicit inversion methods arises from the inability to express constraints as inequalities. Constraints are thus taken as equalities that are activated based on some heuristic methods that do not always guarantee the optimality of the solution. Furthermore, these methods cannot be easily extended to any type of constraint.

2.1.2 Constrained Convex Optimization Methods

The problem expressed in (2.5) can also be seen as the resolution of an optimization problem seeking a solution minimizing the norm of $J(\boldsymbol{q})\dot{\boldsymbol{q}} - \boldsymbol{v}^*$ and satisfying the constraints expressed in Equations (2.2) to (2.3). The general form of an optimization problem is

$$\begin{aligned} &\textit{minimize} && f(\boldsymbol{x}) \\ &\text{subject to} && \boldsymbol{h}(\boldsymbol{x}) \leq \boldsymbol{0} \end{aligned} \tag{2.10}$$

where $f(\boldsymbol{x}) \in \mathbb{R}^n \rightarrow \mathbb{R}$ is called an objective function to be minimized according to the optimization variable $\boldsymbol{x} = (x_1, ..., x_n) \in \mathbb{R}^n$. The optimization can be subject to inequality constraints expressed as a function of the optimization variable in $\boldsymbol{h}(\boldsymbol{x}) \in \mathbb{R}^n \rightarrow \mathbb{R}^{n_c}$, with n_c the number of constraints. An optimal solution to the optimization problem has the smallest objective value among the set of solutions satisfying the constraints.

In a general form, the objective function associated with problem (2.5) can be written as $f(\boldsymbol{x}) = \|E\boldsymbol{x} - \boldsymbol{f}\|_2^2$ with $\|\cdot\|_2$ the Euclidean norm. In the specific case of Problem (2.5), $E = J$, $\boldsymbol{x} = \dot{\boldsymbol{q}}$ and $\boldsymbol{f} = \boldsymbol{v}^*$. This function is convex and can be written in a quadratic form

$$\begin{aligned} \|E\boldsymbol{x} - \boldsymbol{f}\|_2^2 &= (E\boldsymbol{x} - \boldsymbol{f})^T (E\boldsymbol{x} - \boldsymbol{f}) \\ &= \boldsymbol{x}^T E^T E\boldsymbol{x} - 2\boldsymbol{f}^T E\boldsymbol{x} + \boldsymbol{f}^T \boldsymbol{f} \\ &= \frac{1}{2}\boldsymbol{x}^T H\boldsymbol{x} + \boldsymbol{g}^T \boldsymbol{x} + r \end{aligned} \tag{2.11}$$

with $H \in \mathbb{R}^{n \times n}$ and $\boldsymbol{g} \in \mathbb{R}^n$ respectively the Hessian matrix and the gradient vector of the objective function and r a scalar. Linear quadratic programming is a branch of optimization for which the objective function is quadratic and the constraint functions are affine. Linear quadratic programming methods seek the optimal solution, $\boldsymbol{x}^{opt}$, to the following problem

$$\begin{aligned} \boldsymbol{x}^{opt} = \quad & \underset{\boldsymbol{x}}{\arg\min} \quad \tfrac{1}{2}\boldsymbol{x}^T H \boldsymbol{x} + \boldsymbol{g}^T \boldsymbol{x} + r \\ & \text{s.t.} \qquad \boldsymbol{l_b} \leq A\boldsymbol{x} \leq \boldsymbol{u_b} \\ & \qquad\qquad C\boldsymbol{x} = \boldsymbol{d}. \end{aligned} \tag{2.12}$$

The constraint functions are divided in equality and inequality constraints with A and C the corresponding constraint matrices and $\boldsymbol{l_b}$ and $\boldsymbol{u_b}$ respectively the inequality constraint lower and upper bound and $\boldsymbol{d}$ a vector associated to the equality constraint. If the objective function is convex, *i.e.*

$$f(\alpha\boldsymbol{x} + \beta\boldsymbol{y}) \leq \alpha f(\boldsymbol{x}) + \beta f(\boldsymbol{y}) \tag{2.13}$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and all $\alpha, \beta \in \mathbb{R}^+$ with $\alpha + \beta = 1$, a local minimum of the function is also its global minimum. This property eases the problem resolution and reduces computation time. Problem (2.5) can thus be solved using quadratic programming methods.

2.1.2.1 Multi-tasking

Constrained convex optimization methods also allow the simultaneous realisation of several tasks. In order to avoid interferences between different tasks, task prioritization strategies can be used.

Hierarchical prioritization Hierarchical prioritization strategies are used to strictly determine the level of importance of a task with relation to others. As presented in Algorithm 1, the solution of a task of higher level is used in the equality constraint of the following optimization problem.

This hierarchization can be applied for any number, n_t, of tasks. It ensures that the solutions of the lower level tasks do not interfere with the tasks of higher level. However, if the previous tasks use every degree of freedom available, the tasks of lower level will not be achieved. When a robot has to realise several tasks over time, the priority of each task may vary depending on the context situation. Strict hierarchical prioritization

Algorithm 1: Hierarchical task prioritization

1. Compute the optimal solution for the main task

$$\begin{aligned} \boldsymbol{x}^{opt} = \quad & \arg\min_{\boldsymbol{x},\boldsymbol{v}} && \|\boldsymbol{v}\|_2^2 \\ & \text{s.t.} && \boldsymbol{l_b} \leq A\boldsymbol{x} \leq \boldsymbol{u_b} \\ & && C\boldsymbol{x} = \boldsymbol{d} \\ & && \boldsymbol{v} = E_0\boldsymbol{x} - \boldsymbol{f_0}. \end{aligned}$$

2. Store $\boldsymbol{v}_0^{opt}$, the result of the first optimization.
3. Use the results of the former optimization in the equality constraint of the next task optimization

$$\begin{aligned} \text{for } i = 1...n_t \quad \boldsymbol{x}^{opt} = \quad & \arg\min_{\boldsymbol{x},\boldsymbol{v}} && \|\boldsymbol{v}\|_2^2 \\ & && \boldsymbol{l_b} \leq A\boldsymbol{x} \leq \boldsymbol{u_b} \\ & && C\boldsymbol{x} = \boldsymbol{d} \\ & \text{s.t.} && \boldsymbol{v}_0^{opt} = E_0\boldsymbol{x} - \boldsymbol{f_0} \\ & && \vdots \\ & && \boldsymbol{v}_{i-1}^{opt} = E_{i-1}\boldsymbol{x} - \boldsymbol{f_{i-1}} \\ & && \boldsymbol{v} = E_i\boldsymbol{x} - \boldsymbol{f_i} \end{aligned}$$

Store $\boldsymbol{v}_i^{opt}$, the result of the i^{th} optimization.

return $\boldsymbol{x}^{opt}$

makes it complicated to address this transition of tasks priorities. Instead, one can use soft priorities to alleviate this problem.

Weighted prioritization In weighted prioritization, each task is represented as a weighted norm. The QP formulation is the sum of the weighted tasks such that

$$\begin{aligned} \boldsymbol{x^{opt}} = \quad & \arg\min_{\boldsymbol{x}} && \sum_{i=1}^{n_t} \|E_i\boldsymbol{x} - \boldsymbol{f}_i\|_{W_i}^2 \\ & \text{s.t.} && \boldsymbol{l_b} \leq A\boldsymbol{x} \leq \boldsymbol{u_b} \\ & && C\boldsymbol{x} = \boldsymbol{d}. \end{aligned} \tag{2.14}$$

where W_i is a weighting matrix for the ith task. The weights are used to define a soft hierarchization of task importance towards each other. This does not ensure that tasks do not interfere with one another, but by adapting the weights it is possible to smoothly transition between different tasks.

Hybrid prioritization Hybrid hierarchization schemes can be used to combine the benefits of both methods. An approach to continuously transition a priority between hierarchical and weighted prioritization using scalar values is proposed in the work of

[Liu et al., 2016]. This approach provides an interesting technique to switch between two strategies but is computationally more expensive.

2.1.2.2 Problem resolution

Quadratic programming problems cannot be solved analytically. Iterative methods must be used to find an optimal solution satisfying the defined constraints. Interior point methods (also referred to as barrier methods) express the problem as an unconstrained optimization one, where constraints are expressed as objective functions. Problem (2.12) is rewritten as

$$\begin{aligned} \boldsymbol{x}^{opt} = \quad & \arg\min_{\boldsymbol{x}} \quad f(\boldsymbol{x}) + \sum_{i=1}^{n_c} \alpha_i(\boldsymbol{x}) \\ & \text{s.t.} \qquad C\boldsymbol{x} = \boldsymbol{d}. \end{aligned} \tag{2.15}$$

with

$$\alpha_i(\boldsymbol{x}) = \begin{cases} 0 \text{ if } A_i\boldsymbol{x} - b_i < 0 \\ \infty \text{ if } A_i\boldsymbol{x} - b_i > 0 \end{cases} \tag{2.16}$$

Where n_c is the number of constraints, A_i and b_i are respectively the constraint matrix and the associated limit for the i^{th} constraint. When a solution gets close to a constraint, $\alpha_i(\boldsymbol{x})$ gets a more penalizing weight. This prevents the solution from going beyond the bound. Because $\alpha(\boldsymbol{x})$ is not convex, it is approximated using log functions such that $\alpha_i(\boldsymbol{x}) = -\log(A_i\boldsymbol{x} - b_i)$. When $A_i\boldsymbol{x}$ is close to b_i, α_i gets close to ∞. Interior-point methods work well for small problems with a good approximation accuracy.

It can be noted that when a constraint is reached (or activated), a subset of the solution to the optimization problem lies on this constraint. The problem can thus be rewritten as an optimization problem with equality constraints where only the activated constraints are considered. The challenge is then to discover which constraint is activated using as few operations as possible. This is the topic of active-set methods. For more detailed information on convex problem optimization resolution methods, the interested reader can refer to [Boyd and Vandenberghe, 2004]. Quadratic programming problem resolution is still an active research topic. New quadratic programming solvers are developed regularly. Some recent ones include qpOASES[2] [Ferreau et al., 2014] and

[2] qpOASES quadratic programming Online Active Set Strategy: www.qpOASES.org/

OSQP[3] [Stellato et al., 2017]. These new solvers are developed to increase the resolution performances in terms of speed to find a solution, ratio of problems solved, *etc.*

Until recently, these algorithms were not fast enough to solve complex problems in real time. B. Faverjon and P. Tournassoud proposed an obstacle avoidance algorithm using convex optimization where the distance to the closest obstacle is set as a constraint [Faverjon and Tournassoud, 1987]. Rather than generating a control input inducing a motion in the opposite direction to the obstacle, this obstacle avoidance technique proposes to prevent motions in the direction of this obstacle. Due to computation capacities limits at the time, this optimization was performed off-line to avoid collisions with a known environment. A. Kapoor *et al.* use a convex optimization scheme where constraints are used to define virtual fixtures that the robot must not cross [Kapoor et al., 2006]. The control loop in these experiments has a periodicity of 30 ms which is sufficient for position control. For torque controlled robots, a smaller control loop periodicity is required to account for the robot dynamic [Khosla, 1987]. The recent improvements in the field of optimization and the progress in computing capacities of recent computers now allows solving problems under the millisecond for robotic manipulators [Meguenani et al., 2017] as well as humanoid robots [Kuindersma et al., 2014].

Quadratic programming methods have interesting properties that makes them suitable to deal with multiple tasks while strictly enforcing constraints. Through this work, the control problem is formulated as a quadratic programming one. The remaining of this chapter details the construction of this problem. Each step in the definition of the problem is explained for a generic n dof manipulator and the contributions to safety are outlined.

2.2 Task definition

Table 1.1 presents different application cases for a robot. In each application a robot must realise a specific task that can be expressed in terms of a position to reach, a force to exert, *etc.* Problem (2.5) illustrates the elements that are required for the expression of a robot task: the knowledge of the robot current state ($\boldsymbol{q}$, $\dot{\boldsymbol{q}}$) and the definition of a desired state to reach ($\boldsymbol{q}^{des}$, $\dot{\boldsymbol{q}}^{des}$). The robot desired motion is defined by a task planner and is tracked using servoing techniques and linked to a desired state using the robot model. This section details the expression of common robot tasks in an optimization scheme.

[3]OSQP An Operator Splitting Solver for Quadratic Programs: `www.osqp.org/`

2.2.1 Task planning

In this document, task planning consists in the definition of a trajectory to follow in order to realise a task. This trajectory is defined by a path to go from an initial position to a goal position and a timing law. Depending on the nature of the task, the path can be expressed either in the Cartesian space or in the joint configuration space ([Khatib and Siciliano, 2008] Chapter 6). The timing law depends on the task specification (maximal velocity, stop at a specific location, *etc.*). Different techniques exist to define a trajectory. A straightforward solution to define a timing law consists in using the maximal allowed acceleration and deceleration to perform the motion. This led to the definition of s-curves motions [Nguyen et al., 2008] where the motion profile is defined as a polynomial function. Rapidly exploring random trees techniques can be used to compute a collision-free trajectory [Lavalle, 1998]. Bounds on the robot joint acceleration and velocity can also be taken into account [Kunz and Stilman, 2012]. Trajectory generation can also be considered as an optimization problem seeking a collision free path under actuation constraints. As such several trajectory planners have been proposed in the literature such as [Ratliff et al., 2009] or in [Kalakrishnan et al., 2011].

2.2.2 Task servoing

Task servoing techniques are used to track the trajectory determined during tasks planning. It aims at rejecting the disturbances induced by errors in the robot model that would prevent the correct following of the trajectory. A common task servoing technique relies on a proportional–integral–derivative (PID) controller which is defined as

$$\boldsymbol{\xi}^* = K_p \boldsymbol{e} + K_d \dot{\boldsymbol{e}} + K_i \int_{t_0}^{t} \boldsymbol{e} d\tau + \boldsymbol{\xi}^{des} \tag{2.17}$$

where K_p, K_d and K_i are respectively the proportional, derivative and integral gains of the controller. $\boldsymbol{\xi}^*$ is the controlled acceleration, $\boldsymbol{\xi}^{des}$ is a desired acceleration term (also called feed-forward term), $\boldsymbol{e}$ and $\dot{\boldsymbol{e}}$ represent the current pose error and its derivative. They are not explicitly defined as they are representation dependant. If the task is expressed in the joint space then $\boldsymbol{\xi}^* = \ddot{\boldsymbol{q}}^*$ and $\boldsymbol{e} = \left(\boldsymbol{q}^{des} - \boldsymbol{q}\right)$ with $\boldsymbol{q}^{des} \in \mathbb{R}^n$ the joint position to reach obtained from the trajectory generation algorithm. If the task is expressed in the operational space then $\boldsymbol{\xi}^* = \dot{\boldsymbol{v}}^*$ and $\boldsymbol{e} = \left(\boldsymbol{X}^{des} - \boldsymbol{X}\right)$ [4]. $\boldsymbol{X} \in SO(3) \times \mathbb{R}^3$ and $\boldsymbol{X}^{des} \in SO(3) \times \mathbb{R}^3$ respectively represent the robot current and desired end-effector position and orientation in the operational space. The robot orientation representation

[4] In $\mathbb{R}^3$, $\boldsymbol{a} - \boldsymbol{b}$ represents the difference between $\boldsymbol{a}$ and $\boldsymbol{b} \in \mathbb{R}^3$. In $SO(3)$ it represents the rotation axis necessary to rotate from a to $b \in SO(3)$

is not unique. This orientation can be represented using rotation matrices, a sequence of rotations around independent angles (Euler angles, roll-pitch-yaw angles) or unit quaternions. Quaternion representation is often preferred since it removes the representation singularities inherent to angle representations and for its computational efficiency. In the remaining of this document, the unit quaternions are used to represent the robot orientation.

The feed-forward term, $\boldsymbol{\xi}^{des}$, represents the theoretical acceleration that is required to correctly perform the task. In practice, the robot model is subject to uncertainties and the PID controller corrects the possible modelling errors. To that extent, the proportional and derivative terms act as a virtual-spring damper. K_p and K_d respectively represent the stiffness and damping of the controller. The farther the robot is from its desired position, the more acceleration the controller will require to correct the error. The integral term is used to correct the accumulation of errors and cancel any static error.

PID controllers are designed to correct small errors. However, when interacting with the robot, the tracking errors with the desired robot position may greatly increase. This is why in practice, the integral term must be treated carefully when physical interaction with the robot can happen. Indeed, at each time step, the integral term computes more important accelerations to reject the perturbation. If the desired acceleration reaches a dangerous level, the robot may apply important wrenches to correct the tracking errors. If the contact breaks, the commanded acceleration can be important and the release of energy can also be dangerous. One way to prevent such behaviour is to saturate the integral term using anti-windup techniques [Bohn and Atherton, 1995].

2.2.3 Task expression

Equation (2.11) describes the generic formulation of a task. This task can be expressed according to different control variables such as the robot joint acceleration or torque.

Acceleration task An acceleration task relates the robot joint accelerations to the derivative of a twist attached to the system expressed either in the Cartesian space or in joint space. In a general form, this task can be written as

$$\begin{aligned} \boldsymbol{T}(\boldsymbol{\xi}) &= \left\| \boldsymbol{\xi}^* - \boldsymbol{\xi} \right\|_2^2 \\ &= \left\| \boldsymbol{\xi}^* - \left(\dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \dot{\boldsymbol{q}} + J(\boldsymbol{q}) \ddot{\boldsymbol{q}} \right) \right\|_2^2 . \end{aligned} \tag{2.18}$$

$J(\boldsymbol{q})$ and $\dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ are built accordingly to the space chosen considered for $\boldsymbol{\xi}$: in the Cartesian space, $J(\boldsymbol{q})$ is the robot natural Jacobian expressed at some point of interest, in the articular space $J(\boldsymbol{q})$ is the identity matrix.

From Equation (2.18), it can be seen that the objective function is expressed linearly with respect to the control variable $\ddot{\boldsymbol{q}}$ and can thus be used in a quadratic programming formulation.

Torque task When working in collaboration with a robot, one may need to control the interactions between the robot and its environment. In this case, expressing the task with relation to the commanded torque is recommended. A torque task can be described as an acceleration task accounting for the robot equation of motion (Equation (2.1)). Two methods exist to express the robot task relatively to its joint torques.

The first method uses the properties of the optimization formulation to define an equality constraint including the equation of motion. In the operational space, the QP problem is formulated as

$$\begin{aligned} \boldsymbol{\tau}^{opt} = \underset{\boldsymbol{\tau}, \ddot{\boldsymbol{q}}}{\arg\min} \quad & \left\| \dot{\boldsymbol{v}}^* - \left(J(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \right) \right\|_2^2 \\ \text{s.t.} \quad & M(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} \end{aligned} \tag{2.19}$$

This ensures that the solution found by the QP solver satisfies the equation of motion. The result of the optimization, $\ddot{\boldsymbol{q}}^{opt} \in \mathbb{R}^n$, is then used in the equation of motion to find the optimal torque, $\boldsymbol{\tau}^{opt} \in \mathbb{R}^n$, to actuate the robot.

Another approach is to directly express the task as a function of the joint torque such that

$$\boldsymbol{T}(\boldsymbol{\tau}) = \left\| \dot{\boldsymbol{\xi}}^* - J(\boldsymbol{q})M^{-1}(\boldsymbol{q}) \left(\boldsymbol{\tau} - \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{g}(\boldsymbol{q}) \right) - \dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \right\|_2^2. \tag{2.20}$$

The two formulations are equivalent and lead to the same result. The first formulation may however increase the computation time as it introduces an additional constraint and an additional optimization variable.

2.3 Constraints in quadratic programming

The robot is subject to a certain set of constraints imposed by design or by the law of physics. Constraints are physical limitations that cannot be violated. The equation

of motion is an example of an equality constraint imposed on the robot. Joint actuation limits (see Equation (2.2) to (2.4)) are intrinsic inequality constraints applied on the robot. It is also possible to consider the robot dangerousness as a constraint. As presented in Section 1.6, the robot variation of kinetic energy can be used to define a robot dangerousness. By expressing this variation of energy as a function of the robot actuation torque, it is possible to constrain the solution computed by the QP solver so that the robot does not reach a dangerous amount of energy.

This section describes the expression of different constraints acting on a n dof robot inside a quadratic programming problem. The second part of this section is focused on the expression of a constraint preventing the robot from being dangerous in case of unwanted contact.

2.3.1 Intrinsic constraints

The robot intrinsic constraints should always be taken into account to determine commands that are feasible by the robot. This section provides the expression of the three intrinsic constraints mentioned in Equations (2.2) to (2.4).

Torque limits The torque limits are directly linked to the control variable. The expression of the corresponding constraint is straightforward:

$$\boldsymbol{\tau}^{min} \leq \boldsymbol{\tau}_{k+1} \leq \boldsymbol{\tau}^{max}. \tag{2.21}$$

$\boldsymbol{\tau}_{k+1} \in \mathbb{R}^n$ represents the torque solution found by the QP solver at the next discrete time step $k+1$. $\boldsymbol{\tau}^{min}$ and $\boldsymbol{\tau}^{max} \in \mathbb{R}^n$ are torque actuation limits imposed by the actuators.

According to the formulation of the quadratic programming problem presented in Equation (2.12), the torque constraint can be formulated as

$$\begin{aligned} \boldsymbol{lb}_{\boldsymbol{\tau}} \leq A_{\boldsymbol{\tau}_{k+1}} \boldsymbol{\tau} \leq \boldsymbol{ub}_{\boldsymbol{\tau}} \\ \text{with } A_{\boldsymbol{\tau}} = I_n, \\ \boldsymbol{lb}_{\boldsymbol{\tau}} = \boldsymbol{\tau}^{min}, \\ \boldsymbol{ub}_{\boldsymbol{\tau}} = \boldsymbol{\tau}^{max}. \end{aligned} \tag{2.22}$$

Joint velocity limits The actuators joint velocity limits are defined as

$$\dot{\boldsymbol{q}}^{min} \leq \dot{\boldsymbol{q}}_{k+1}(\boldsymbol{\tau}) \leq \dot{\boldsymbol{q}}^{max}; \tag{2.23}$$

with $\dot{\boldsymbol{q}}_{k+1} \in \mathbb{R}^n$ the robot joint velocity at the next control time step, $\dot{\boldsymbol{q}}^{min}$ and $\dot{\boldsymbol{q}}^{max} \in \mathbb{R}^n$ the joint velocity limits. To express $\dot{\boldsymbol{q}}_{k+1}$ as a function of $\boldsymbol{\tau}$, it must be described in its discrete form using a first order Taylor expansion

$$\dot{\boldsymbol{q}}_{k+1}(\boldsymbol{\tau}) = \dot{\boldsymbol{q}}_k + \ddot{\boldsymbol{q}}_k(\boldsymbol{\tau})dt. \tag{2.24}$$

With dt an integration time sufficiently small to validate the Taylor expansion. Using the equation of motion the inequality constraint can be linked to the optimization variable, $\boldsymbol{\tau}$, such that

$$\dot{\boldsymbol{q}}^{min} \leq \dot{\boldsymbol{q}}_k + M_k^{-1}\left(\boldsymbol{\tau} - \boldsymbol{g}_k - \boldsymbol{n}_k\right) dt \leq \dot{\boldsymbol{q}}^{max}. \tag{2.25}$$

For the sake of clarity, $M_k = M(\boldsymbol{q}_k)$, $\boldsymbol{g}_k = \boldsymbol{g}(\boldsymbol{q}_k)$ and $\boldsymbol{n}_k = \boldsymbol{n}(\boldsymbol{q}_k)$. Equation (2.25) can finally be written in the form

$$\boldsymbol{lb}_{\dot{\boldsymbol{q}}} \leq A_{\dot{\boldsymbol{q}}}\boldsymbol{\tau}_{k+1} \leq \boldsymbol{ub}_{\dot{\boldsymbol{q}}} \tag{2.26}$$

$$\begin{aligned} \text{with } A_{\dot{\boldsymbol{q}}} &= M_k^{-1}, \\ \boldsymbol{lb}_{\dot{\boldsymbol{q}}} &= \frac{\dot{\boldsymbol{q}}^{min} - \dot{\boldsymbol{q}}_k}{dt} + M_k^{-1}\left(\boldsymbol{g}_k + \boldsymbol{n}_k\right), \\ \boldsymbol{ub}_{\dot{\boldsymbol{q}}} &= \frac{\dot{\boldsymbol{q}}^{max} - \dot{\boldsymbol{q}}_k}{dt} + M_k^{-1}\left(\boldsymbol{g}_k + \boldsymbol{n}_k\right). \end{aligned}$$

Joint position limits The joint limits are defined as

$$\boldsymbol{q}^{min} \leq \boldsymbol{q}_{k+1}(\boldsymbol{\tau}) \leq \boldsymbol{q}^{max}. \tag{2.27}$$

With $\boldsymbol{q}_{k+1} \in \mathbb{R}^n$ the robot joint position at the next control time step, $\boldsymbol{q}^{min}$ and $\boldsymbol{q}^{max} \in \mathbb{R}^n$ the joint position limits. Using a second order Taylor expansion and the equation of motion, this constraint can also be expressed as a function of $\boldsymbol{\tau}$ such that

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + \dot{\boldsymbol{q}}_k dt + \frac{dt^2}{2} M_k^{-1}\left(\boldsymbol{\tau} - \boldsymbol{g}_k - \boldsymbol{n}_k\right). \tag{2.28}$$

Equation (2.28) can also be written in the form

$$\boldsymbol{lb_q} \leq A_{\boldsymbol{q}} \boldsymbol{\tau}_{k+1} \leq \boldsymbol{ub_q} \tag{2.29}$$

$$\begin{aligned} \text{with } A_{\boldsymbol{q}} &= M_k^{-1}, \\ \boldsymbol{lb_q} &= \frac{2}{dt^2}\left(\boldsymbol{q}^{min} - \boldsymbol{q}_k - \dot{\boldsymbol{q}}_k dt\right) + M_k^{-1}\left(\boldsymbol{g}_k + \boldsymbol{n}_k\right), \\ \boldsymbol{ub_q} &= \frac{2}{dt^2}\left(\boldsymbol{q}^{max} - \boldsymbol{q}_k - \dot{\boldsymbol{q}}_k dt\right) + M_k^{-1}\left(\boldsymbol{g}_k + \boldsymbol{n}_k\right). \end{aligned}$$

The choice of the integration time step, dt, in Equation (2.25) and (2.28) is not a trivial question. This integration time can be associated to a horizon of time before the system "sees" the bound on the constraints. If dt is too small, the robot will have less time to react. For example, if a robot goes close to a joint limit too fast, a small dt will mean that it will have to use important braking torques to avoid the constraints. This can lead to jerk inputs that are not feasible by the actuators. Larger value will tend to produce more conservative torques that does not allow using the full acceleration capacities of the robot. This is because the deceleration capacities of the robot are not taken into account in the constraint formulation [Rubrecht et al., 2010]. In his work, A. Meguenani proposes algorithms to determine the correct integration time according to the deceleration capabilities of each joint [Meguenani, 2018]. While efficient, these algorithms are computationally expensive and only work in joint space.

2.3.2 Constraints related to safety

Apart from intrinsic constraints, a robot can also be subject to other constraints related to its environment. Safety should be considered as a constraint that must be strictly respected. This section details different constraint formulations intended to improve an operator safety in the vicinity of a robot.

2.3.2.1 Constraint on the robot workspace

In a first attempt to prevent collisions between a robot and its environment, one can prevent the robot from entering in specific areas of the operational space. The implementation of a virtual fixture inside a QP solver is experimented in [Kapoor et al., 2006] and yields promising results. Usually virtual fixtures are expressed as tasks with a spring-damper implementation that pushes the robot away from the fixtures boundaries [Joly

and Andriot, 1995]. The expression of the constraint does not implement such spring-damper behaviour. Instead, the QP solver computes solutions that keep the robot inside the boundaries.

Let us consider a robot where motions are restrained to a specific workspace. Its limits are included between $\boldsymbol{X}^{min}$ and $\boldsymbol{X}^{max} \in \mathbb{R}^3$. A constraint on the robot Cartesian position is expressed as

$$\boldsymbol{X}^{min} \leq \boldsymbol{X}_{k+1}(\boldsymbol{\tau}) \leq \boldsymbol{X}^{max}. \tag{2.30}$$

With $\boldsymbol{X}_{k+1} \in \mathbb{R}^3$, the end-effector position in the operational space at the next control time step. This constraint must be expressed as a function of the joint torques. Using a Taylor expansion one can state that

$$\boldsymbol{X}_{k+1}(\boldsymbol{\tau}) = \boldsymbol{X}_k + \boldsymbol{v}_k dt + \frac{1}{2}\dot{\boldsymbol{v}}_k(\boldsymbol{\tau})dt^2. \tag{2.31}$$

Using the equation of motion of the robot and the derivative of $\boldsymbol{v} = J\dot{\boldsymbol{q}}$, it is possible to link the robot Cartesian position with the joint torques and express the constraint as

$$\boldsymbol{X}^{min} \leq \boldsymbol{X}_k + J_k\dot{\boldsymbol{q}}_k dt + \frac{dt^2}{2}\left(\dot{J}_k\dot{\boldsymbol{q}}_k + J_k M_k^{-1}\left(\boldsymbol{\tau} - \boldsymbol{g}_k - \boldsymbol{n}_k\right)\right) \leq \boldsymbol{X}^{max} \tag{2.32}$$

Finally, this constraint can be put in the form

$$\boldsymbol{lb}_{\boldsymbol{X}} \leq A_{\boldsymbol{X}}\boldsymbol{\tau}_{k+1} \leq \boldsymbol{ub}_{\boldsymbol{X}} \tag{2.33}$$

$$\begin{aligned}
\text{with } A_{\boldsymbol{X}} &= J_k M_k^{-1} \\
\boldsymbol{lb}_{\boldsymbol{X}} &= \frac{2}{dt^2}\left(\boldsymbol{X}^{min} - \boldsymbol{X}_k - J_k\dot{\boldsymbol{q}}_k dt\right) - \dot{J}_k\dot{\boldsymbol{q}}_k + J_k M_k^{-1}\left(\boldsymbol{n}_k + \boldsymbol{g}_k\right) \\
\boldsymbol{ub}_{\boldsymbol{X}} &= \frac{2}{dt^2}\left(\boldsymbol{X}^{max} - \boldsymbol{X}_k - J_k\dot{\boldsymbol{q}}_k dt\right) - \dot{J}_k\dot{\boldsymbol{q}}_k + J_k M_k^{-1}\left(\boldsymbol{n}_k + \boldsymbol{g}_k\right)
\end{aligned}$$

This constraint can be used to dynamically constrain the robot workspace. Consider a human working close to the robot with its position being recorded reactively. The position of the virtual wall can be set to follow the human motions. This constraint will force the quadratic programming solver to find torque inputs keeping the robot at

a safe distance from the operator. If no solution is found, an emergency stop signal can be triggered to stop the robot before a collision occurs. The distance between the robot and the wall can be computed accounting for the deceleration capabilities of the robot to ensure that when the emergency stop is triggered the robot inertia does not pull the robot towards the human.

2.3.2.2 Expression of a kinetic energy constraint

As stated in Section 1.3, safety for the environment of the robot requires a limitation of its kinetic energy. Indeed, this energy would, for example, have to be dissipated in case of an impact through deformation (and potentially damage) of the robot/environment pair while in contact. Assuming that there are two bodies colliding, the dissipated energy during collision is a function of both bodies mass and velocity. However, little information is available about the obstacle colliding with the robot. On the other hand, the kinetic energy of the robot effector at time t can be written

$$e_c\left(t\right) = \frac{1}{2}\boldsymbol{v}^T\left(t\right)\Lambda\left(\boldsymbol{q}\left(t\right)\right)\boldsymbol{v}\left(t\right) \tag{2.34}$$

Given $\dot{\boldsymbol{v}}^c(t)$ the controlled acceleration, the velocity of the end-effector at time $t+T$ can be written

$$\boldsymbol{v}\left(t+T\right) = \boldsymbol{v}(t) + \int_t^{t+T} \dot{\boldsymbol{v}}^c(t)dt. \tag{2.35}$$

Considering a discrete time controller of control period Δt such that $T = n\Delta t$ and $t = k\Delta t$, Equation (2.35) can be written in its discrete form

$$\boldsymbol{v}_{k+n} = \boldsymbol{v}_k + \sum_{i=k}^{k+n-1} \dot{\boldsymbol{v}}_i^c \Delta t. \tag{2.36}$$

The variation of kinetic energy between the time t_k and $t_k + T$ can thus be written in the following discrete form

$$\begin{aligned}
\Delta e_c &= e_{c,k+n} - e_{c,k} \\
&= \tfrac{1}{2}\boldsymbol{v}_{k+n}^T \Lambda_{k+n} \boldsymbol{v}_{k+n} - \tfrac{1}{2}\boldsymbol{v}_k^T \Lambda_k \boldsymbol{v}_k \\
&= \tfrac{1}{2}\left(\boldsymbol{v}_k + \sum_{i=k}^{k+n-1} \dot{\boldsymbol{v}}_i^c \Delta t\right)^T \Lambda_{k+n} \left(\boldsymbol{v}_k + \sum_{i=k}^{k+n-1} \dot{\boldsymbol{v}}_i^c \Delta t\right) - \tfrac{1}{2}\boldsymbol{v}_k^T \Lambda_k \boldsymbol{v}_k
\end{aligned} \tag{2.37}$$

Assuming that the modification of the robot configuration between the control instants k and $k+n$ is small enough, the variation of the robot operational inertia can be neglected ($\Lambda_{k+n} \approx \Lambda_k$) and Equation (2.37) can be rewritten

$$\Delta e_c \approx \left(\boldsymbol{v}_k \Delta t + \frac{1}{2} \sum_{i=k}^{k+n-1} \dot{\boldsymbol{v}}_i^c \Delta t^2 \right)^T \Lambda_k \sum_{i=k}^{k+n-1} \dot{\boldsymbol{v}}_i^c. \tag{2.38}$$

Note that, $\left(\boldsymbol{v}_k \Delta t + \frac{1}{2} \sum_{i=k}^{k+n-1} \dot{\boldsymbol{v}}_i^c \Delta t^2 \right)$ represents the expected variation of operational position of the end-effector given the controlled acceleration trajectory $\dot{\boldsymbol{v}}_{k \to k+n}^c$ over the control window $[k; k+n]$ and an initial state $\{\boldsymbol{X}_k; \boldsymbol{v}_k\}$, where $\boldsymbol{X}_k \in \mathbb{R}^6$ is the end effector position at time k. This pose variation is noted $\Delta \boldsymbol{x}_{k \to k+n}$. On the other hand, $\Lambda_k \sum_{i=k}^{k+n-1} \dot{\boldsymbol{v}}_i^c$ represents the sum of the equivalent control wrenches over the control window $[k; k+n]$. This sum can be written $\sum_{i=k}^{k+n-1} \boldsymbol{f}_i^c$ where $\boldsymbol{f}_i^c = \Lambda_k \dot{\boldsymbol{v}}_i^c$. Equation (2.38) thus represents an approximation of the work energy theorem introduced in Equation (1.9) over the control window $[k; k+n]$ with $\Delta e_c = \Delta \boldsymbol{x}_{k \to k+n} \sum_{i=k}^{k+n-1} \boldsymbol{f}_i^c$.

At the next time step, Equation (2.38) leads to

$$e_{c,k+1}(\boldsymbol{\tau}) = e_{c,k} + \left(\boldsymbol{v}_k \Delta t + \frac{1}{2} \dot{\boldsymbol{v}}_k^c(\boldsymbol{\tau}) \Delta t^2 \right)^T \Lambda_k \dot{\boldsymbol{v}}_k^c(\boldsymbol{\tau}). \tag{2.39}$$

This expression of the kinetic energy is quadratic with relation to $\boldsymbol{\tau}$. Optimization problems featuring quadratic constraint functions are called quadratically constrained quadratic program (QCQP). In practice QCQP problem are NP-hard and thus computationally more expensive to solve than linear quadratic programming problems which are solvable in polynomial time. Since this constraint involves safety and should be solved in real-time, a solution to express this constraint linearly with relation to the control torque is preferred. It can be noted that in practice, $\dot{\boldsymbol{v}}_k^c$ will not be actually reached on the real system and the expected acceleration is rather $\dot{\boldsymbol{v}}_k^*$. This desired acceleration is known at instant k and $\Delta \boldsymbol{x}_{k \to k+n}$ can be interpreted as the expected pose variation that would be induced if this acceleration is actually achieved.

This leads to a modified expression of the kinetic energy at the next time step:

$$e_{c,k+1}(\boldsymbol{\tau}) = e_{c,k} + \left(\boldsymbol{v}_k \Delta t + \frac{1}{2} \dot{\boldsymbol{v}}_k^* \Delta t^2 \right)^T \Lambda_k \dot{\boldsymbol{v}}_k^c(\boldsymbol{\tau}). \tag{2.40}$$

Equation (2.40) shows the possibility to express the kinetic energy at the next time step as a function of the control torque using the equation of motion:

$$e_{c,k+1}(\boldsymbol{\tau}) = e_{c,k} + \left(\boldsymbol{v}_k \Delta t + \frac{1}{2}\dot{\boldsymbol{v}}_k^* \Delta t^2\right)^T \Lambda_k \left(J_k M_k^{-1} \left(\boldsymbol{\tau} - \boldsymbol{n}_k - \boldsymbol{g}_k\right) + \dot{J}_k \dot{\boldsymbol{q}}_k\right). \tag{2.41}$$

The safety constraint is then expressed as

$$e_{c,k+1}(\boldsymbol{\tau}) \leq e_c^{lim}. \tag{2.42}$$

which can be written as

$$\boldsymbol{lb}_{e_c} \leq A_{e_c} \boldsymbol{\tau}_{k+1} \leq \boldsymbol{ub}_{e_c} \tag{2.43}$$

$$\begin{aligned} \text{with } A_{e_c} &= \boldsymbol{\Delta x}_k^T \Lambda_k J_k M_k^{-1} \\ \boldsymbol{lb}_{e_c} &= 0 \\ \boldsymbol{ub}_{e_c} &= e_c^{lim} - e_{c,k} + \boldsymbol{\Delta x}_k^T \Lambda_k \left(\dot{J}_k \dot{\boldsymbol{q}}_k - J_k M_k^{-1} \left(\boldsymbol{g}_k + \boldsymbol{n}_k\right)\right) \end{aligned}$$

Equation (2.43) is linear with respect to the control torque and can be inserted inside a quadratic programming problem. The remaining of this section depicts how this constraint can be used in the transient and quasi-static contact cases.

Transient contact The formulation of the kinetic energy constraint in Equation (2.43) can be directly used to limit the energy that can be dissipated during a transient contact. e_c^{lim} can be fixed according to the limits recommended by ISO TS 15066 considering the human body parts that can enter in contact with the robot.

Quasi-static contact During quasi-static contact, when contact is established, at instant $k = k_0$, Equation (2.40) can be written

$$e_{c,k_0+1}(\boldsymbol{\tau}) = e_{c,k_0} + \left(\boldsymbol{v}_{k_0} \Delta t + \frac{1}{2}\dot{\boldsymbol{v}}_{k_0}^* \Delta t^2\right)^T \Lambda_{k_0} \dot{\boldsymbol{v}}_{k_0}^c(\boldsymbol{\tau}). \tag{2.44}$$

Since the robot is pushing against an immobile obstacle, $\boldsymbol{v}_{k_0} = 0$ m/s and $e_{c,k_0} = 0$ J. Equation 2.44 can be simplified such that

$$e_{c,k_0+1}(\boldsymbol{\tau}) = \frac{1}{2}\left(\dot{\boldsymbol{v}}_{k_0}^* \Delta t^2\right)^T \boldsymbol{f}_{k_0}^c(\boldsymbol{\tau}) \tag{2.45}$$

with $\boldsymbol{f}^c_{k_0} = \Lambda_{k_0}\dot{\boldsymbol{v}}^c_{k_0}$ the robot pushing force at the instant k_0. Equation (2.45) states that the efforts applied by the robot during a quasi-static contact are a function of its kinetic energy and the operational acceleration obtained from the PID controller.

As long as the perturbation remains, the PID controller integrates the error between the current robot position and the desired one. As $\dot{\boldsymbol{v}}^*_k$ increases, $e_{c,k+1}$ in Equation (2.45) also increases until it reaches the constraint at instant $k = k_1$ (with $k_1 \geq k_0$) and

$$e_{c,k_1+1}(\boldsymbol{\tau}) = e_c^{lim} = \frac{1}{2}\left(\dot{\boldsymbol{v}}^*_{k_1+1}\Delta t^2\right)^T \boldsymbol{f}^c_{k_1}(\boldsymbol{\tau}). \tag{2.46}$$

Even though the robot kinetic energy limit is reached, the PID controller keeps on integrating the tracking error resulting from the interaction. As mentioned in Section 2.2.2, the output of the PID controller is saturated. Therefore, after a certain time $k = k_2$ (with $k_2 \geq k_1$), the desired robot acceleration will reach a limit, $\dot{\boldsymbol{v}}^{*,sat}$, so that

$$e_{c,k_2+1}(\boldsymbol{\tau}) = e_c^{lim} = \frac{1}{2}\left(\dot{\boldsymbol{v}}^{*,sat}\Delta t^2\right)^T \boldsymbol{f}^c_{k_2}(\boldsymbol{\tau}). \tag{2.47}$$

e_c^{lim} being defined by the transient contact case and knowing the PID saturation, $\dot{\boldsymbol{v}}^{*,sat}$, one can compute an integration period Δt to indirectly limit the wrenches applied by the robot so that $\boldsymbol{f}^c_{k_2} = \boldsymbol{f}^{lim}$. From Equation (2.47), Δt can be chosen so that

$$\Delta t = \sqrt{\frac{2e_c^{lim}}{(\dot{\boldsymbol{v}}^{*,sat})^T\boldsymbol{f}^{lim}}}. \tag{2.48}$$

$\boldsymbol{f}^{lim}$ can for example be chosen based on the maximum applicable efforts for different body parts recommended by ISO Norm 15066. The choice of Δt is highly dependent on the choice of the PID saturation $\dot{\boldsymbol{v}}^{*,sat}$. This saturation term is defined during the tuning of the PID controller. However, there is no straightforward solution to select a specific saturation value. This saturation is related to the maximal Cartesian acceleration achievable by the robot. This Cartesian acceleration is usually not given by the robot manufacturer and must be estimated. Another solution is to consider a saturation related to the maximal torque achievable by the robot actuators. This raises some stability issues that can be dealt with as presented in [Alvarez-Ramirez et al., 2008] but involves a variable saturation and thus a variable Δt. This analysis on the robot kinetic energy can help to choose a saturation value. Indeed, $\dot{\boldsymbol{v}}^{*,sat}$ should not be set too low, otherwise the PID saturation might occur before the robot kinetic energy limit is reached ($k_2 \leq k_1$)). In such case, the kinetic energy constraint will never be activated and the robot will be sub-optimally used.

By choosing an adequate saturation value, the integration period, Δt, can be defined according to Equation (2.48) so that the kinetic energy constraint restrains the robot from exerting a force greater than $\boldsymbol{f}^{lim}$ from the instant k_2 and until the contact is released.

2.3.2.3 Generalisation for any point of interest

The formulation of the kinetic energy constraint is expressed at the robot end-effector since it is usually the point yielding the most kinetic energy. However, one may want to constrain the kinetic energy of any point of the robot. For a 7 dof robot the elbow joint can also yield an important amount of energy that can be dangerous in case of unwanted contact. One may also want to constrain the kinetic energy of the closest point to an obstacle. Several points could also be constrained at the same time, although it would require more computation time for the quadratic programming solver.

To that extent, Equation (2.39) can be extended to any point on the robot such that

$$e_{c,k+1}\big|_p = e_{c,k}\big|_p + \left(\boldsymbol{v}_k\big|_p \Delta t + \frac{1}{2}\dot{\boldsymbol{v}}_{\boldsymbol{k}}^{\boldsymbol{c}}\big|_p \Delta t^2\right)^T \Lambda_k \dot{\boldsymbol{v}}_{\boldsymbol{k}}^{\boldsymbol{c}}\big|_p, \tag{2.49}$$

where the $\big|_p$ represents a point on the robot. The relationship between Equation (2.49) and the optimization variable is still quadratic with relation to τ. Since the desired acceleration, $\dot{\boldsymbol{v}}_k^*$, is expressed at the control point, the simplifications realised in Equation (2.40) cannot be used. This situation has not been studied in this work.

2.4 Redundancy and quadratic programming

A robot that possess more degrees of freedom than the ones required to execute a task is said to be redundant with relation to the task. It means that there is an infinity of solutions to accomplish the task. This is an interesting property which enhances the mobility of the robot, allowing it to reach objects in cluttered environments or to realise secondary tasks. However, from a control point of view, a redundant robot presents many challenges. Choosing a specific configuration yielding an interesting additional robot behaviour is not always simple.

This section proposes solutions to control redundancy using quadratic programming methods. It details some possible implementations and presents a control solution minimizing the robot perceived mass. As a reminder, this mass is linked to the robot kinetic energy (see Equation (1.5)) and to the force exerted by the robot at the moment of

impact (see Equation (1.6)). Minimizing it can reduce the robot dangerousness during a contact.

2.4.1 The regularisation task in convex optimization methods

A way to determine a unique control input for a robot that is redundant relatively to its tasks is to add a task requiring every available degree of freedom. This task is called a regularization task and is noted $R(\boldsymbol{x})$. This regularisation can be realised automatically by the quadratic solver or implemented to perform specific secondary tasks. This task should not influence the results of the main tasks. To that extent, the two hierarchization techniques presented in Section 2.1.2.1 can be used. In a weighted prioritization strategy, the QP formulation becomes

$$\begin{aligned} \boldsymbol{x^{opt}} = \quad & \arg\min_{\boldsymbol{x}} \quad \sum_{i=1}^{n_t} \|E_i\boldsymbol{x} - \boldsymbol{f}_i\|^2_{W_i} + \|E_0\boldsymbol{x} - \boldsymbol{f_0}\|^2_{W_0} \\ & \text{s.t.} \quad A\boldsymbol{x} \leq \boldsymbol{b} \\ & \qquad C\boldsymbol{x} = \boldsymbol{d} \end{aligned} . \tag{2.50}$$

With $R(\boldsymbol{x}) = \|E_0\boldsymbol{x} - \boldsymbol{f_0}\|^2_{W_0}$, the regularization task, $E_0 \in \mathbb{R}^n$, $\boldsymbol{f_0} \in \mathbb{R}^n$ and $W_0 = \epsilon I_{n\times n}$, with $\epsilon \ll 1$ a sufficiently small weight so that the regularization task does not interfere with the main tasks.

The next section details some possible implementation of the regularization task for a torque controlled robot.

2.4.2 Torque minimization task

The simplest regularization task consists in minimizing the optimization variable. For a torque controlled robot the regularization task becomes

$$R(\boldsymbol{\tau}) = \|\boldsymbol{\tau}\|^2_{W_0} . \tag{2.51}$$

When seeking a solution for this problem, the QP solver will find the torque solution minimizing the overall joint torques.

2.4.3 Gravity compensation task

The previous regularization task prevents physical interaction with the robot. There is a single configuration realizing the task while minimizing the joint torques. Hence, it is not possible to take advantage of the additional degrees of freedom of the robot to physically interact with it. This can be interesting for example to keep the servoing of a point while performing motion in the null-space of the main task Jacobian by physically manipulating the robot. Such interaction can be useful in applications where the robot is sharing its workspace with a human (see Table 1.1). To enable such type of interaction, another regularization task can be written as

$$R(\boldsymbol{\tau}) = \|\boldsymbol{\tau} - \boldsymbol{g}(\boldsymbol{q})\|^2_{W_0} . \tag{2.52}$$

This regularization function minimizes the difference between the computed torque and the gravity induced external torque. It results that the combination of joints that does not contribute to the resolution of the main tasks are compensating for gravity.

2.4.4 Posture task

When using a robotic manipulator, one may want to keep the robot in a reference position. From the operator point of view, this can ease the prediction of the robot internal motion. A regularization task can be added to implement a spring-damper behaviour at the joint torque level[5] that keeps the robot towards a desired joint configuration. At the torque level, the servoing of this configuration can be expressed using a proportional-derivative controller such that

$$\boldsymbol{\tau}^* = K_p \left(\boldsymbol{q}^{des} - \boldsymbol{q} \right) - K_d \dot{\boldsymbol{q}} \tag{2.53}$$

with $\boldsymbol{\tau}^* \in \mathbb{R}^n$ a desired torque pushing the robot towards the desired joint configuration $\boldsymbol{q}^{des}$.

The regularisation task is then expressed as

$$R(\boldsymbol{\tau}) = \|\boldsymbol{\tau} - \boldsymbol{\tau}^*\|^2_{W_0} . \tag{2.54}$$

[5] A PD controller at the acceleration level could also be defined. However, it would require to use the robot equation of motion to transform this acceleration in terms of torques. This requires to use the robot model which can feature small errors and lead to imprecisions that can be avoided by expressing the controller at the torque level.

2.4.5 Using redundancy to improve safety

In this work, special considerations are taken to use a robot redundancy to modulate its perceived mass in a specific direction. This mass is directly linked to the robot kinetic energy transferred during an impact. Minimizing it should thus improve the robot safety. If the robot motions are planned off-line, it is possible to compute off-line a set of configurations minimizing the robot perceived mass in the direction of motion. However, if the robot motions are updated on-line or if there are obstacles in the robot environment, it may be better to compute on-line new configurations minimizing the robot perceived mass. Furthermore, if a human is present in the robot environment, minimizing the transferred kinetic energy between the robot and human requires to minimize the perceived mass in the direction of the human rather than in the direction of motion.

This section first recalls the definition of a robot perceived mass and its link with the robot configuration. The computation of the robot null-space motion is then presented and exemplified on specific cases. An algorithm to find the perceived mass global minimum off-line using only the robot model and its trajectory is first proposed. A second algorithm to find a local minimum of this perceived mass on-line is then developed. Using external sensors detecting obstacles in the environment, this algorithm can be used to reactively minimize the robot perceived mass in the direction of the closest obstacle.

2.4.5.1 The robot perceived mass

As explained in Section 1.3.2, the robot kinetic energy is a function of its velocity and of its perceived mass. As a reminder, the mass perceived at the tip of the end-effector and projected in a direction $\boldsymbol{u}$ is expressed as

$$m_{\boldsymbol{u}}(\boldsymbol{q}) = \boldsymbol{u}^T \left(J_{lin}(\boldsymbol{q}) M^{-1}(\boldsymbol{q}) J_{lin}^T(\boldsymbol{q}) \right)^{-1} \boldsymbol{u}. \tag{2.55}$$

It can be noted that this mass is a function of the joint configuration. I.D. Walker analyses the influence of the robot configuration on the impulsive forces generated by a collision [Walker, 1994]. It links the robot perceived mass to these impulsive forces and proposes an impact ellipsoid to select the robot configuration yielding the less impact forces. In the work of V. Padois *et al.* , a gradient descent algorithm is proposed to find a robot configuration minimizing its perceived mass and results show the reduction of the impact forces [Padois et al., 2004]. N. Mansfeld *et al.* propose to realise a gradient descent to minimize a robot perceived mass projected in the direction of motion and

use it in a safe velocity controller [Mansfeld et al., 2017]. These papers use analytical inversion methods that project the solution in the null-space of the Jacobian of the robot main task.

The robot perceived mass is directly linked to the robot kinetic energy. As such, there exist a specific robot configuration yielding the minimum kinetic energy. Using the techniques developed in the previously stated works it is then possible to advantageously use robot redundancy to reduce the dangerousness of a robot during a transient contact.

2.4.5.2 Robot null-space motion

Let us consider a n dof robot performing a m dimensional task. To find the joint configurations that do not change the end-effector position, one can compute the null-space base matrix $Z(\boldsymbol{q}) \in \mathbb{R}^{r\times n}$ of $J(\boldsymbol{q}) \in \mathbb{R}^{m\times n}$, i.e. such that $J(\boldsymbol{q})Z(\boldsymbol{q})^T = \mathbb{0}^{m\times r}$, with $r = n - m$. If $J(\boldsymbol{q})$ is full rank and not singular, it is possible to linearly combine the columns of $J(\boldsymbol{q})$ to build a matrix $J'(\boldsymbol{q}) = [J_m(\boldsymbol{q}) \quad J_r(\boldsymbol{q})]$. $J_m(\boldsymbol{q}) \in \mathbb{R}^{m\times m}$ is composed of m independent columns and is thus locally invertible and $J_r(\boldsymbol{q}) \in \mathbb{R}^{m\times r}$ is composed of the remaining columns of $J'(\boldsymbol{q})$ that are not included in $J_m(\boldsymbol{q})$. The null-space base matrix can similarly be decomposed in $Z'(\boldsymbol{q}) = [Z_m(\boldsymbol{q}) \quad Z_r(\boldsymbol{q})]$, with $Z_m \in \mathbb{R}^{r\times m}$ and $Z_r \in \mathbb{R}^{r\times r}$ to be determined. The condition $J(\boldsymbol{q})Z(\boldsymbol{q})^T = \mathbb{0}^{m\times r}$ can be equivalently written $J'(\boldsymbol{q})Z'(\boldsymbol{q})^T = \mathbb{0}^{m\times r}$ and developed such that

$$J_m(\boldsymbol{q})Z_m(\boldsymbol{q})^T + J_r(\boldsymbol{q})Z_r(\boldsymbol{q})^T = \mathbb{0}^{m\times r}. \tag{2.56}$$

A straightforward, solution to Equation (2.56), proposed in [Zghal et al., 1990], consists in choosing $Z_r(\boldsymbol{q}) = I^{r\times r}$ so that

$$Z(\boldsymbol{q}) = [-J_r(\boldsymbol{q})^T J_m(\boldsymbol{q})^{-T} \quad I^{r\times r}] \tag{2.57}$$

However, there exists an infinity of solution to Equation (2.56). Y. C. Chen and I. D. Walker propose to weight $Z(\boldsymbol{q})$ with the determinant of $J_m(\boldsymbol{q})$ leading to a second expression

$$Z(\boldsymbol{q}) = [-J_r(\boldsymbol{q})^T adj(J_m(\boldsymbol{q}))^T \quad det(J_m(\boldsymbol{q}))I^{r\times r}] \tag{2.58}$$

where $adj(J_m(\boldsymbol{q}))$ is the adjoint matrix[6] of $J_m(\boldsymbol{q})$ [Chen and Walker, 1993].

[6]The adjoint of a square matrix is the transpose of its cofactor matrix

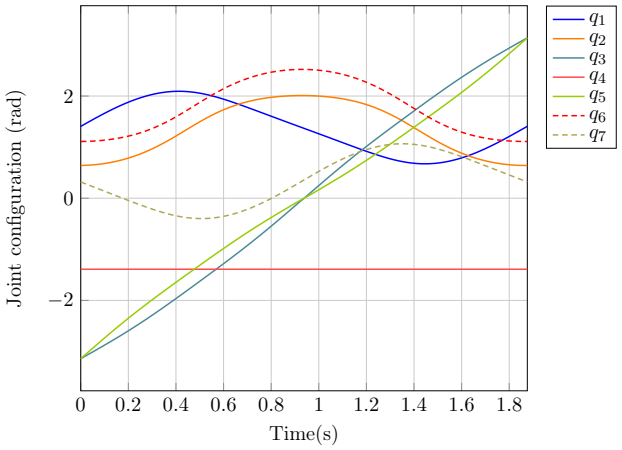


FIGURE 2.2: Joint configuration inducing self motion

For a positioning task requiring the servoing of the robot position and orientation, a 7 dof robot has 1 degree of freedom available. Hence, $Z(\boldsymbol{q})$ becomes a vector, denoted $\boldsymbol{z}(\boldsymbol{q})$. In his book, C. Ott proposes an analytical solution to compute its corresponding null-space velocity vector

$$z_i(\boldsymbol{q}) = (-1)^{n+1} det(J_i(\boldsymbol{q})) \tag{2.59}$$

where $J_i(\boldsymbol{q})$ is $J(\boldsymbol{q})$ with the ith column omitted [Ott, 2008]. Note that this formula only stands when the degree of redundancy is one. The joint configuration that does not induce a modification of the end-effector position in the operational space, $\boldsymbol{q}_{ns} \in \mathbb{R}^n$, can then be computed by integrating $\boldsymbol{z}(\boldsymbol{q})$ over time starting from an initial position, $\boldsymbol{q}_0$, at the time t_0

$$\boldsymbol{q}_{ns}(t) = \boldsymbol{q}_0 + \int_{t_0}^{t} \boldsymbol{z}(\boldsymbol{q}_{ns}(t))dt. \tag{2.60}$$

This integration can be performed until one of the joints has performed a full rotation. With Equation (2.60), it is possible to find every joint configuration keeping the end-effector at the same operational position. These configurations are presented in Figure 2.2. It can be observed that both q_3 and q_5 perform a full rotation during this integration.

2.4.5.3 Finding the perceived mass global minimum

It is possible to compute the global minimum of the robot perceived mass projected in a direction from Equation (2.55) and (2.60). Using Equation (2.55), the perceived mass projected along $\boldsymbol{u}$ as a function of q_3 is plotted in Figure 2.3 a. It shows the evolution of

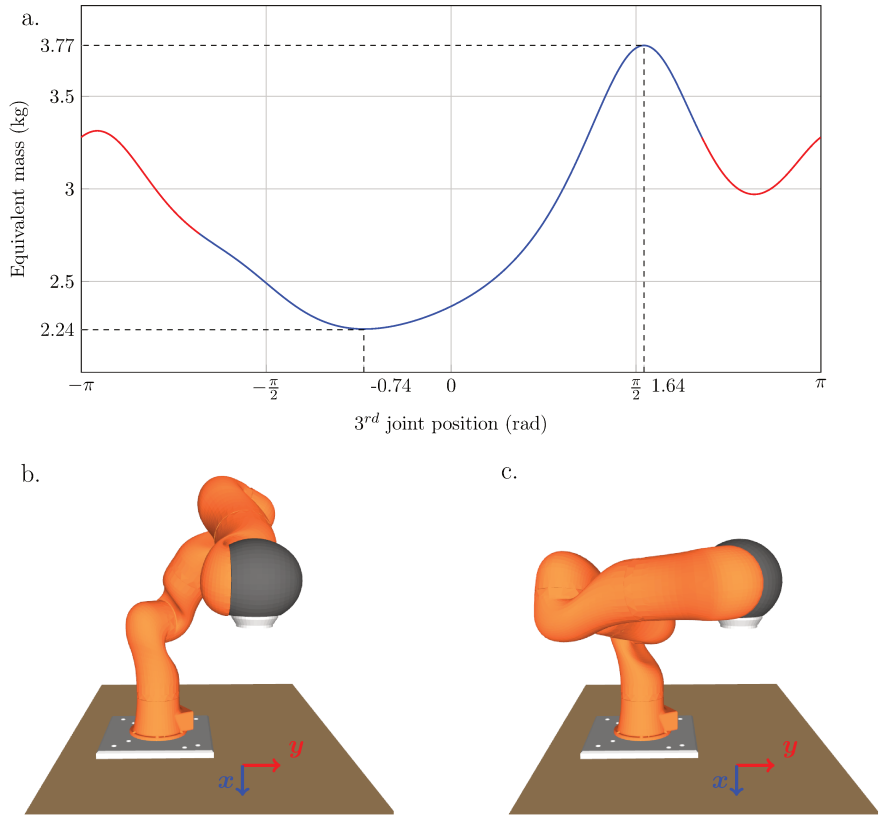


FIGURE 2.3: a. Equivalent mass projected in the y-direction as a function of the robot third joint position. b. Configuration corresponding to the minimal perceived mass in the y-direction. c. Configuration corresponding to the maximal perceived mass in the y-direction.

the perceived mass of a 16 kg KUKA LWR4+ robot during self-motion in the y direction. The red curve corresponds to the configurations that are outside of the joint position limits and cannot be reached by the robot. According to the robot configuration, the perceived mass ranges from 2.24 kg (configuration represented in Figure 2.3 b.) to 3.77 kg (configuration represented in Figure 2.3 c.). It represents a mass reduction of 30% and a similar reduction of the kinetic energy during the impact.

Due to the nature of the computation of $\boldsymbol{q}_{ns}(t)$ it is not possible to obtain an analytical formulation of $m_u(\boldsymbol{q})$. In a first attempt to improve safety, a global minimum of the perceived mass can be determined in the direction of motion. By choosing a small integration step, dt, it is possible to obtain a good knowledge of the evolution of the perceived mass and to pick the configuration corresponding to its minimum. However, finding this configuration cannot be done on-line.

Applying this method off-line on k waypoints along a trajectory gives a good approximation of the configurations to impose along the trajectory to minimize the robot perceived mass along the direction of motion. These configurations can be stored in an array, Q_{ns}^{min}, and associated to an array storing the robot Cartesian position along the

trajectory, X_{arr}^{des}. Given the robot current Cartesian position, an algorithm determines the closest Cartesian position belonging to the trajectory and picks the corresponding configuration minimizing the robot perceived mass along the trajectory. This configuration is then set in the regularization task as a goal configuration, $\boldsymbol{q}_{ns}^{des}$, to reach. This whole process is presented in Algorithm 2.

Algorithm 2: Algorithm to determine the global minimum of the robot perceived mass in the direction of motion

Offline
for *i points on the trajectory* **do**
  Given $X_{arr}^{des}(i)$, compute $Q_{ns}^{min}(i)$ using Equations (2.55) and (2.60)
end
return X_{arr}^{des}, Q_{ns}^{min}

Online
Get the current end-effector position $\boldsymbol{X}$
$\boldsymbol{X}^{min} = X_{arr}^{des}(0)$
for *i points on the trajectory* **do**
  if $\|X_{arr}^{des}(i) - \boldsymbol{X}\| \leq \boldsymbol{X}^{min}$ **then**
    $\boldsymbol{X}^{min} = X_{arr}^{des}$
    $i_{min} = i$
  end
end
return $\boldsymbol{q}_{ns}^{des} = Q_{ns}^{min}(i_{min})$

2.4.5.4 Local perceived mass minimization in the direction of an obstacle

The global minimum computed in the previous section can be interesting if there is no mean to know the position of an obstacle in real-time. The robot kinetic energy is maximal in the direction of motion and as such, the safest approach is to reduce the perceived mass in this direction. However, for applications in a shared workspace as presented in Table 1.1, the robot motions may be updated on-line. In such case, the previous method will give suboptimal results. If the presence of an obstacle can be detected, it is wiser to minimize the perceived mass in its direction. For a simple motion along a line, Algorithm 2 takes several minutes to compute every joint configuration along the trajectory and select, for each way-point, the configuration minimizing the robot perceived mass in a direction. If a new configuration must be defined in real-time, it might be better to look for a local minimum of the robot perceived mass. This is the goal of this subsection.

$\boldsymbol{\nabla m_u}(\boldsymbol{q})$ represents the gradient of the perceived mass at a given time such that

$$\boldsymbol{\nabla m_u}(\boldsymbol{q}) = \frac{\partial m_{\boldsymbol{u}}}{\partial \boldsymbol{q}} \tag{2.61}$$

$\boldsymbol{\nabla m_u}(\boldsymbol{q})$ provides a local estimation of the shape of $m_{\boldsymbol{u}}(\boldsymbol{q})$. Gradient descent algorithms can be used to determine the configuration leading to a local minimum of $m_{\boldsymbol{u}}$. Starting from the current robot configuration, this algorithm moves along the curve $m_{\boldsymbol{u}}(\boldsymbol{q})$ by taking steps proportional to the negative direction of $\boldsymbol{\nabla m_u}(\boldsymbol{q})$. The efficiency of this technique depends on the selected step size. A step too small may increase the number of iterations required to find the optimum solution. A big step size value may induce oscillating behaviours near the optimal solution. Line-search algorithms can be used to determine a correct step size. A new joint configuration in the null-space of the main task Jacobian, $\boldsymbol{q}_{ns}(k+1)$, is determined iteratively through a gradient descent algorithm such that

$$\boldsymbol{q}_{ns}(k+1) = \boldsymbol{q}_{ns}(k) - \alpha \left(I_n - J(\boldsymbol{q})^{+} J(\boldsymbol{q})\right)^T \boldsymbol{\nabla m_u}(\boldsymbol{q}) \Delta t. \tag{2.62}$$

k is a discrete time step, $\alpha < 1$ is a step size. $(I_n - J(\boldsymbol{q})^{+} J(\boldsymbol{q}))$ is a projector on the end-effector Jacobian null-space. This projector ensures that the configuration obtained through the gradient descent does not influence the position of the end-effector. The step size is determined using the Armijo-Wolfe line search algorithm [Wolfe, 1969]. The gradient descent is stopped either when the new solution does not change the previous one by more than a defined threshold ε, *i.e.* when $\|\boldsymbol{q}_{ns}(k+1) - \boldsymbol{q}_{ns}(k)\| < \varepsilon$ or if it reaches a maximum iteration step, k_{max}. The computation time of one iteration of this algorithm is constant. k_{max} can thus be defined to ensure that the algorithm does not take more time than the control loop periodicity to find a solution. Overall, the algorithm used to determine $\boldsymbol{q}_{ns}(k+1)$ is detailed in Algorithm 3.

Algorithm 3: Algorithm to determine the local minimum of the robot perceived mass in the direction $\boldsymbol{u}$

$\boldsymbol{q}_{ns}(0) = \boldsymbol{q}(t)$;
while $k < k_{max}$ *or* $\|\boldsymbol{q}_{ns}(k+1) - \boldsymbol{q}_{ns}(k)\| > \varepsilon$ **do**
  $J^{+}(\boldsymbol{q}_{ns}) = J(\boldsymbol{q}_{ns})^T \left(J(\boldsymbol{q}_{ns}) J(\boldsymbol{q}_{ns})^T\right)^{-1}$;
  $\boldsymbol{\nabla m_u}(\boldsymbol{q}_{ns}(k)) = \frac{\partial m_u(\boldsymbol{q}_{ns})}{\partial \boldsymbol{q}_{ns}}$;
  compute α;
  $\boldsymbol{q}_{ns}(k+1) = \boldsymbol{q}_{ns}(k) - \alpha \left(I_n - J^{+}(\boldsymbol{q}_{ns}) J(\boldsymbol{q}_{ns})\right)^T \boldsymbol{\nabla m_u}(\boldsymbol{q}_{ns}(k)) \Delta t$;
  $k = k + 1$;
end
return $\boldsymbol{q}_{ns}^{des} = \boldsymbol{q}_{ns}(k+1)$

To enforce a joint configuration, a regularization torque, $\boldsymbol{\tau_{m_u}}$ is defined such that

$$\boldsymbol{\tau}_{m_{\boldsymbol{u}}} = k_p(\boldsymbol{q}_{ns}^{des} - \boldsymbol{q}) - k_d\dot{\boldsymbol{q}} \tag{2.63}$$

with k_p and k_d some proportional and derivative gains. The regularisation task is then expressed as in Equation (2.4.4). Finally, the regularisation task is written as a function of the joint torque as:

$$\boldsymbol{R}(\boldsymbol{\tau}) = \|\boldsymbol{\tau} - \boldsymbol{\tau}_{m_{\boldsymbol{u}}}\|_{W_0}^2 . \tag{2.64}$$

Experimental results are shown in Chapter 4.

Conclusion

This chapter details the formulation of a quadratic programming problem for the control of a robotic manipulator. It details the expression of the robot tasks and its constraints.

In this work, the robot variation of kinetic energy is expressed as a constraint. This ensures at any time that the robot cannot use more energy than a defined limit. During transient contact it ensures that the amount of kinetic energy that can be transferred between the robot and a human body part is not dangerous. Through some assumptions, this constraint also allows to limit the wrenches that can be exerted by the robot during quasi-static contact.

A robot perceived mass in a specific direction depends on the inertial properties of each link and on their configuration in 3D space. This mass is directly linked to the robot kinetic energy and thus its dangerousness. Consequently, reducing this perceived mass provides a solution to reduce the robot dangerousness. Since the inertial properties of a robot joints are defined by design, this mass can only be reduced by acting on the robot configuration. Using optimization algorithms, it is possible to find a configuration minimizing this perceived mass. By controlling the robot redundancy through the regularization task, this work proposes to minimize the robot perceived mass in the direction of an obstacle.

The contributions proposed in this chapter are tested in simulation then implemented on a KUKA LWR4+ robot. The next chapter details the experimental setup necessary to validate the proposed approach.

Chapter 3

Experimental setup description and applicative context

The controller presented in Chapter 2 has been implemented on an industrial robot with the tomosynthesis application in mind. This application requires to dynamically position an X-ray source in 3D space while pointing towards a target. Safety is of great importance as the X-ray source moves around a patient. Further, whatever the application, moving the X-ray source very slowly so as to ensure safety can increase the duration of the procedure up to a value that would not be acceptable for medico-economic constraints. Therefore, there is a need for moving "as fast as possible" while ensuring safety. This chapter first details the expression of the tasks for the specific context of tomosynthesis. In a second part, it presents the sensors used and developed at ISIR to validate the controller.

3.1 Application to the defined context

In the context of 3D X-ray imaging, the robot must realise two tasks:

- positioning the X-ray generator in 3D space,
- pointing towards a target (the X-ray detector).

This section details the expression of these tasks according to the robot control variable: the joint torques. In the remaining of this manuscript, a frame attached to a point i is denominated $\mathcal{F}_i$. This frame consists in an origin, denoted $\boldsymbol{X}_i = (x_i, y_i, z_i)$ and

three orthonormal vectors forming a basis of $SO(3)$ denoted $R_i = (\boldsymbol{i}_i, \boldsymbol{j}_i, \boldsymbol{k}_i)$. Frames are expressed relatively to the robot fixed base frame $\mathcal{F}_0\left(\boldsymbol{X}_0, (\boldsymbol{i}_0, \boldsymbol{j}_0, \boldsymbol{k}_0)\right)$, with $\boldsymbol{X}_0 = (0,0,0)$, $\boldsymbol{i}_0 = (1,0,0)$, $\boldsymbol{j}_0 = (0,1,0)$ and $\boldsymbol{k}_0 = (0,0,1)$, represented in Figure 3.1. This robot base is attached to a plane, Π, determined by the point $\boldsymbol{X}_0$ and the normal vector $\boldsymbol{k}_0$.

In this work, tasks are expressed in terms of a controlled Cartesian space acceleration $\dot{\boldsymbol{v}}^* \in \mathbb{R}^6$, where $\boldsymbol{v} \in \mathbb{R}^6$ is a twist expressed relatively to $\mathcal{F}_0$. This twist can be decomposed in a linear part and an angular part, respectively $\boldsymbol{\nu} \in \mathbb{R}^3$ and $\boldsymbol{\omega} \in \mathbb{R}^3$ such that

$$\boldsymbol{v} = \begin{pmatrix} \boldsymbol{\nu} \\ \boldsymbol{\omega} \end{pmatrix}. \tag{3.1}$$

3.1.1 Positioning task

The X-ray source is attached to the robot end-effector. A frame $\mathcal{F}_s\left(\boldsymbol{X}_s, (\boldsymbol{i}_s, \boldsymbol{j}_s, \boldsymbol{k}_s)\right)$ is attached to the X-ray source (where photons are emitted), with $\boldsymbol{k}_s$ the main direction of the X-ray beam. The positioning task is expressed as the servoing of the robot X-ray source position in the 3D space, $\boldsymbol{X}_s \in \mathbb{R}^3$, to a desired position, $\boldsymbol{X}_s^{des} \in \mathbb{R}^3$. Thus, positioning the robot in 3D space only requires to consider the linear part of $\boldsymbol{v}_s$, the X-ray source twist. A PID controller is used to control the X-ray source desired Cartesian position and is defined as

$$\dot{\boldsymbol{\nu}}_s^* = K_p \boldsymbol{e}_\nu + K_d \dot{\boldsymbol{e}}_\nu + K_i \int_{t_0}^{t} \boldsymbol{e}_\nu d\tau + \dot{\boldsymbol{\nu}}_s^{des} \tag{3.2}$$

with $\boldsymbol{e}_\nu = \left(\boldsymbol{X}_s - \boldsymbol{X}_s^{des}\right)$, the tracking error, $\dot{\boldsymbol{e}}_\nu = \left(\boldsymbol{\nu}_s - \boldsymbol{\nu}_s^{des}\right)$ and $\dot{\boldsymbol{\nu}}_s^{des}$ a feed forward term. $\boldsymbol{X}_s^{des}, \boldsymbol{\nu}_s^{des}$ and $\dot{\boldsymbol{\nu}}_s^{des}$ are provided online by a trajectory generator and supposed not known in advance. This is realistic in a context where the trajectory would be adapted online at instant k based on the X-ray images taken at instant $k-1$, $k-2$, $\ldots$, $k-n$.

The positioning task can thus be written

$$T_{positioning}(\boldsymbol{\tau}) = \left\| \boldsymbol{\nu}_s^* - J_{lin} M^{-1} \left(\boldsymbol{\tau} - \boldsymbol{n} - \boldsymbol{g}\right) - \dot{J}_{lin} \dot{\boldsymbol{q}} \right\|_2^2 \tag{3.3}$$

where $J_{lin} \in \mathbb{R}^{3 \times n}$ represents the Jacobian associated with the linear velocity of the laser frame.

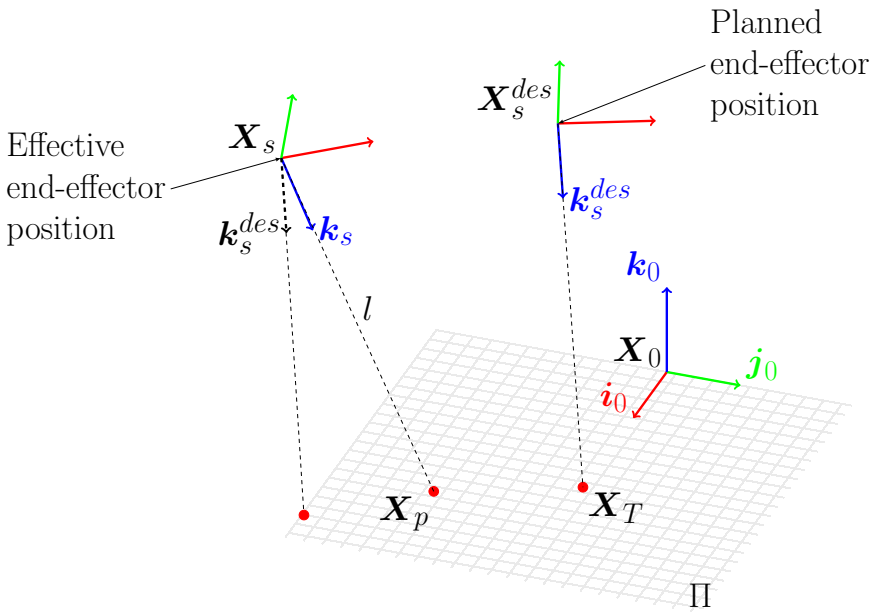


FIGURE 3.1: An illustration depicting the parameters required for the definition of the pointing task. This illustration also shows that the pointing task cannot be planned off-line because an error when positioning the X-ray source will lead to an error when pointing towards the target point $\boldsymbol{X}_T$.

3.1.2 Pointing task

In the context of this work, a pointing task is considered. The robot must point towards a target, the X-ray detector centre, denoted $\boldsymbol{X}_T$, at all time. As depicted in Figure 3.1, the pointing task could be defined as a desired orientation ensuring that $\boldsymbol{X}_T \in (\boldsymbol{X}_s^{des}, \boldsymbol{k}_s)$. However, in that case, any discrepancy between $\boldsymbol{X}_s^{des}$ and $\boldsymbol{X}_s$ would lead to a pointing error even if $\boldsymbol{k}_s = \boldsymbol{k}_s^{des}$. It means that the pointing task cannot be defined off-line and should instead be defined on-line, with respect to the current position of the end-effector. By doing so, an indirect priority is given to the pointing task over the positioning task. This is justified by the applicative context of this work. Two methods can be used to determine this pointing task.

3.1.2.1 Orientation of the laser frame

The pointing task can be defined as the servoing of the robot end-effector orientation. This orientation keeps the X-ray source centred on the target. The target point is T, its position is $\boldsymbol{X}_T \in \mathbb{R}^3$.

Because $\boldsymbol{X}_s$ can differ from $\boldsymbol{X}_s^{des}$ (due to real-time obstacle avoidance, for example), the desired value for $\boldsymbol{k}_s$ is not computed off-line as $\boldsymbol{k}_s^{des} = \frac{\boldsymbol{X}_T - \boldsymbol{X}_s^{des}}{\|\boldsymbol{X}_T - \boldsymbol{X}_s^{des}\|}$. Rather, it is computed on-line as: $\boldsymbol{k}_s^{des} = \frac{\boldsymbol{X}_T - \boldsymbol{X}_s}{\|\boldsymbol{X}_T - \boldsymbol{X}_s\|}$.

The orientation error, $\boldsymbol{e}_\omega$, is then computed as the geodesic rotation from the current $\boldsymbol{k}_s$ to the desired $\boldsymbol{k}_s^{des}$ such that

$$\boldsymbol{e}_\omega = \begin{cases} \theta \boldsymbol{u} \text{ with } \begin{cases} \boldsymbol{u} = \frac{\boldsymbol{k}_s \times \boldsymbol{k}_s^{des}}{\|\boldsymbol{k}_s \times \boldsymbol{k}_s^{des}\|} \\ \theta = asin(\frac{\|\boldsymbol{k}_s \times \boldsymbol{k}_s^{des}\|}{\|\boldsymbol{k}_s\| \|\boldsymbol{k}_s^{des}\|}) \end{cases} & \text{if } \boldsymbol{k}_s \neq \boldsymbol{k}_s^{des} \\ \boldsymbol{0} & \text{otherwise} \end{cases} \tag{3.4}$$

The servoing of the pointing task is thus expressed as

$$\dot{\boldsymbol{\omega}}_s^* = K_p \boldsymbol{e}_\omega + K_d \dot{\boldsymbol{e}}_\omega + K_i \int_{t_0}^{t} \boldsymbol{e}_\omega d\tau. \tag{3.5}$$

The positioning task can thus be written

$$T_{pointing}(\boldsymbol{\tau}) = \left\| \boldsymbol{\omega}_s^* - J_{ang} M^{-1} (\boldsymbol{\tau} - \boldsymbol{n} - \boldsymbol{g}) - \dot{J}_{ang} \dot{\boldsymbol{q}} \right\|_2^2 \tag{3.6}$$

where $J_{ang} \in \mathbb{R}^{3 \times n}$ represents the Jacobian associated with the angular velocity of the laser frame.

3.1.2.2 Positioning of the X-ray source projection point

The pointing task can also be defined as the servoing of the intersection point between the X-ray beam and the detector plane, denoted $\boldsymbol{X}_p \in \mathbb{R}^3$ (represented in Figure 3.1), and the position of the detector centre. To define a PID controller for the servoing of this task, one must determine the position and the velocity of this intersection point. The determination of these two quantities is based on the work of [Vitrani, 2006].

Intersection point position Throughout this section the notation $\overrightarrow{\boldsymbol{X}_i \boldsymbol{X}_j}$ represents the geometric vector going from the point $\boldsymbol{X}_i$ to the point $\boldsymbol{X}_j$. The distance between the point $\boldsymbol{X}_s$ and $\boldsymbol{X}_p$ is equal to l such that $\overrightarrow{\boldsymbol{X}_s \boldsymbol{X}_p} = l \boldsymbol{k}_s$. Furthermore, we assume (without loss of generality) that

$$\overrightarrow{\boldsymbol{X}_0 \boldsymbol{X}_p} \cdot \boldsymbol{k}_0 = 0 \tag{3.7}$$

Where the symbol "$\cdot$" corresponds to a scalar product. Given that $\overrightarrow{\boldsymbol{X}_0 \boldsymbol{X}_p} = \overrightarrow{\boldsymbol{X}_0 \boldsymbol{X}_s} + \overrightarrow{\boldsymbol{X}_s \boldsymbol{X}_p}$, it can be deduced that

$$\begin{cases} l = -\frac{\overrightarrow{\boldsymbol{X}_0\boldsymbol{X}_s}\cdot\boldsymbol{k}_0}{\boldsymbol{k}_s\cdot\boldsymbol{k}_0} \\ \overrightarrow{\boldsymbol{X}_0\boldsymbol{X}_p} = \overrightarrow{\boldsymbol{X}_0\boldsymbol{X}_s} - \frac{\overrightarrow{\boldsymbol{X}_0\boldsymbol{X}_s}\cdot\boldsymbol{k}_0}{\boldsymbol{k}_s\cdot\boldsymbol{k}_0}\cdot\boldsymbol{k}_s \end{cases} \quad \text{if } \boldsymbol{k}_s\cdot\boldsymbol{k}_0 \neq 0 \tag{3.8}$$

The computation of the position of the intersection point $\overrightarrow{\boldsymbol{X}_0\boldsymbol{X}_p}$ is valid if $\boldsymbol{k}_s\cdot\boldsymbol{k}_0 \neq 0$, meaning that the orientation of the end-effector should not be co-linear to the plane Π. It also requires this scalar product to be negative to have the X-ray beam projected in the correct direction.

Intersection point velocity This intersection point velocity is also required for the PID controller formulation.

The velocity of the intersection point relatively to the robot base, $\boldsymbol{\nu}_p \in \mathbb{R}^3$, is

$$\begin{aligned} \boldsymbol{\nu}_p\Big|_{\mathcal{F}_0} &= \frac{d\overrightarrow{\boldsymbol{X}_0\boldsymbol{X}_s}}{dt}\Big|_{\mathcal{F}_0} + \frac{d\overrightarrow{\boldsymbol{X}_s\boldsymbol{X}_p}}{dt}\Big|_{\mathcal{F}_0} \\ &= \boldsymbol{\nu}_s + \frac{d\overrightarrow{\boldsymbol{X}_s\boldsymbol{X}_p}}{dt}\Big|_{\mathcal{F}_0} \end{aligned} \tag{3.9}$$

The velocity of $\boldsymbol{X}_p$ relatively to $\boldsymbol{X}_s$ in the referential $\mathcal{F}_s$ is expressed as

$$\begin{aligned} \frac{d\overrightarrow{\boldsymbol{X}_s\boldsymbol{X}_p}}{dt}\Big|_{\mathcal{F}_0} &= \frac{d\overrightarrow{\boldsymbol{X}_s\boldsymbol{X}_p}}{dt}\Big|_{\mathcal{F}_s} + \boldsymbol{\omega}_s \times \overrightarrow{\boldsymbol{X}_s\boldsymbol{X}_p} \\ &= \dot{l}\boldsymbol{k}_s + \boldsymbol{\omega}_s \times l\boldsymbol{k}_s. \end{aligned} \tag{3.10}$$

The symbol $\times$ corresponds to a cross product. It can be noted that $\boldsymbol{\nu}_p\cdot\boldsymbol{k}_0 = 0$ which corresponds to the fact that $\boldsymbol{X}_p$ belongs to plane Π which is normal to $\boldsymbol{k}_0$. Replacing this term in Equation (3.9) leads to

$$\boldsymbol{\nu}_p\Big|_{\mathcal{F}_0}\cdot\boldsymbol{k}_0 = \boldsymbol{\nu}_s\cdot\boldsymbol{k}_0 + \dot{l}\boldsymbol{k}_s\cdot\boldsymbol{k}_0 + \boldsymbol{\omega}_s \times l\boldsymbol{k}_s\cdot\boldsymbol{k}_0 = 0. \tag{3.11}$$

$\dot{l}$ can be extracted from Equation (3.11) and replaced in Equation (3.10) and (3.9) such that

$$\begin{aligned}\boldsymbol{\nu}_p &= \boldsymbol{\nu}_s - \frac{\boldsymbol{k}_s \cdot \boldsymbol{\nu}_s \cdot \boldsymbol{k}_0 + \boldsymbol{k}_s \cdot \boldsymbol{\omega}_s \times \boldsymbol{k}_s \cdot \boldsymbol{k}_0}{\boldsymbol{k}_s \cdot \boldsymbol{k}_0} + \boldsymbol{\omega}_s \times \boldsymbol{k}_s \\ &= \frac{\boldsymbol{k}_s \cdot \boldsymbol{k}_0 \cdot \boldsymbol{\nu}_s - \boldsymbol{k}_s \cdot \boldsymbol{\nu}_s \cdot \boldsymbol{k}_0 - \boldsymbol{k}_s \cdot \boldsymbol{\omega}_s \times \boldsymbol{k}_s \cdot \boldsymbol{k}_0 + \cdot \boldsymbol{k}_s \cdot \boldsymbol{k}_0 \cdot \boldsymbol{\omega}_s \times \boldsymbol{k}_s}{\boldsymbol{k}_s \cdot \boldsymbol{k}_0}.\end{aligned} \tag{3.12}$$

To simplify this equation, the following relation between cross and scalar product can be used:

$$\boldsymbol{a} \times (\boldsymbol{b} \times \boldsymbol{c}) = (\boldsymbol{a} \cdot \boldsymbol{c}) \cdot \boldsymbol{b} - (\boldsymbol{a} \cdot \boldsymbol{b}) \cdot \boldsymbol{c}.$$

Using this relation, the velocity of the projection point can be expressed according to known quantities and is equal to

$$\boldsymbol{\nu}_p = \frac{1}{\boldsymbol{k}_s \cdot \boldsymbol{k}_0} \boldsymbol{k}_s \times ((\boldsymbol{\nu}_s + \boldsymbol{\omega}_s \times l\boldsymbol{k}_s) \times \boldsymbol{k}_0) \tag{3.13}$$

The servoing of the pointing task is then defined as

$$\dot{\boldsymbol{\nu}}_p^* = K_p \boldsymbol{e} + K_d \dot{\boldsymbol{e}} + K_i \int_{t_0}^{t} \boldsymbol{e} d\tau \tag{3.14}$$

with $\boldsymbol{e} = (\boldsymbol{X}_p - \boldsymbol{X}_T)$ the pointing task error and $\dot{\boldsymbol{e}} = (\boldsymbol{\nu}_T - \boldsymbol{\nu}_p)$, $\boldsymbol{\nu}_T$ the velocity of the target point. In the considered applicative context, the X-ray detector is still and this target velocity is null.

Equation (3.13) can be rewritten using the parameter dependant Jacobian, J', such that $\boldsymbol{v}_p = J'\boldsymbol{v}_s$ and

$$\boldsymbol{v}_p = J' J \dot{\boldsymbol{q}} \tag{3.15}$$

The pointing task can thus be written

$$\begin{aligned} T_{pointing}(\boldsymbol{\tau}) &= \left\| \dot{\boldsymbol{\nu}}_p^* - \dot{\boldsymbol{\nu}}_p \right\|_2^2 \\ &= \left\| \dot{\boldsymbol{\nu}}_p^* - J_{p,lin} M^{-1} (\boldsymbol{\tau} - \boldsymbol{n} - \boldsymbol{g}) - \dot{J}_{p,lin} \dot{\boldsymbol{q}} \right\|_2^2 \end{aligned} \tag{3.16}$$

with $J_p = J'J$ and $\dot{J}_p$ its derivative.

The two presented pointing tasks (Equation (3.6) and Equation (3.16)yield a similar robot behaviour. In the remaining of this work, the latter expression will be used.

3.1.3 General control scheme

Figure 3.2 summarizes the control scheme used for the pointing task. The first part on the left defines the different goals related to the robot tasks. In the specific context of this work, it defines the desired position of the X-ray source, $\boldsymbol{X}_s^{des}(t)$, and the target position, $\boldsymbol{X}_T^{des}(t)$. A function computes the position and velocity of the projection of the laser beam on the Π plane according to the robot current state. This information is given to a PID controller computing the instantaneous operational space acceleration, $\boldsymbol{\nu}_p^*$ and $\boldsymbol{\nu}_p^*$, required to reach the respective goals. Using the robot model, it is possible to define the objective functions as a quadratic programming problem. Based on this model, the different constraints and the regularisation term are also defined. The quadratic programming problem is defined and solved at each time step and generates an optimal solution to the objective functions satisfying the constraints. This solution is expressed as a torque command which is sent to the robot.

The controller presented in Chapter 2 has been implemented in C++ and tested on a redundant robot, the KUKA LWR 4+. It has been tested in simulation beforehand using a dynamic simulator (Gazebo [Koenig and Howard, 2004]). Several experiments have been conducted to verify that the robot kinetic energy stays under a certain limit and that the forces in quasi-static contact do not reach a dangerous level. The remaining of this chapter describes the experimental setup used to validate the controller. External sensors

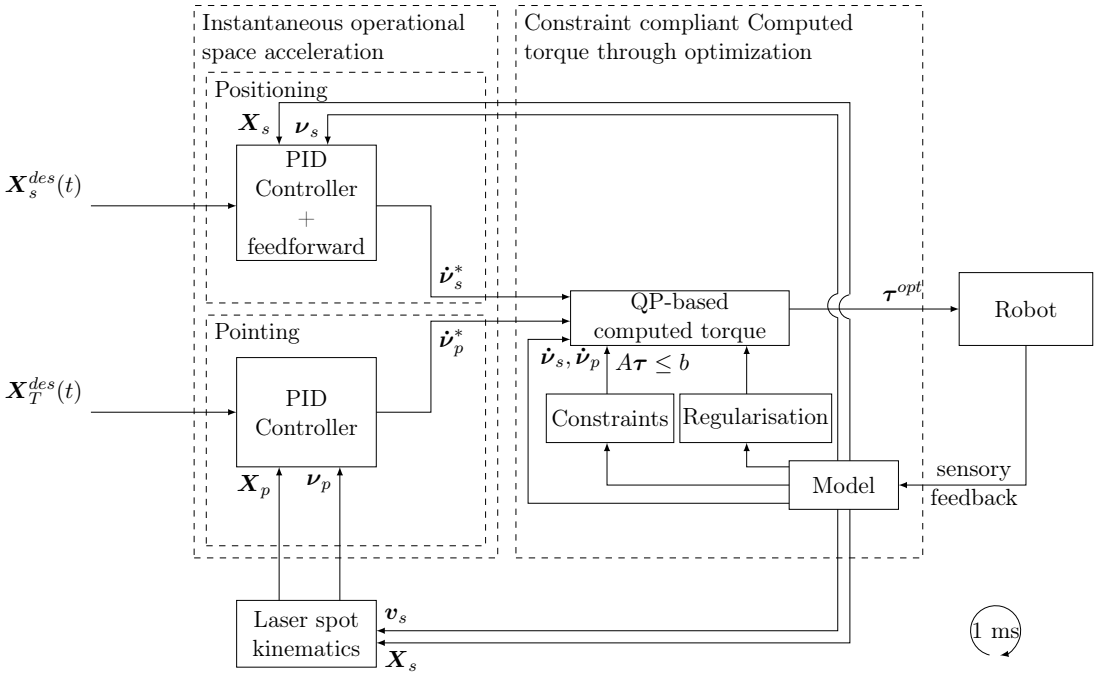


FIGURE 3.2: Representation of the general control scheme used for the experiments in Chapter 4

.

FIGURE 3.3: The Kuka LWR4+ robot

are used and a platform is designed to validate the energetic aspect of the proposed controller.

3.2 The KUKA LWR4+

The controller depicted in Figure 3.2 has been implemented on a KUKA Light Weight Robot (LWR) 4 +, shown in Figure 3.3. This 7 degrees of freedom robot has been specifically designed for human robot interactions [Albu-Schäffer et al., 2007]. Unlike standard industrial robots, the LWR4+ features a payload of 7 Kg for a weight of 16 Kg. Each axis uses a harmonic drives transmission featuring a high coefficient of reduction but also a significant articular flexibility.

This robot features torque sensors placed at the end of each gearbox which provide information on the forces applied on the robot. Using this information, the robot can interact with its environment and can detect human/robot interactions [Haddadin et al., 2011] and be used in comanipulation tasks [Ficuciello et al., 2016]. This robot is suited to be used in collaborative tasks and has been used for manufacturing tasks [Cherubini et al., 2016] but also in medical applications [Chatelain et al., 2017].

This robot is dedicated to research facilities and a more recent version has been put on the market: the LWR IIWA[1]. A version of this robot is certified for medical applications [IEC, 2017]. The proposed framework has been partially implemented on this industrial version at General Electric Healthcare, but the results are not shown in this manuscript.

[1] www.kuka.com/en-de/products/robot-systems/industrial-robots/lbr-iiwa

3.3 Task validation

In the proposed context, the robot must both track a Cartesian trajectory (Equation (3.3)) and point toward a fixed point (Equation (3.16)). This section details the definition of both task errors and how they are recorded during experiments. For safety reasons, an X-ray generator could not be placed on the robot at this stage of the project. Instead, a laser is attached on the robot to simulate the X-ray beam centre.

3.3.1 Positioning error

The positioning error is defined as the Euclidean distance between $\boldsymbol{X}_s$ and $\boldsymbol{X}_s^{des}$

$$\varepsilon_{pos} = \|\boldsymbol{X}_s - \boldsymbol{X}_s^{des}\|. \tag{3.17}$$

The measure of the positioning error relies on the robot encoders and its geometric model parameters. Magneto-resistive encoders are placed on the motors sides, before the gearboxes. A study has been conducted at ISIR to validate that the position recorded by the KUKA encoder reflects the end-effector position despite flexibility or backlash. To that aim, an external measuring system has been used to record the position of each axis relatively to the robot base. It resulted that the precision of the robot was correct without charge with an error inferior to 0.2 mm. When the robot is lifting a load of 5 kg, the error rises up to 3 mm. This is caused by the gearbox and the structure flexibility. To alleviate this issue, it is possible to estimate the robot joint flexibility and to compensate for it at the control level [Jubien et al., 2014]. In the context of this work, the robot is not lifting any load. The positioning error is thus considered as reliable.

3.3.2 Pointing error

The beam emitted by an X-ray source is often represented by a cone. In this work, it is considered that this cone has a revolution axis starting from the position of the laser and intersecting Plane Π. The pointing error is defined as the Euclidean distance between $\boldsymbol{X}_p$ and $\boldsymbol{X}_T$.

$$\varepsilon_{point} = \|\boldsymbol{X}_p - \boldsymbol{X}_T\|. \tag{3.18}$$

To get a physical representation of $\boldsymbol{X}_p$, a laser is attached to the LWR4+ end-effector. Thus, $\boldsymbol{X}_p$ represents the laser projection on the plane Π. To measure the laser

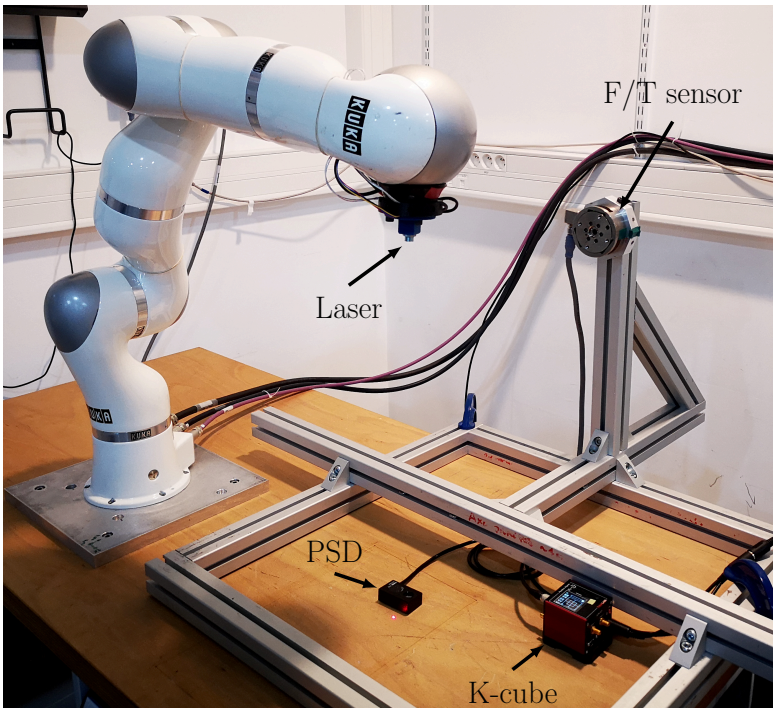


FIGURE 3.4: Setup for tasks validation

beam position, a tetra-lateral Position Sensitive Device (PSD) is used[2]. It consists in a Positive Intrinsic Negative (PIN) diode with a resistive layer. When photons hits the sensing area, a change in local resistance occurs and produces a flow of electrons. The current is then gathered by electrodes placed at the corner of the sensor. The measure of the intensity of each electrode gives information regarding the laser spot localization on the sensor in both x and y direction. The sensor resolution depends on the laser intensity and can be as low as 0.75 μm. The sensor range is 9 mm in each direction. The sensor signals are gathered by a Thorlabs K-cube[3] and retrieved via UART serial communication.

A calibration of the laser relatively to the robot end-effector is required to correctly control the beam position. The steps to follow in order to correctly perform the calibration of the laser and the PSD sensor positions are detailed in Appendix A. The overall setup for task validation is depicted in Figure 3.4.

3.4 Constraints validation

This work proposes a control scheme imposing constraints on the robot. These constraints are linked to the robot dangerousness. This section presents the sensors that are

[2] www.thorlabs.com/thorproduct.cfm?partnumber=PDP90A
[3] www.thorlabs.com/thorproduct.cfm?partnumber=KPA101

used to validate the correct enforcement of these constraints.

3.4.1 Force measurements

The proposed work requires a quantification of the efforts exerted by the robot during quasi-static contact. To do so an ATI Gamma sensor[4] (FTN GAMMA SI-130-10) is used. This force sensor can measure up to ±130 N in the x and y directions and ±400 N in the z direction. This sensor is well suited to assess the respect of the ISO Norm 15066 on quasi-static contact considering the range of efforts to be applied (from 65 N for the face to 220 N for the knees). This sensor, represented in Figure 3.4, is mounted on a rigid structure so that the force exerted by the robot during collision are recorded along the z direction.

3.4.2 Energy measurements

An external device was designed at ISIR to quantify the energy transferred from the robot to an obstacle during transient contact. This device, depicted in Figure 3.5, measures the potential energy stored in a spring when a contact occurs. Assuming that the robot kinetic energy is transferred to the platform and that the friction effects are negligible, this potential energy is equal to the kinetic energy transferred by the robot.

The platform consists in a rigid fixed base on which a rotating beam is mounted. A spring with stiffness k is attached on one side of the beam. A rubber band is used to preload the spring. The robot enters in collision at the other extremity of the beam. Ball bearings are used to reduce friction at best. A 13-bit encoder is placed at the centre of rotation of the structure and a micro-controller queries its position at 500 Hz and transmits the information via a serial port. The resulting elongation, Δx, of the spring is obtained using geometric relations.

A schematic view of this platform is depicted in Figure 3.6. To compute the elongation of the spring, the following variables are defined:

- b: the distance between the spring fixation point on the beam and the encoder centre.
- c: the nominal length of the spring
- l: the spring length after contact
- θ: the pivot angle after contact (measured by the encoder)

[4] www.ati-ia.com/pt-BR/products/ft/ft_models.aspx?id=Gamma

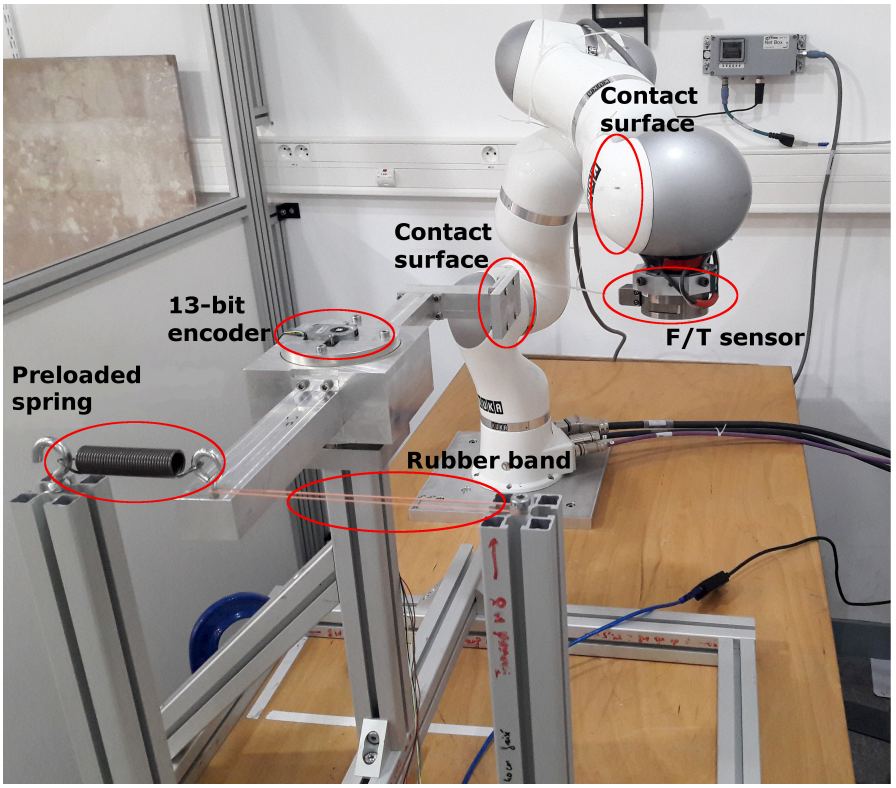


FIGURE 3.5: Device for measuring the energy transferred by a robot during an impact. The elongation of the spring is measured by an encoder to determine the potential energy accumulated when an impact occurs

It results that $\Delta x = l - c$. Using the cosine formula, the spring elongation after contact can be determined such that:

$$l^2 = b^2 + d^2 - 2bdcos(\phi + \theta). \tag{3.19}$$

with $d = \sqrt{b^2 + c^2}$ and $\phi = arctan\left(\frac{c}{b}\right)$. The elongation of the spring Δx is thus equal to

$$\Delta x(\theta) = \sqrt{b^2 + d^2 - 2bdcos(\phi + \theta)} - c. \tag{3.20}$$

The resulting potential energy stored in the spring, e_p, after an impact is

$$e_p = \frac{1}{2} k \Delta x^2. \tag{3.21}$$

In the developed platform, the spring stiffness is measured beforehand using a micrometric table and a dynamometer with a 0.05N resolution. It resulted a measured stiffness of 738 N/m. The distance b is equal to 0.236 m. The spring is preloaded to

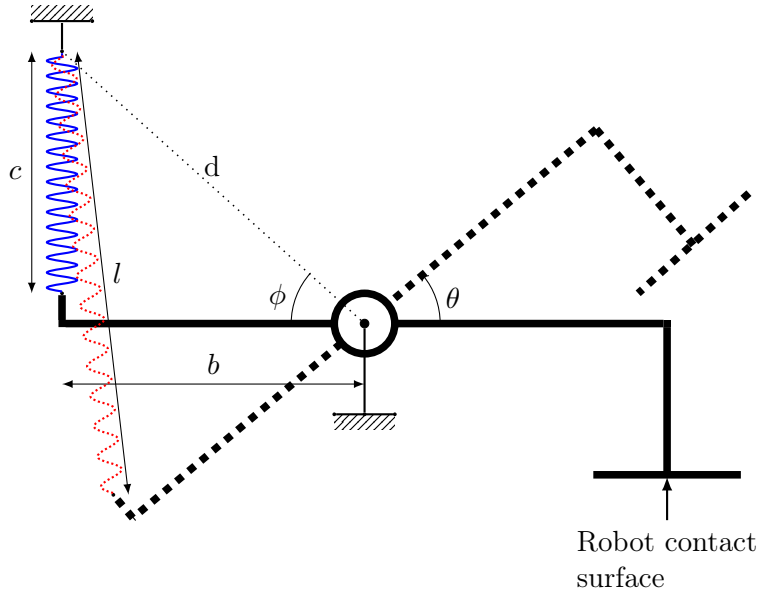


FIGURE 3.6: Geometric view of the platform

ensure that it elongates as soon as the contact occurs. To do so a rubber band with a negligible stiffness is used. The spring nominal length is measured with a calliper and is equal to 0.1005 m. The encoder used is an AMT 102 13-bit encoder. The platform is made modular so that the energy can be measured with different robot configurations. To that aim, aluminium profiles are used so that the whole platform can move along the $\boldsymbol{i}_0\boldsymbol{j}_0$ plane. Using beams with different size, the robot can also collide with the platform at different heights. This platform is used in Section 4.2 to validate the kinetic energy constraints expressed in Equation (2.49).

The instant of collision must be precisely known to be able to compare the robot kinetic energy at the instant of impact with the energy dissipated in the collision platform. One could set the instant of collision as the instant where a rising edge of the collision platform encoder signal is observed. However, since the collision platform is communicating via a serial port, the informations given by the platform encoder are not synchronised with the information given by the robot. At the instant of collision, the inertial forces acting on the robot keep on pushing it in the direction of motion. To indirectly detect the instant of contact, one could record the forces acting on the robot and detect a peak force at the instant of collision. To that aim, during the experiments, the 6^{th} axis of the robot will enter in collision with the platform and an ATI F/T sensor, placed at the robot end-effector (see Figure 3.5), will record the inertial forces acting on the robot.

3.5 Vision system

The torque sensors used in the LWR4+ provide information on the occurrence of a collision. However, by nature they fail to anticipate contact. External sensors able to detect the position of objects around the robot are required to avoid obstacles or to adapt its behaviour around obstacles.

In this work RGBD cameras (one Kinect 2 and two Xtions) are used to measure at 30 Hz the position of an obstacle in the robot environment. The process described in the following is implemented in an open source C++[5] code. An algorithm removes the robot from the depth image using its forward kinematics model. Depth images of the scene are acquired to have knowledge of the background. Any object entering the field of the camera becomes a potential obstacle. Obstacles are then clustered in point-clouds and only the closest object to the robot is considered. This process allows using several RGBD cameras for the detection of obstacles. By placing these cameras at different location it is possible to reduce the obstruction phenomenon caused by the robot or objects placed in the scene. Figure 3.7 depicts the global setup with 3 RGBD cameras.

To simplify the distance computation time, the closest cluster is inserted in an elliptic cylinder. The revolution axis of the cylinder is located at the barycentre of the cluster. The furthest point on the cluster on each axis defines the radiuses of the ellipse. The position of the closest point is then used to determine the distance between the robot and the closest obstacle. This position is also used to determine the direction of the closest obstacle towards the robot. This information is then used for different experiments presented in Chapter 4.

It should be noted that these RGBD cameras are rarely used in industrial environments. This is because they are sensitive to light intensity, dust, *etc.* and thus do not fulfil the requirements set by ISO TS 13849 on safety-related parts of control systems [ISO/TS-13849, 2015]. To obtain information about the proximity of an obstacle to the robot, one can use safety laser scanners. These devices use a laser technology to measure the distance to an obstacle and fulfil the requirements set by ISO 13849. However, these solutions come at a much higher cost. The simplest ones, such as the SICK sensor S300[6], project a laser on a plane. Only the distance between an object intersecting this plane is recorded. Thus, the information gathered by the device is less rich than with a RGBD sensor that can record the position of objects inside a given volume.

[5] www.github.com/kuka-isir/kinects_human_tracking

[6] www.sick.com/us/en/opto-electronic-protective-devices/safety-laser-scanners/s3000-professional/c/g187229

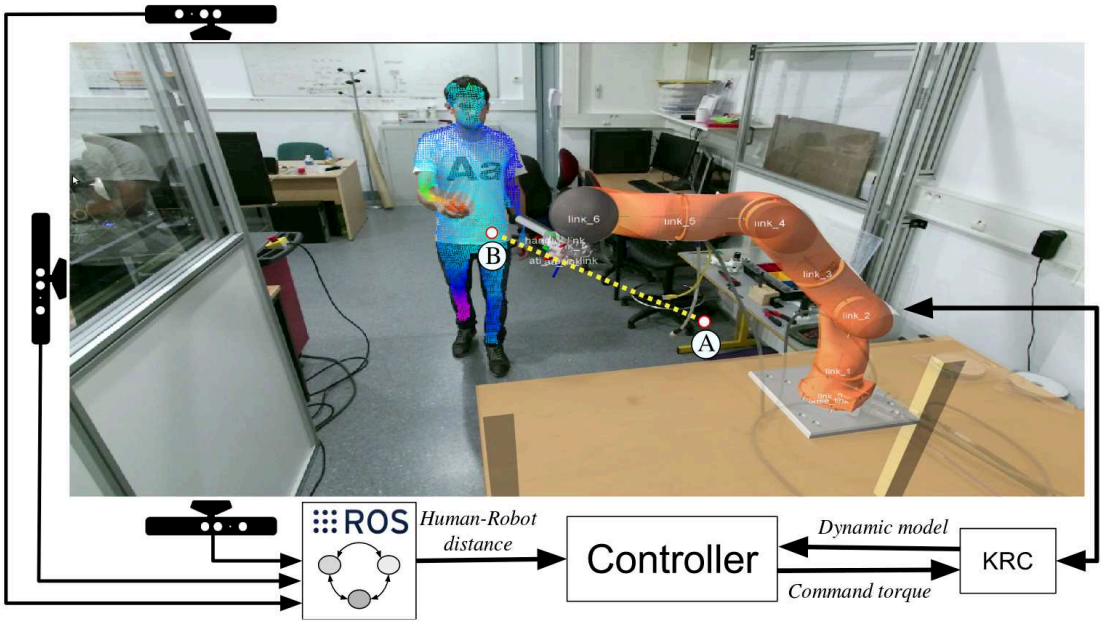


FIGURE 3.7: Scene observed by the RGBD cameras (extracted from [Meguenani et al., 2017])

3.6 Software and communication

The proposed controller is intended to work on a torque controlled robot. In order to be able to use the robot full dynamics, its is required to be able to receive the robot state and send torque commands at 1 kHz. To that aim, an advanced software architecture is used. Sensors data are acquired by separate micro-controllers then published as ROS topics. The QP problem is solved using a state-of-the-art solver: qpOASES [Ferreau et al., 2014]. The control framework is implemented as a C++ OROCOS component [Bruyninckx, 2002] inside a generic software architecture developed at ISIR for robot manipulators[7]. This architecture allows to either simulate the robot using Gazebo [Koenig and Howard, 2004] or to send commands to the robot using the KUKA Fast Research Interface (FRI) [Schreiber et al., 2010]. These commands are sent via a PC running on a Xenomai kernel[8] using the RTnet communication library[9] to ensure minimum jitter during real-time Ethernet communication with the robot.

Conclusion

The work presented in this manuscript has been entirely tested on a KUKA LWR4+ robot. With the tomosynthesis applicative context in mind, it uses a state-of-the-art

[7] www.github.com/kuka-isir/rtt_lwr
[8] www.xenomai.org
[9] www.rtnet.org/

quadratic programming solver on a real-time enabled computer to send torque commands to a KUKA robot at 1 kHz. Several sensors are used to measure both the correct achievement of the defined tasks and the correct enforcement of the constraints. A specific platform has been designed to measure the energy dissipated by the robot during an impact. This setup is used in the next chapter for several experiments on the proposed control architecture.

Chapter 4

Experimental results

Chapter 2 exposes several contributions to improve the safety around a robot sharing its workspace with a human operator, especially in the event of an undesired collision. In this chapter, the proposed control architecture and methodology are implemented on a KUKA LWR4+ robot. The correct tasks realisation are validated using the external sensors described in Chapter 3. The corresponding results are presented in Section 4.1. Section 4.2 introduces the results related to the correct constraints enforcement in various scenarios involving contact with the robot. Finally, Section 4.3 presents several regularization tasks to control the robot redundancy. This section demonstrates the effectiveness of the regularization task minimizing the robot perceived mass.

4.1 Tasks validation

Robots are used for the realisation of high-level objectives. In this work these objectives are the positioning of the X-ray generator and the pointing towards an X-ray detector. This section provides experimental results to verify that the tasks expressed in the proposed control architecture are correctly performed. The regularization task used for these experiments is set to compensate for gravity (see Equation (2.52)). The robot behaviour resulting from this regularisation task can be observed in Section 4.3.1.

The first experiment validates the expression of the tasks with a trajectory computed off-line. In a second experiment, the reference position is updated on-line using the distance between an obstacle and the robot to demonstrate that the tasks can be redefined reactively. The setup depicted in Figure 3.4 is used for the validation.

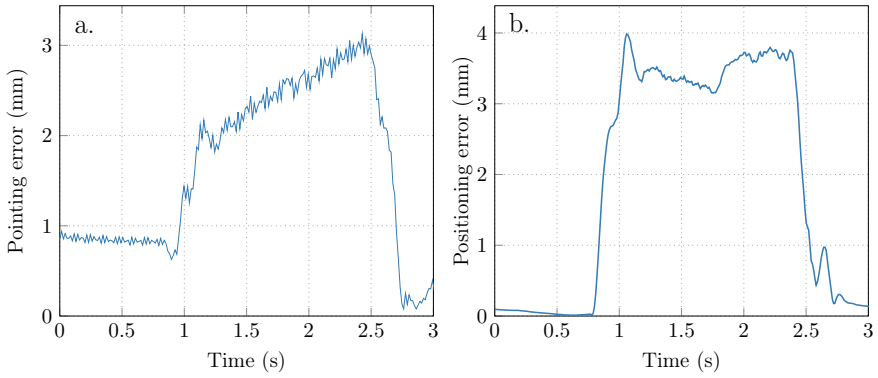


FIGURE 4.1: Evolution of (a.) the pointing error, (b.) the positioning error when performing a motion in the nominal case

4.1.1 Nominal case

During the experiments, the robot performs a standard motion to obtain a 3D image using X-rays. To do so it moves along a line from (0.5,-0.2,0.4) m to (0.5,0.2,0.4) m in $\mathcal{F}_0$ while pointing at a target located at (0.5,0.0,0.013) m. A trajectory is created with the KDL library [Smits, 2018] with a timing law constituted of a trapezoidal velocity profile with a maximum velocity of 0.25 m/s and a maximal acceleration of 1 m/s^2. The trajectory is defined offline with an initial configuration chosen so that the robot does not go close to its intrinsic constraints and the kinetic energy constraint is not activated. The PID controller gains are tuned using the Ziegler–Nichols method [Ziegler and Nichols, 1942]. The gains used in these experiments are: $K_p = diag(2280, 2280, 2080, 4000, 4000, 4000)s^{-2}$, $K_d = diag(47, 47, 20, 45.4, 45.4, 45.4)s^{-1}$ and $K_i = diag(6, 6, 6, 25, 25, 25)s^{-3}$

A video associated with this experiment is available here (first part). Figure 4.1 shows a typical result of this experiment. During the motion, the mean positioning error is 3.5 mm and the average pointing error is 2.20 mm. To stress the importance of a stiff PID control for the pointing task, it should be noted that, at a height of 0.4 m, an orientation error of 1° around the x or y-axis induces a pointing error of 7 mm. The remaining errors that can be observed in Figure 4.1 are the result of friction effects at each axis that are not satisfactorily compensated. This can be explained by the fact that the torque commands sent via the KUKA FRI are not the torques actually sent to the robot actuators. Indeed, KUKA internally uses a low-level torque control loop about which little information is publicly available and which may induce some errors with the torque computed by the quadratic programming solver. Furthermore, these friction effects could probably be reduced by tuning the integral term more aggressively but at the risk of losing stability.

These results show the correct implementation of the PID controllers inside the quadratic programming problem for both tasks. The expression of the pointing task depending on the position of the laser (see Equation (3.16)) is correctly performed and yields tracking results that are satisfying for the considered applicative context.

4.1.2 On-line trajectory definition

In the context of this work, the robot must adapt its trajectory in real-time. The proposed architecture is well suited for such application where new inputs can be taken into account every millisecond. To assess the ability to perform motions defined on-line, this section proposes to redefine the robot trajectory according to an on-line input: the position of an obstacle in the robot environment. This implementation can be interpreted as an obstacle avoidance implementation. While rudimentary, it shows the versatility of the proposed solution.

At each time step, the localization of an obstacle in the robot environment is observed using three RGBD cameras as described in Section 3.5. If the distance, d, between the obstacle and the robot is less than a defined limit, d_{lim}, a new desired position is computed to keep the robot at a safe distance. Let $\boldsymbol{u}_{obs} \in \mathbb{R}^3$ be a unit vector representing the direction of the obstacle towards the robot. $\boldsymbol{X}_s^{goal}$ is a goal, fixed position, to reach. Algorithm 4 details the computation of the robot new desired position according to the distance to an obstacle.

Algorithm 4: On-line definition of the end-effector desired position

Compute the direction to the obstacle, $\boldsymbol{u}_{obs}$
if $d \leq d_{lim}$ **then**
  $\boldsymbol{X}_s^{des} = \boldsymbol{X}_s^{goal} + (d_{lim} - d)\boldsymbol{u}_{obs}$
else
  $\boldsymbol{X}_s^{des} = \boldsymbol{X}_s^{goal}$
end
return $\boldsymbol{X}_s^{des}$

For this experiment, the robot is required to stay at a fixed point. A human moves close to the robot and is considered as an obstacle in Algorithm 4. A safe distance $d_{lim} = 0.4$ m is enforced.

Figure 4.2 depicts the results of this experiment. When the human is out of the detection range, the distance, d, is null and both the pointing and the positioning task errors are stable with a mean positioning error of 0.4 mm and a mean pointing error of 0.7 mm. When the human reaches the safe distance limit, a new desired position is defined

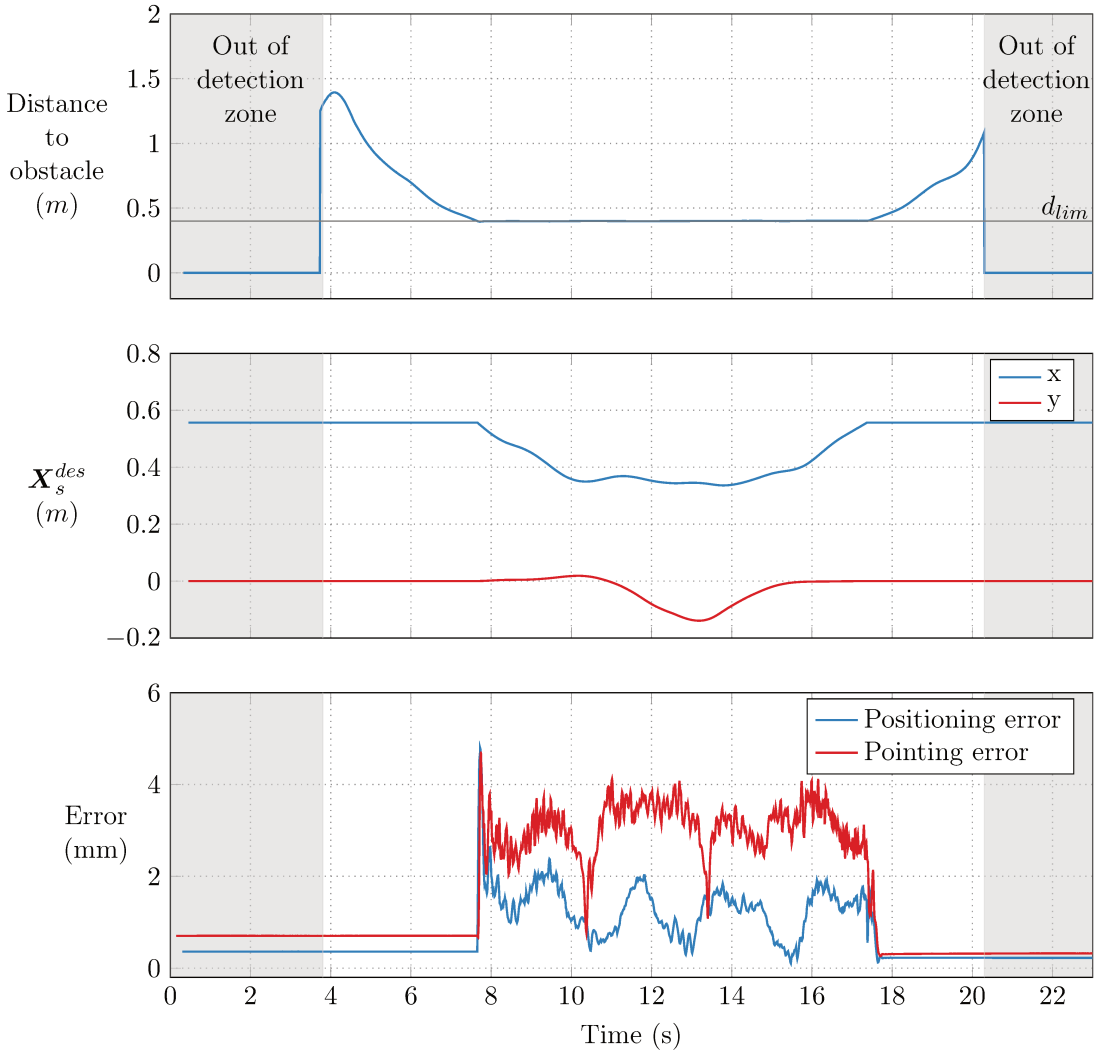


FIGURE 4.2: Redefinition of the robot desired position on-line. A new desired Cartesian position is computed reactively to stay away from an obstacle.

to keep the robot away from the obstacle. It can be seen that this on-line redefinition of the desired position successfully keeps the robot at a safe distance. The positioning and pointing errors are slightly affected by this perturbation but with a magnitude similar to the previous experiment where the robot was following a trajectory. The mean positioning error is 3.0 mm and the mean pointing error is 1.23 mm. It should be noted that at some point the robot might reach its joint limits. In such case, the defined obstacle avoidance task will not be correctly achieved. If one wants a strict distance between the robot and the human, obstacle avoidance should be formulated as a constraint (using for example the formulation presented in Equation (2.33)).

These experiments show the correct definition of the tasks in Section 3.1. The PID controller formulation ensures a good tracking of both the positioning and the pointing tasks. Furthermore, the robot desired state can be redefined on-line, according to a

higher level decision scheme. The next section is focused on the evaluation of the kinetic energy constraint.

4.2 Kinetic energy constraint validation

This section features several experiments showing the interesting properties of the kinetic energy constraint. These experiments validate the correct enforcement of the constraint when a collision occurs and show experimental results during a transient and a quasi-static contact.

4.2.1 Model based kinetic energy computation validation

The equation of motion is important to express constraints as functions of the control variable, $\boldsymbol{\tau}$. This equation highly depends on the model used by the robot. When the equation of motion is used in the description of the robot tasks, the PID controller corrects the small errors of the dynamic model and the velocity kinematics. However, the expression of the kinetic energy constraint requires a precise model of the robot. To validate that the model used for the robot is precise enough, the platform described in Section 3.4.2 is used.

During this experiment, the robot moves along a straight line with a target position located behind the platform. The trajectory is defined so that the robot 6^{th} axis enters in collision with the collision device (see Figure 3.5). The contact is detected when the ATI sensor, placed at the robot end-effector, records a force superior to 0.2 N. When a contact is detected, the positioning task is cancelled so that only the regularization task remains, leaving the robot in gravity compensation mode. This is to ensure that the PID controller does not integrate the error resulting from the collision and thus injects more energy in the platform.

Figure 4.3.a. compares the energy of the robot and the energy stored in the spring when a collision occurs. The orange curve represents the average kinetic energy of the robot during the experiment over 10 trials. The blue curve represents the average potential energy stored in the spring. In dark is the force recorded by the ATI sensor along the y-axis. Overall, the robot average kinetic energy at the instant of contact is 0.516 J. The average potential energy stored in the spring rises up to 0.528 J. It results a mean error of 3.17 % and a standard deviation of 0.012 J between the model based kinetic energy and the measurement of the platform. This experiment thus validates the model-based computation of the kinetic energy.

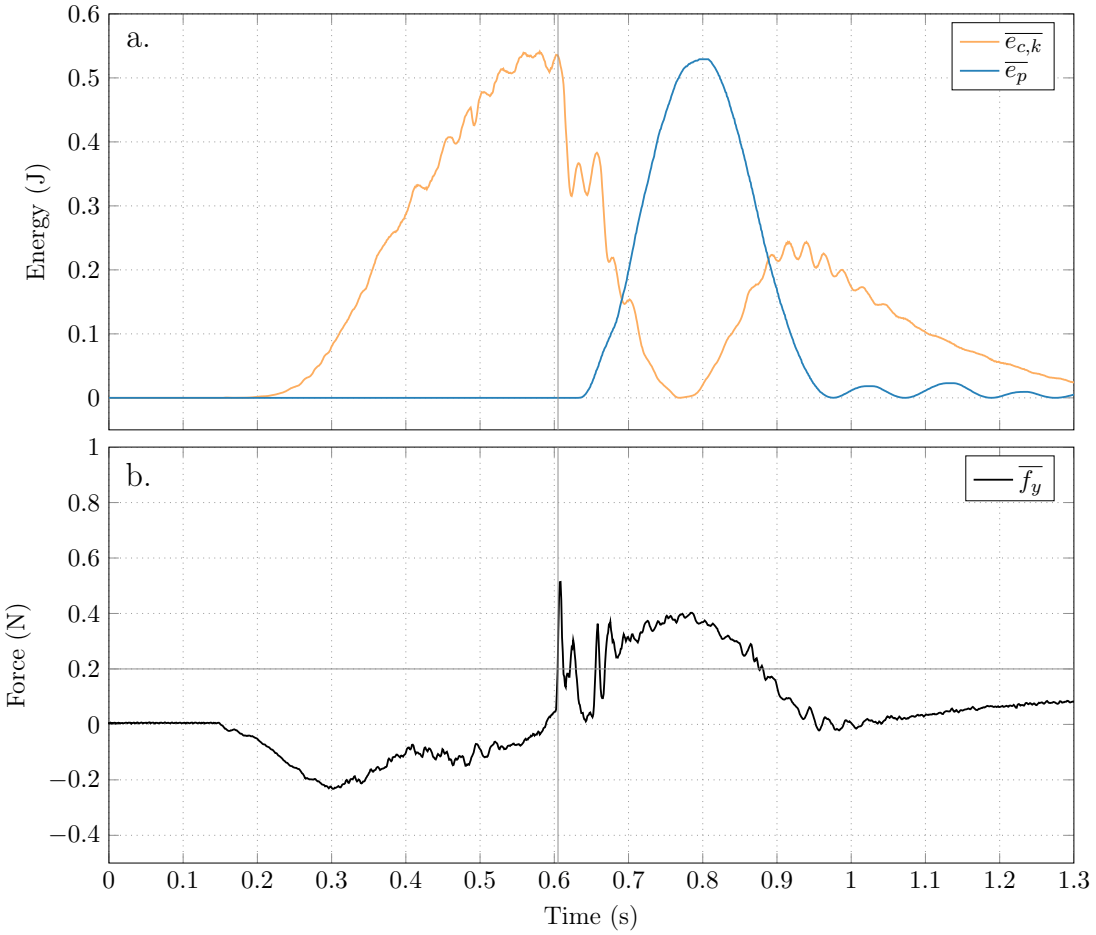


FIGURE 4.3: Measure of the dissipated kinetic energy during an impact with the platform. (a.) Comparison between the current kinetic energy and the potential energy recorded by the platform. (b.) Force measured by the ATI sensor in the y-direction. The vertical black line represent the instant when a collision has been detected corresponding to a threshold of the measured force of 0.2 N

	Mean (J)	SD (J)
Robot	0.1938	0.0078
Platform	0.1993	0.0077
Error	0.0055 (2.84%)	0.0021

TABLE 4.1: Mean energy and standard deviation of the measure for a velocity of 30 cm/s

	Mean (J)	SD (J)
Robot	0.5156	0.0102
Platform	0.5285	0.0083
Error	0.0129 (3.17%)	0.0105

TABLE 4.2: Mean energy and standard deviation of the measure for a velocity of 50 cm/s

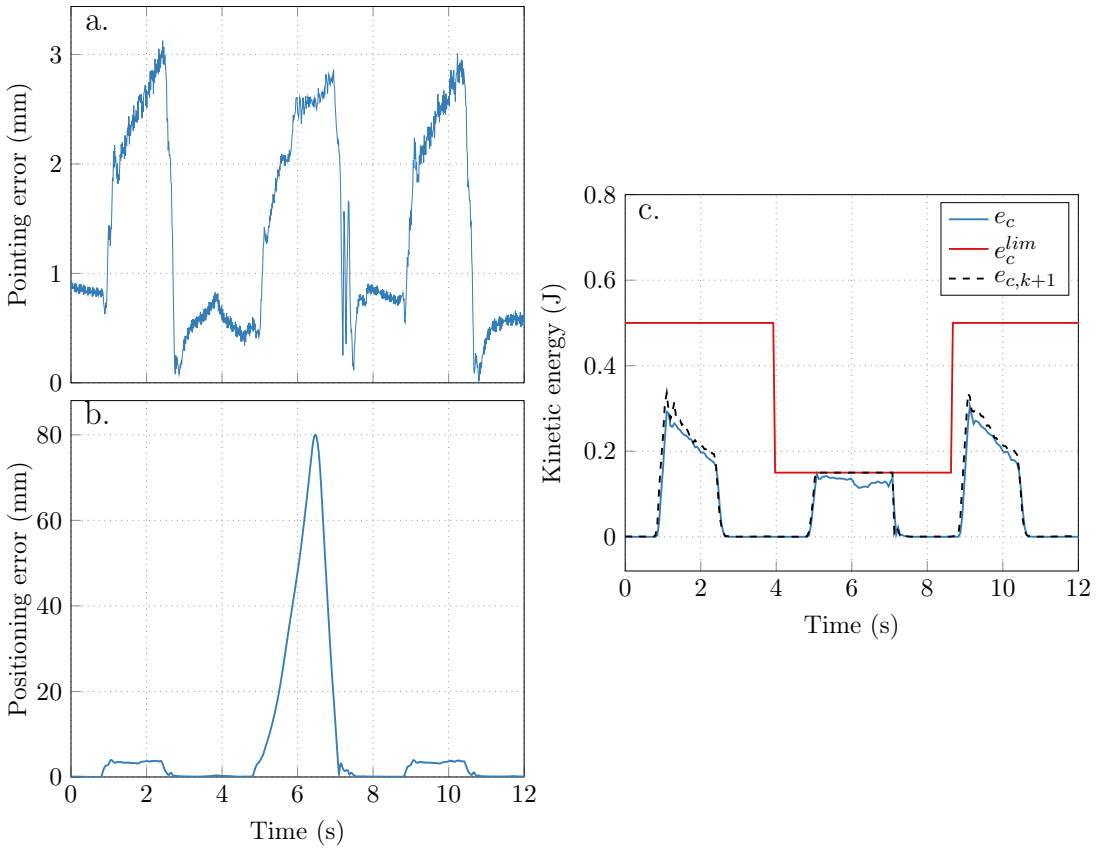


FIGURE 4.4: Evolution of (a.) the pointing error, (b.) the positioning error, (c.) the current kinetic energy (blue line), e_c^{lim} (red), the provisional kinetic energy $e_{c,k+1}$ (dashed).

Tables 4.1 and 4.2 show the results of the measures over 10 trials for different velocities. The repeatability of the platform is good and the measured errors are sufficiently small to validate the robot model.

4.2.2 Kinetic energy limit

In the following experiments, the kinetic energy constraint is activated. The robot behaviour as well as the correct task realisation are observed. The same conditions as in Section 4.1 are used in this experiment. The robot is moving back and forth along the $\boldsymbol{j}_0$-direction. The first motion is realised with a kinetic energy limit far above the kinetic energy required to perform the motion (0.5 J). For the second motion, the kinetic energy limit is set to 0.15 J.

A video associated with this experiment is available here. Figure 4.4 shows the results of this experiment. The first part of this experiment actually corresponds to the results of the first experiment displayed in Figure 4.1. It can be seen that during steady state, the pointing error is below 1 mm. When e_c^{lim} is set to 0.15 J, the available kinetic energy

is insufficient to correctly follow the trajectory. Indeed, the torque computed by the QP solver must induce a provisional kinetic energy $e_{c,k+1}$ (dashed in Figure 4.4c.) that does not go beyond e_c^{lim}. This phenomenon can be observed at around 5 s in Figure 4.4c. Consequently, the optimal solution implies a slower motion which results in a positioning error of up to 80 mm with relation to the planned trajectory. However, the expression of the pointing task as a function of the current robot position allows maintaining the pointing error similar to the one in the nominal conditions.

4.2.2.1 Transient contact

In this section the robot behaviour is analysed during a transient contact with the kinetic energy constraint activated. In a first experiment, the robot enters in collision with the energy measuring device designed at ISIR (see Section 3.4.2). The energy dissipated in this device is compared to the robot kinetic energy at the instant of contact. In a second experiment, a human enters in contact with the robot for a short period of time.

Collision with the platform In this first experiment, the robot enters in contact with the collision platform. Figure 4.5 depicts the kinetic energy that is transferred from the robot to the platform when the kinetic energy constraint is activated.

The kinetic energy limit is set to 0.2 J, below the 0.4 J required to correctly execute the task. The horizon time step Δt is set to 15 ms. Once again, the experiments are realised 10 times. The red curve in Figure 4.5a. represents the kinetic energy limit, e_c^{lim}. The green curve represents the average predicted kinetic energy computed with the robot dynamic model. It is the energy, expressed in Equation (2.40), that is constrained inside the QP solver. The orange curve represents the robot average kinetic energy at the current control instant. The blue curve is the potential energy stored in the platform spring. Figure 4.5b. represents the average force recorded by the ATI sensor in the y-direction. Once again, the contact is detected when the ATI sensor records a force superior to 0.2 N in the y direction. The predicted energy being expressed relatively to a horizon of time Δt, it represents the energy that the robot would have in that horizon of time. The QP solver finds a torque solution ensuring that this predicted kinetic energy never exceeds the limit, e_c^{lim}. Consequently, the resulting real kinetic energy is inferior to the predicted one. When the collision occurs, this current kinetic energy is on average equal to 0.172 J. The potential energy stored in the spring after impact rises up to 0.170 J. Over 10 trials, the standard deviation of the error is 0.002 J.

This experiment shows that it is possible to constrain the robot kinetic energy to prevent a dangerous transfer of energy in case of transient contact with the robot. The

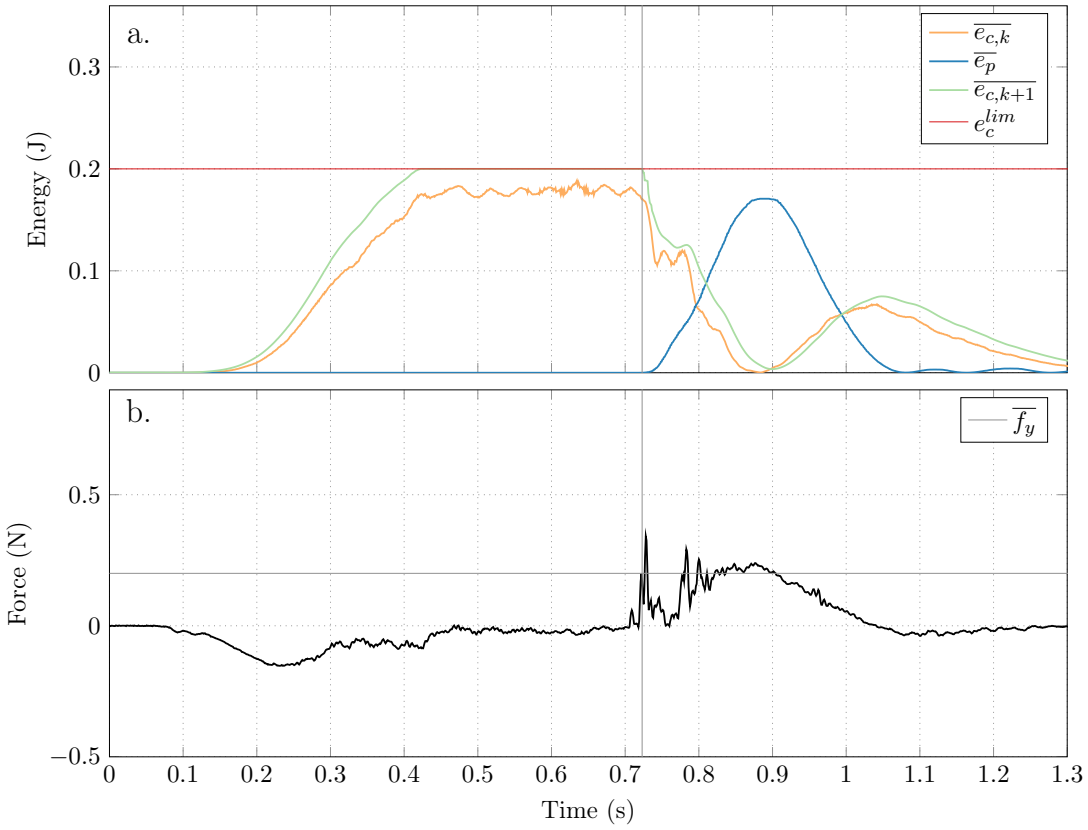


FIGURE 4.5: Energy dissipation during a transient contact recorded by the platform. (a.) Comparison with the provisional kinetic energy, the current kinetic energy and the potential energy recorded by the platform. (b.) Force measured by the ATI sensor in the y direction. The vertical black lines represents the instant when a collision has been detected corresponding to a threshold of the measured force of 0.2 N

FIGURE 4.6: Physical interaction with the robot. An operator restrains the robot motion along its y-axis.

actual robot kinetic energy is always smaller than the defined limit. This is because the constraint is formulated in terms of a predicted energy. However, this can be seen as a safety margin. The next experiment is now focused on a contact with a human while the robot is performing its tasks.

Collision with a human In this second experiment, the robot performs a back and forth motion while pointing towards a fix target. The regularisation task is set to minimize the gravity induced torques. At some point, a human operator is preventing the robot from moving for a brief period as shown in Figure 4.6, the contact is then released. Figure 4.7 depicts the results of this experiment. A video associated with this experiment is available here

It can be observed that when the robot is disturbed and almost stopped, the current kinetic energy becomes null while the provisional kinetic energy quickly reaches the defined limit, e_c^{lim}. This is because the integral term in the PID controller accumulates the tracking error and computes a desired Cartesian acceleration, $\dot{\boldsymbol{v}}_s^*$, aiming at rejecting the perturbation. This acceleration is used in the constraint developed in Equation (2.41) to determine the robot provisional kinetic energy, $e_{c,k+1}$, in Δt seconds. The formulation of this constraint inside the QP ensures that the kinetic energy limit is never crossed. It has been checked during the experiment that the kinetic energy constraint reaches its limit before saturation of the PID controller. The interaction with the human induces a trajectory tracking error of up to 145 mm and a small pointing error of around 3 mm. Furthermore, the energy constraint prevents the transformation of the accumulated error in the PID controller into a control torque that would result in a sudden release of energy. As a matter of fact, it can be observed that the provisional kinetic energy (dashed in Figure 4.7c.) stays at the limit even after contact breaks. This allows a safer interaction. The system then reduces its error until accomplishment of the trajectory tracking task with a safe amount of energy.

This experiment shows the interesting characteristics of the proposed solution with a robotic behaviour providing a meaningful protection against sudden release of energy when contact breaks. It shows the interesting properties of expressing the pointing task as a function of the positioning task. Indeed, when the robot is stopped by the operator, if the pointing task was defined through the same trajectory generator as for the positioning task, then it would have led to an incorrect orientation towards the target. By reactively taking into account the position of the robot, the pointing task keeps the laser spot close to the defined the target.

4.2.2.2 Quasi static contact

When the robot hits a fixed obstacle, the applied contact wrench must not exceed a limit specified by the ISO TS 15066. To assess that the kinetic energy constraint can also be used to indirectly limit the contact wrench, the ATI sensor recording forces and torques, is mounted on a girder that is placed on the robot trajectory. In this experiment, the

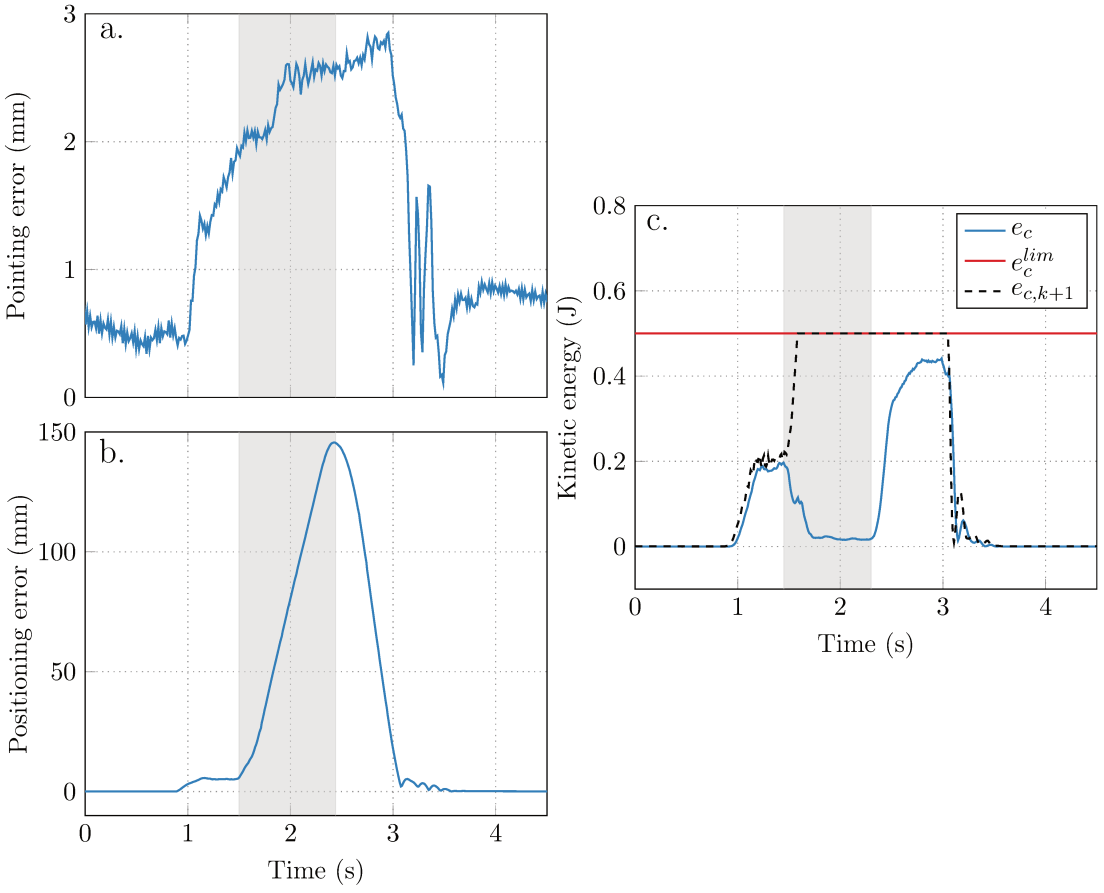


FIGURE 4.7: Evolution of (a.) the pointing error, (b.) the positioning error, (c.) the current kinetic energy (blue line), e_c^{lim} (red), the provisional kinetic energy $e_{c,k+1}$ (dashed). The grey areas represent the interaction phase with a human.

robot is required to follow a straight line with a goal position beyond the ATI sensor. Figure 4.8 shows the results of this experiment for a defined limit, $\boldsymbol{f}^{lim}$, of 70 N. A video associated with this experiment is available here.

When contact occurs, a similar behaviour as the one in the previous experiment can be observed only this time the disturbance is constant and against a fixed obstacle. The force applied by the robot when contact is established can be observed in Figure 4.8d.. Once the controller energy reaches its limit and $\dot{\boldsymbol{v}}_k^*$ reaches its saturation, there is a stabilization of the applied efforts to a value slightly above the limit (72 N). The observed overshoot at the instant of contact is a consequence of the energy dissipated during the impact and is unavoidable, even if the amount of energy to be dissipated is monitored. The 2 N error that can be observed during steady state is due to the imperfect model of the robot and to the approximations made in the equations developed in Section 2.3.2.2. Once contact is established, the positioning error rises to important values of more than 140 mm. Once again, during the disturbance, the pointing task is not affected by the constraint and keeps being fulfilled.

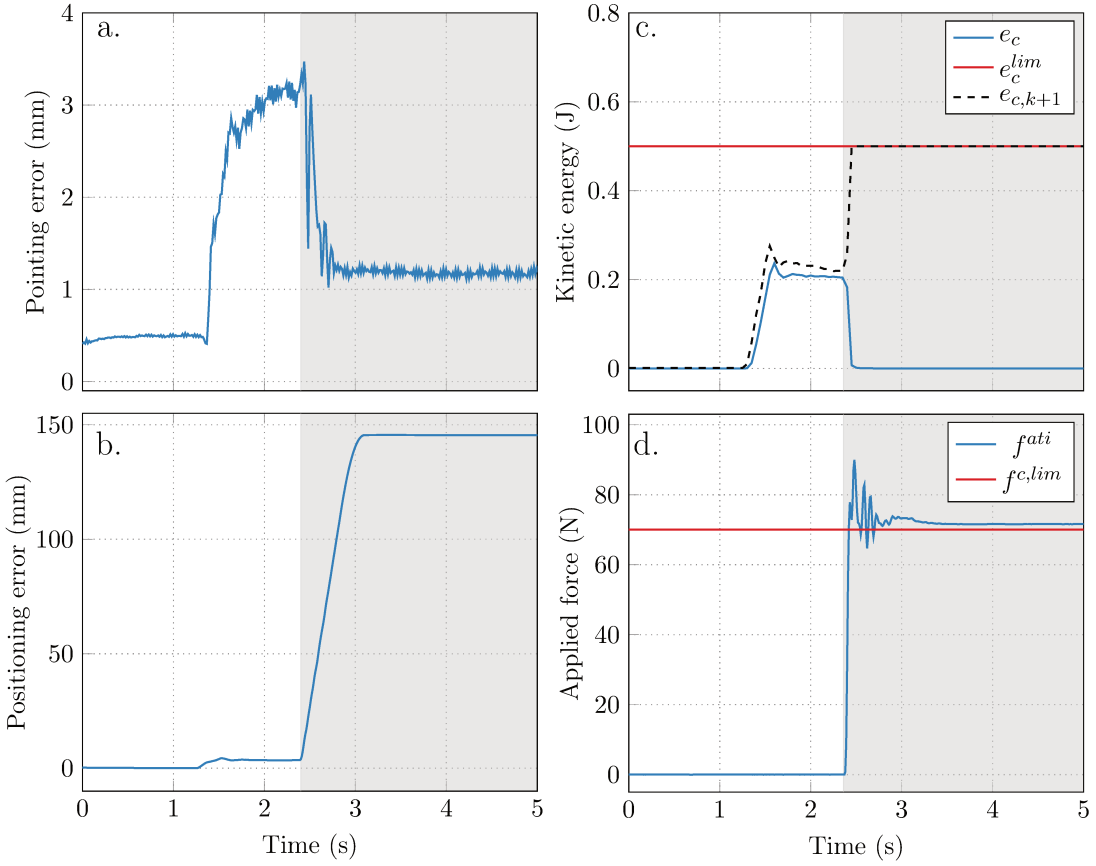


FIGURE 4.8: Evolution of (a.) the pointing error, (b.) the positioning error, (c.) the current kinetic energy (blue line), e_c^{lim} (red), the provisional kinetic energy $e_{c,k+1}$ (dashed) and (d.) the contact wrench (blue) and its limit (red). The grey areas represent the interaction phase with the measuring system.

4.2.3 The interesting properties of the pointing task

From the results presented previously one could argue that pointing towards a fixed target makes it easy for the robot to keep a small pointing error especially when it is stopped (either by hand or by the ATI sensor). However, the definition of the pointing task as a function of the robot current position yields interesting robot behaviours. Another experiment, not related to the applicative context has been conducted where the robot end-effector is now required to follow a rectangle-shaped trajectory. The target follows the same trajectory on the table plane. In the following experiment the robot must track the desired trajectory will point towards the moving target. At some point a human prevents the robot from moving in the $\boldsymbol{j}_0$-direction by pushing the end-effector with its hand.

A video associated with this experiment is available here. Figure 4.9 depicts the results of this experiment. The end-effector position is depicted in the top left corner. The desired end-effector position is represented in blue, the current position during the

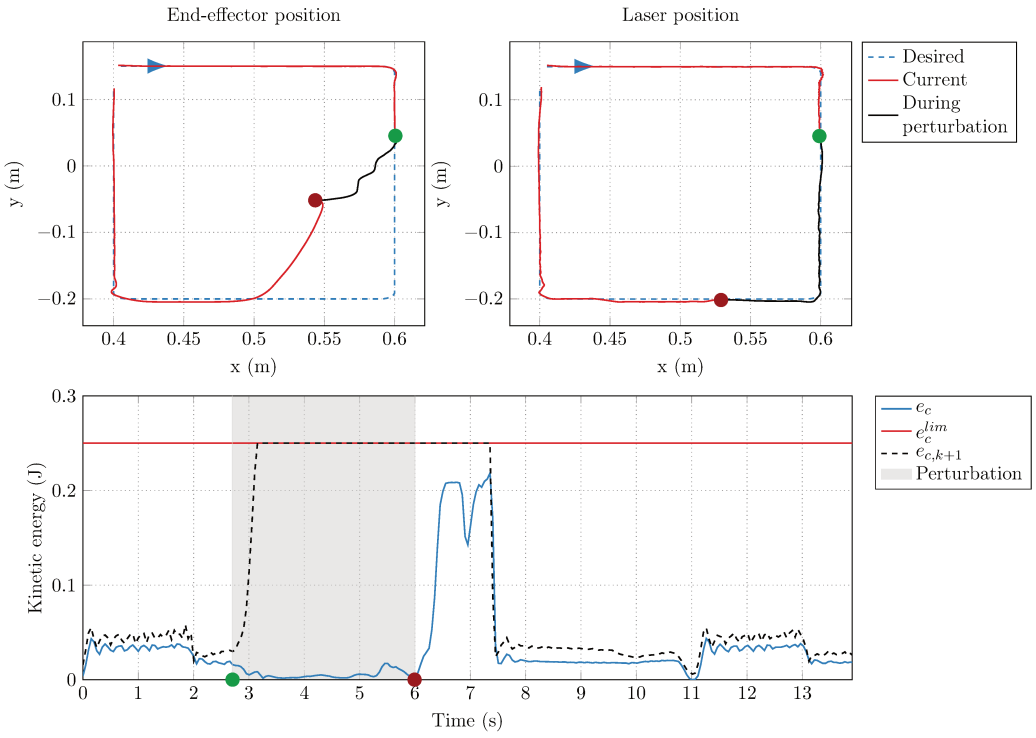


FIGURE 4.9: Evolution of the positioning task, the pointing task and the kinetic energy constraints while following a rectangle-shaped trajectory. The grey area represents the interaction phase with the robot.

motion is represented in red and the robot position during the perturbation is represented in black. The target position and the theoretical laser position are plotted in the top right corner. This theoretical position is obtained by projecting the end-effector orientation along the z-axis onto the detector plan. The last plot depicts the evolution of the robot kinetic energy (in blue), its predicted kinetic energy (in black) and its kinetic energy limit (in red).

The results are similar to the experiments in Section 4.2.2.1 only this time the pointing task behaves differently. Without the perturbation, the mean positioning error is 2.1 *mm* during the motion with a peak error of 3.7 *mm*. The mean pointing error is 2.0 *mm*. When the perturbation arises, the robot is not able to correctly follow the positioning task. It results a peak error of 167 *mm*. However, the pointing task is still performed correctly with a mean error of 2.2 *mm* and a peak error of 5.7 *mm* occurring during the release of the disturbance. This is thanks to the formulation of the pointing task according to the robot position and not according to some pre-planned trajectory. Once again, from the kinetic energy point of view it can be seen that the robot is prevented from spending an amount of energy superior to 0.25 *J*. At some point it can be seen that while the robot is pushed by the human, it performs motions along the x-axis. This is because the system follows the trajectory at best and is not restrained from moving along its x-axis. When the perturbation disappears, the robot is

free to move and reduces its error until it successfully tracks back the desired trajectory. Once again, although the robot controller is formulated as a PID, it cannot spend more than 0.25 J to reduce the tracking error, even when there is a direct contact with a human operator. When contact breaks, the kinetic energy constraint formulated inside the quadratic programming problem prevents a dangerous release of energy when trying to reduce the tracking error.

4.3 The regularisation task

When dealing with redundant robots, the regularisation task in a quadratic programming problem can be used to ensure the convergence of the solution towards a desired goal. This regularisation task can have a concrete influence on the robot behaviour. The previous experiments use the same regularisation task set to compensate for the gravity induced torque. However, other regularisation tasks can be defined to control a robot redundancy. This section presents two additional regularisation tasks, one minimizing the robot joint torque and one minimizing the robot perceived mass in a direction.

4.3.1 Torque regularisation task

The regularisation tasks minimizing the torque is given in Equation (2.51). For a given Cartesian position, only one robot configuration yields a minimum torque control input. Therefore, when an interaction occurs with the robot, the quadratic programming solver computes torque solutions to keep the robot at a specific configuration. In this experiment, the robot is required to keep a constant Cartesian position. A human interacts with the robot. The results of this experiment can be observed in Figure 4.10. The interaction phase is represented by the grey area in the figure. It results a robot behaviour where the robot is stiffly fighting against the perturbation to stay at a desired configuration. During the interaction, a small positioning error ($\leq$1.5 mm) can be observed. This error comes from the human interaction and could be reduced up to a certain amount by increasing the PID controller gains.

4.3.2 Gravity regularisation task

The previous regularization task can be compared to the gravity torque compensation regularization task. As a reminder, this regularisation task, expressed in Equation (2.52), minimizes the difference between the computed torque and the gravity induced external torque. The robot is required to keep the same Cartesian position as in the previous

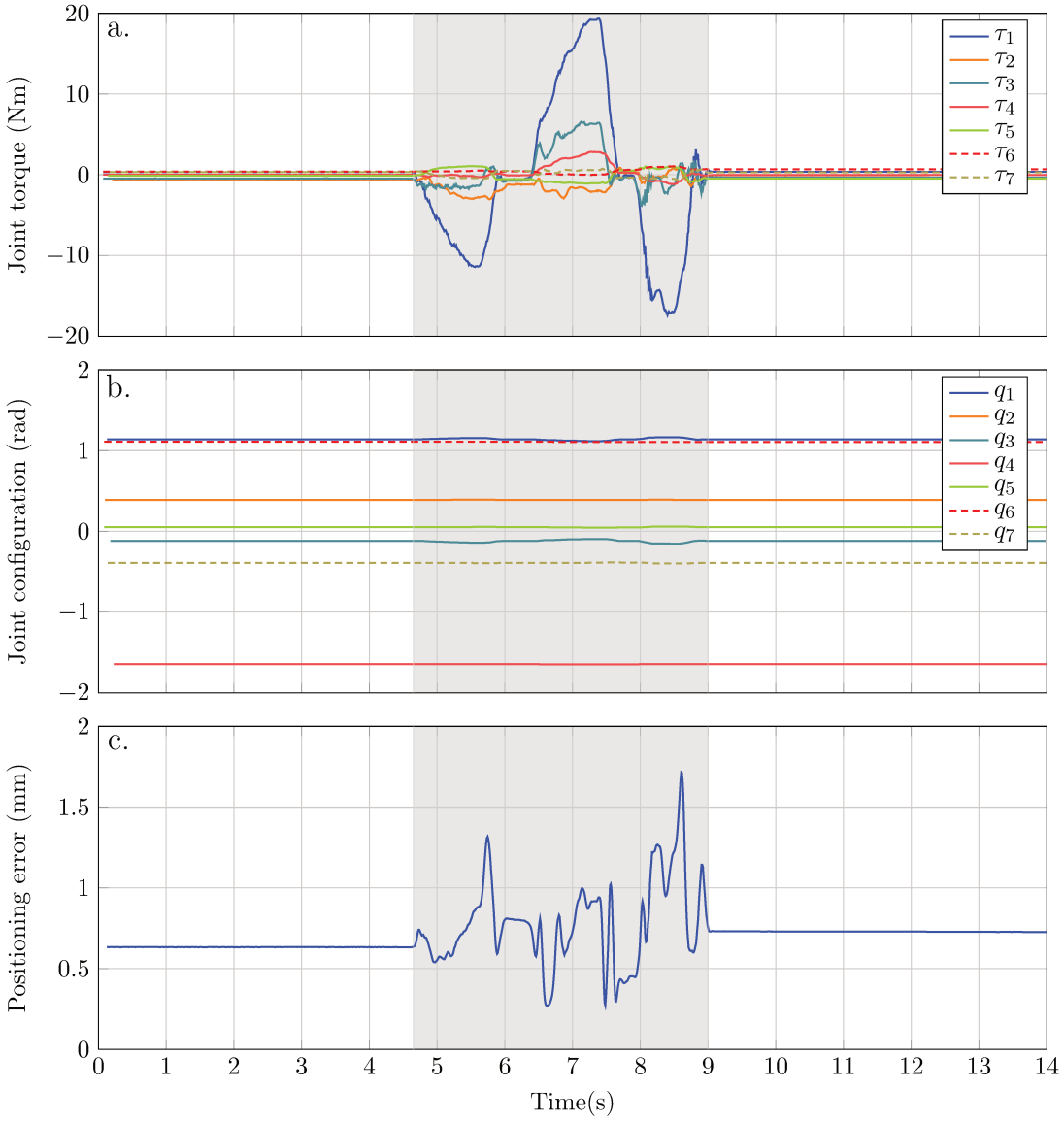


FIGURE 4.10: Evolution of (a.) the joint torque, (b.) the joint configuration and (c.) the positioning error for a regularisation minimizing the joint torques. The grey areas represent the interaction phase with a human.

experiment. Figure 4.11 shows the results of this regularization task. It can be seen that during physical interaction, the robot joints configuration changes to adapt to this interaction while maintaining the end-effector position at the same point. It results a motion in the null-space that is determined by the operator when physically interacting with the robot. Such behaviour can be interesting in co-manipulation task or when working in a shared workspace with the robot (see Table 1.1). It can be noted that noise is introduced at the joint torque level when using this regularization task. This noise is introduced when accounting for the induced gravity $g(\boldsymbol{q})$, which depends on the noisy signal coming from the robot encoders. However, this noise does not affect the joint configuration and is not strong enough to be felt by the operator.

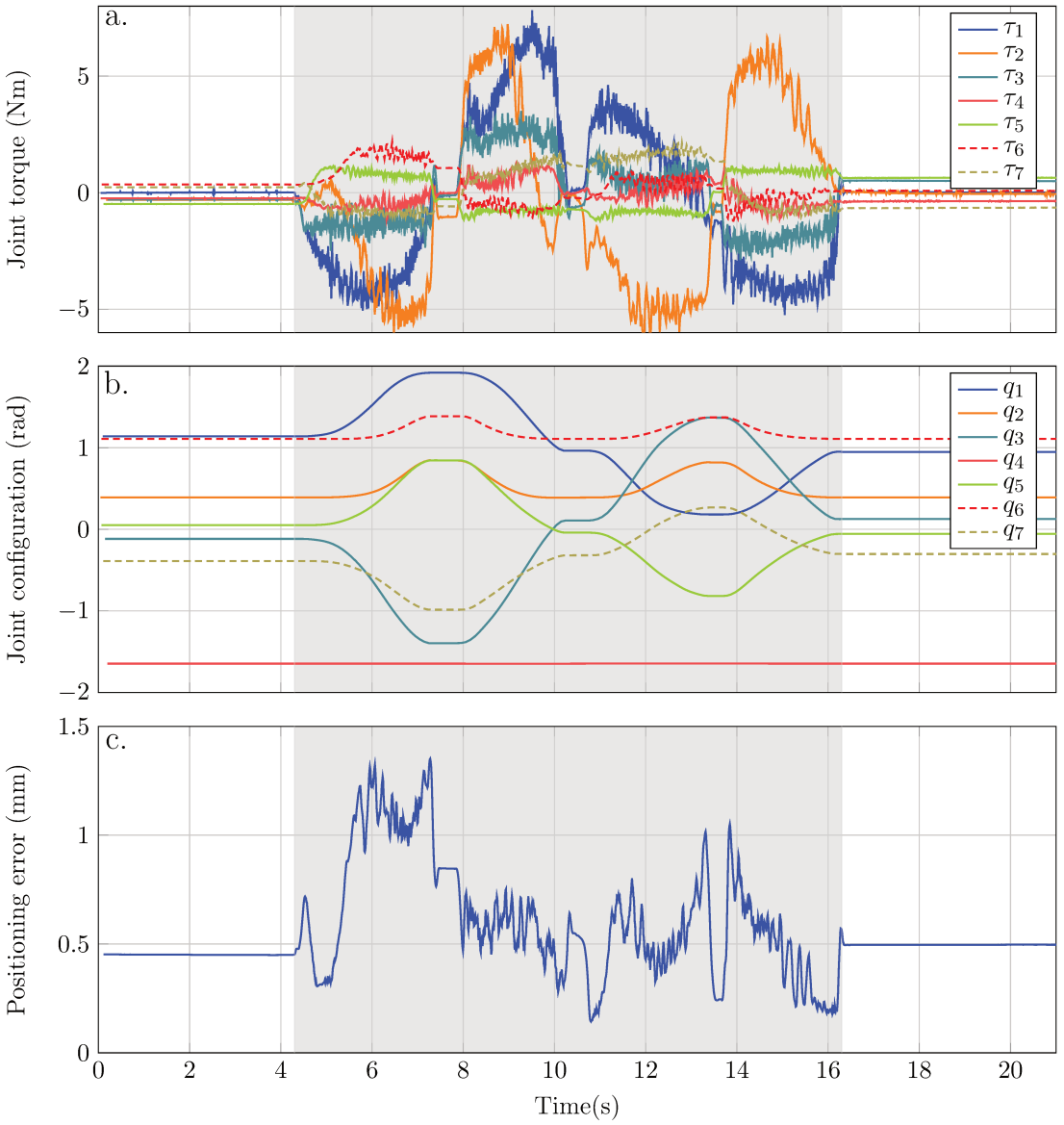


FIGURE 4.11: Evolution of (a.) the joint torque, (b.) the joint configuration and (c.) the positioning error for a regularisation minimizing the gravity induce torques. The grey areas represent the interaction phase with a human.

4.3.3 Equivalent mass minimization

As detailed in Section 2.4.5, the robot configuration can be advantageously used when controlling a redundant robot to minimize the perceived mass in a specific direction. Different experiments are conducted to determine if the local minimization can indeed influence the robot perceived mass. To that extent, the results of the local minimization algorithm proposed in Section 2.4.5.4 are compared with the results of the global minimization algorithm proposed in Section 2.4.5.3. In a second part the perceived mass minimization is performed in the direction of an obstacle rather than in the direction of

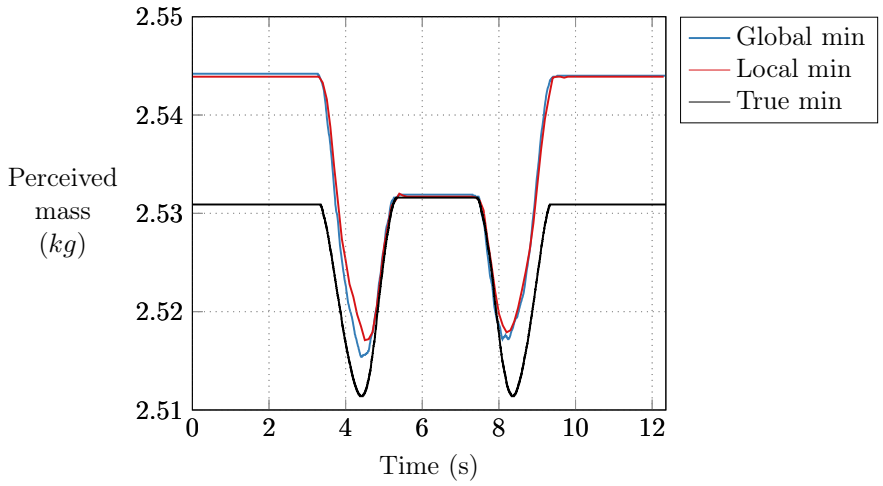


FIGURE 4.12: Comparison between the theoretical minimal perceived mass (in black), the one obtained using the impedance regularization task (in blue) and the one using the local minimization scheme

motion. This is to see if this minimization can be performed reactively while obstacles are moving around the robot.

4.3.3.1 Global optimization vs local optimization

The objective of this section is to compare the results of the global and local perceived mass minimization algorithms through the regularization task. To that extent, the robot follows a trajectory along a line and the perceived mass is minimized in the direction of motion. The results of both algorithms are compared to the theoretical perceived mass minimum along the trajectory.

The trajectory is sampled every 1 mm. The regularisation tasks are expressed at the torque level as

$$R(\boldsymbol{\tau}) = \|\boldsymbol{\tau}_{m_{\boldsymbol{u}}} - \boldsymbol{\tau}\|_2^2 \tag{4.1}$$

with $\boldsymbol{\tau}_{m_{\boldsymbol{u}}} = k_p(\boldsymbol{q}_{ns}^{des} - \boldsymbol{q}) - k_d\dot{\boldsymbol{q}}$.

The resulting perceived mass in the direction of motion induced by these regularization tasks is presented in Figure 4.12. The black curve represents the theoretical global minimum of the perceived mass in the direction of motion computed off-line according to Algorithm 2. For the blue curve, $\boldsymbol{q}_{ns}^{des}$ is obtained using the on-line part of Algorithm 2. For the red curve, $\boldsymbol{q}_{ns}^{des}$ is determined on-line using Algorithm 3.

Using a maximum number of iterations, k_{max}, of 6, the on-line local minimization algorithm (Algorithm 3) can be used in the 1 kHz control loop. It can be observed that both on-line algorithms give sensibly the same results. It means that the local minimization algorithm is fast enough to find a local minimum. However, it can be observed that the two regularisation tasks induce a robot perceived mass slightly different ($< 1\%$) compared to the theoretical mass found off-line by Algorithm 2 (in black). This is probably due to the definition of the controller gains k_p and k_d and frictions in the axes which do not allow to perfectly reach the desired configuration $\boldsymbol{q}_{ns}^{des}$.

Both algorithms can be used to minimize the robot perceived mass in the direction of motion in real-time. However, one should keep in mind that Algorithm 3 only gives a local minimum solution. It can also be noted that this algorithm does not account for the robot configuration limits when computing the desired configurations to reach. However, this is not an issue in this work since these limits are expressed as constraint in the QP problem. In the work of N. Mansfeld, an algorithm is used to prevent the solution of the algorithm to go beyond these limits [Mansfeld et al., 2017].

4.3.3.2 Reactive mass minimization with obstacles

Minimizing the mass in the direction of motion is the best approach if no information is available from the environment. However, if it is possible to know the position of an obstacle, it should be preferred to minimize the perceived mass in its direction. Indeed, it is the kinetic energy in the direction of the obstacle that should be minimized. In this experiment, the position of an obstacle is obtained using an RGBD sensor as described in section 3.5. This position is directly linked to the robot position to compute the direction in which the algorithm should minimize the perceived mass. Algorithm 3 is used to determine the robot perceived mass in the direction of the closest obstacle. The robot holds its position during the whole experiment. In the first part of the experiment, a simple regularization task, set to compensate for gravity is used. An obstacle moves around the robot and the projected mass in the direction of this obstacle is plotted. In the second part of the experiment, the regularization task set to minimize the robot perceived mass in the direction of the obstacle is used. Along the whole experiment, the positioning and pointing errors are plotted to show the influence of the regularization towards the main tasks.

Figure 4.13 depicts the results of this experiment. During the first part of the experiment, the robot perceived mass ranges between 2.6 kg and 4.3 kg. The pointing and positioning errors are correctly tracked and stable. In the second part of the experiment, it can be seen that for the same motions of the obstacle, the robot perceived mass in

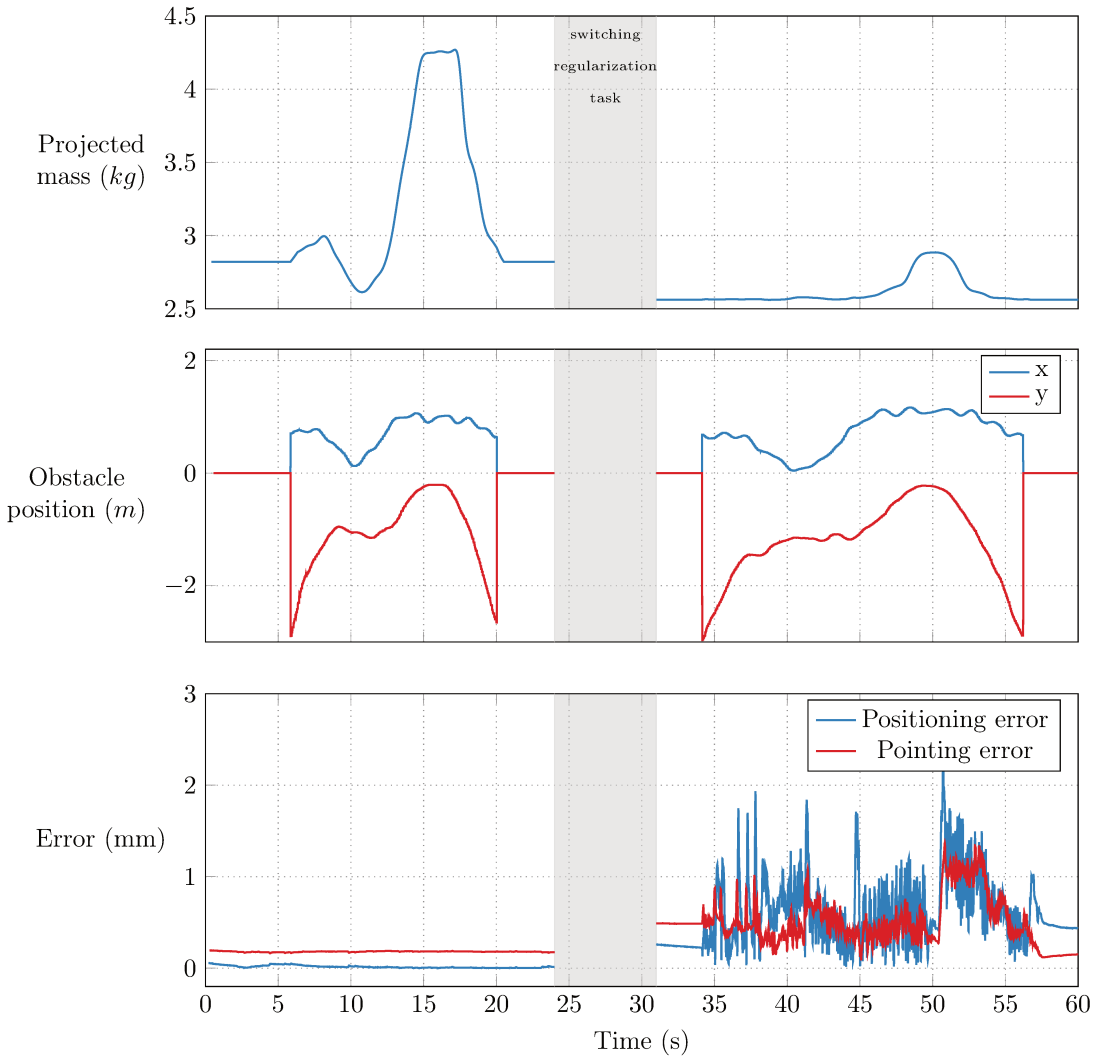


FIGURE 4.13: Reconfiguration of the robot to minimize its perceived mass in the direction of an obstacle. When the obstacle moves around the fixed robot, the perceived mass variates a lot. When the reconfiguration is set to minimize the perceived mass in the direction of the obstacle, the perceived mass greatly diminish.

the direction of the obstacle ranges between 2.5 kg and 2.8 kg which is an improvement of up to 38%. The positioning and pointing tasks are slightly disturbed by the robot reconfiguration. This is due in part to the regularization gains which do not allow to correctly reach the desired configuration $\boldsymbol{q}_{ns}^{des}$. Overall the error never goes beyond 2 mm which is acceptable for the application.

The regularisation task can thus be used to minimize the robot perceived mass. Using external sensors it is possible to minimize the perceived mass in the direction of the closest obstacle. If a contact occurs, it is the direction where the most kinetic energy will be transferred between the robot and the obstacle.

Conclusion

From these experiments it can be seen that the QP-based computed torque controller guarantees to find an optimal control solution without compromising safety. Tasks are correctly performed as long as the control variable satisfies the constraints. Both the theoretical derivation and the experimental results allow pointing out a few remarkable features of the approach.

- The controller exploits PID control with gains that are tuned independently of the safety constraint. Namely, one may choose high gains to maximize the positioning precision; since the controller accounts for safety constraints at its lowest level, using high PID gains will never result in releasing high energy when unexpectedly colliding with an obstacle.
- When a contact has been established with an unexpected obstacle, the wrench is limited to a maximal constant value, independent of the obstacle position. In other words, up to the controller bandwidth, the robot exhibits a null impedance and can be moved without adding extra force.
- The approach requires no more specific controller tuning than those of the PID compensators.
- This implementation of safety constraints does not require any collision detection algorithm nor any switching between different control modes.
- A natural property of the QP solver approach is to ensure constraints priority over tasks. A task requiring too much kinetic energy is degraded (the desired value is not followed any more) while the other tasks continue to be correctly fulfilled. As a practical result, it is experimentally demonstrated that the pointing task towards the target was always fulfilled even when an obstacle prevented the robot from being able to bring the end-effector at its desired value.

The QP formulation of the problem also allows accounting for the robot redundancy. Different regularisation tasks have been used to minimize the joint torques induced by gravity or to minimize the robot perceived mass in a direction. In the first case, the robot realises its main tasks optimally and it is possible to physically interact with it to perform inner motions. In the latter case, this gives the ability to reduce the robot dangerousness in the direction of an obstacle by reducing the kinetic energy that can be transferred in case of transient contact. The experimental results obtained in this chapter showed the ability to use this concept in real time on a 7 dof robot.

The next chapter concludes this work and presents research perspectives.

Chapter 5

Conclusion

Robots are being used in a growing number of applications. Still, there are applicative contexts where the lack of safety with respect to their environment prevents them from being used. This is mainly the case when robots share their workspace with human beings. When robots have to perform autonomous motions in an unknown environment, they must consider several key aspects presented in Table 1.1 which are:

- the respect of constraints imposed either by the laws of physics or by the environment,
- the optimal achievement of their tasks,
- ensure, at each control instant, that the robot is not dangerous towards its environment.

These key aspects are closely linked. In fact, in a reactive environment, it is not possible to consider one of these aspects without considering the others.

Safety is the most important point for the development of more autonomous robotic applications. The robotic field has seen many contributions to obstacle avoidance or to detection and reaction to a collision. However, the design of a robust control solution ensuring, at each time step, that an undesired contact with a robot would not be dangerous, is yet to be proposed and used in concrete applications. To that aim, the ISO TS 15066 proposes to define safety limits characterizing a robot dangerousness. It divides contact in two classes: transient contact and quasi-static contact. The first type of contact corresponds to an impact between a moving robot and a still human. The

dangerousness of such contact comes from the transfer of energy between the two bodies. The second type of contact corresponds to a robot crushing some part of the body against a fix obstacle. The dangerousness of this second type of contact comes from the forces exerted by the robot on a human body part.

5.1 Contributions

In this work, a robot variation of kinetic energy is used as a unified indicator of its dangerousness. This variation of energy links both the forces exerted by the robot and the kinetic energy that it displays when actuated. To improve the safety of people evolving in the robot environment, this work proposes to both minimize this energy and constrain it to a safe limit. This work is based on the fact that a robot kinetic energy is a function of the inertial properties of each body constituting the robot and its velocity. For redundant robots, one can find a joint configuration minimizing the robot inertial properties in some direction. If the robot velocity is specified by the task, minimizing the robot mass is the only way to minimize its kinetic energy through control. To constrain the robot kinetic energy, specific control methods must be used.

To implement these features, the proposed controller is expressed as a quadratic programming optimization. The solution of the optimization problem gives the optimal control input to realise the tasks while satisfying the specified constraints. Safety is thus expressed as a constraint within the quadratic programming problem. Furthermore, it is possible to define second priority tasks inside a quadratic programming problem. One can thus define a task using the robot redundancy to minimize its mass in a specific direction.

The theoretical aspects of this controller are treated in Chapter 2 and tested on a KUKA LWR4+ robot in Chapter 4. Even though this controller can be applied for any robot tasks, this work considers a specific applicative context: 3D X-ray imaging. In this context, the robot tasks consist in positioning the robot in 3D space and pointing towards a target. Using the external sensors presented in Chapter 3, the experiments realised in this work show the ability to correctly perform tasks while satisfying the constraints. It is also shown that the control scheme does not depend on the trajectory generation algorithm. Therefore, new robot positions can be defined on-line based on the presence of objects in the robot workspace.

In the second part of Chapter 4, the kinetic energy constraint is evaluated on several experiments. The quadratic programming solver is able to find control solutions keeping the robot kinetic energy under a specified threshold. An external platform is designed to

measure the actual energy dissipated during contact. The results given by the platform show that the robot kinetic energy is indeed constrained to a defined (safe) limit. Experiments are also conducted with a human physically interacting with the robot. When a transient contact is initiated, the controller constrains the robot kinetic energy so that it does not release a dangerous amount of energy both during contact and when the contact is released. Furthermore, during quasi-static contact, the constraint provides a way to limit the wrenches applied by the robot without doing force control or using external sensors. In both cases contact, collision detection algorithms are not necessary to ensure that the robot adapts its behaviour.

In the last part of Chapter 4, the robot redundancy is used to minimize its perceived mass in the direction of an obstacle. Using local optimization techniques, the experiments show that it is possible to find in real-time a configuration minimizing the robot perceived mass in the direction of an obstacle. The direction of the obstacle can be recorded on-line by a RGBD sensor. It is thus possible to dynamically reduce the robot kinetic energy to a minimal value without modifying the main task objectives in terms of precision and speed.

Referring to Table 1.1, the quadratic programming based controller ensures the optimal realisation of a set of given tasks, either defined off-line or update on-line, while satisfying the robot constraints. Combining the kinetic energy constraint and the mass minimization algorithm, the controller developed in this work offers a new way to improve robot safety in a shared workspace. The kinetic energy constraint is an additional layer of safety taking action after the servoing phase. This safety layer ensures that the robot task does not use more kinetic energy than the defined limits. This limit can be adapted on-line and defined using safety recommendation given by international norms.

5.2 Limitations and perspectives

The control approach proposed in this work yields interesting properties. However, it relies on numerous assumptions that could be more thoroughly explored. During transient contact, the kinetic energy transferred between the robot and the colliding object must be constrained. In the proposed controller, the current kinetic energy constraint voluntarily omits to consider the kinetic energy of the colliding object. This is because it can be difficult to determine the relative mass and velocity of this obstacle. Consequently, the proposed approach underestimates the kinetic energy that should be constrained from the robot side to ensure a safe collision. In the case of a human obstacle, one could try to estimate the mass of the body part entering in collision, using for example the mean value according to some reference data *e.g.* [Dumas et al., 2007]. The speed of the

human can be obtained using external sensors such as the RGBD cameras used in this work. These estimations would induce a more restrictive but safer energetic constraint.

The expression of the kinetic energy constraint should be independent of the PID controller tuning. In our proposed approach this is not entirely the case as the variable Δt depends on the gain and the saturation limits of the PID controller. Expressing the constraint in its quadratic form would alleviate this problem. However, from a computation point of view, the problem would not be solvable at 1 kHz with the optimization softwares and the computation hardware that are currently available.

The minimization of the robot perceived mass yields interesting results. However, it suffers the common limitation of local minimization, *i.e.* the possible convergence towards a local minimum. This minimization can also lead to configurations that are not optimal for the robot motions. For example the minimum could be close to joint limits. Since the controller constrains the joint position, this is not much of an issue. However, being close to the joint limits is not always recommended and can result to unpractical robotic configurations. Other criterion such as the robot manipulability should be considered when defining a desired robot configuration. However, this requires to establish a heuristic trade-off to regulate the relative importance of each criterion. In this case this could have an impact on the robot energy and thus on its dangerousness.

During the experimental trials, the controller showed a stable behaviour. However, no stability analysis has been performed yet. Even though no formal demonstration has been realised, this work can be linked to passivity through its approach on energy. Indeed, a system is said to be passive if and only if the energy entering the system is greater than the energy going out [Hannaford and Ryu, 2002]. Passivity is a sufficient condition for stability. The energy entering the system being constrained in this work, one could therefore use passivity methods to demonstrate its stability.

Bibliography

Albu-Schäffer, A., ., Haddadin, S., Ott, C., Stemmer, A., Wimböck, T., and Hirzinger, G. (2007). The DLR lightweight robot: design and control concepts for robots in human environments. *Industrial Robot: An International Journal*, 34(5):376–385. *4 citations page 2, 15, 18, and 64*

Albu-Schäffer, A., Wolf, S., Eiberger, O., Sami, H., Petit, F., and Chalon, M. (2010). Dynamic Modelling and Control of Variable Stiffness Actuators. In *IEEE International Conference on Robotics and Automation*, pages 2155–2162. *Cited page 16*

Alvarez-Ramirez, J., Santibañez, V., and Campa, R. (2008). Stability of robot manipulators under saturated PID compensation. *IEEE Transactions on Control Systems Technology*, 16(6):1333–1341. *Cited page 45*

Baerlocher, P. and Boulic, R. (2004). An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, 20(6):402–417. *Cited page 29*

Baillieul, J (1985). Kinematic programming alternatives for redundant manipulators. In *IEEE International Conference on Robotics and Automation*, pages 722–728. *Cited page 29*

BGIA (2011). BG / BGIA risk assessment recommendations according to machinery directive Design of workplaces with collaborative robots. Technical Report February, Institute for Occupational Safety and Health of the German Social Accident Insurance. *Cited page 13*

Bicchi, A. and Tonietti, G. (2004). Fast and "soft-arm" tactics. *IEEE Robotics and Automation Magazine*, 11(2):22–33. *2 citations page 2 and 16*

Bohn, C. and Atherton, D. P. (1995). An Analysis Package Comparing PID Anti-Windup Strategies. *IEEE Control Systems*, 15(2):34–40. *Cited page 36*

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press. *Cited page 33*

Bright, G. and Deubler, C. (1999). Design and Implementation of an Intelligent Remote Centre Compliance (IRCC) as a Means of Intelligent Position Feedback for a Construction Robot. In *ISARC*, pages 719–723. *Cited page 15*

Briquet-Kerestedjian, N., Makarov, M., Grossard, M., and Rodriguez-Ayerbe, P. (2017). Stochastic observer design for robot impact detection based on inverse dynamic model under uncertainties. In *IFAC 2017 - 20th World Congress of the International Federation of Automatic Control*, Toulouse, France. *Cited page 18*

Bruyninckx, H. (2002). OROCOS: design and implementation of a robot control software framework. In *IEEE International Conference on Robotics and Automation - Tutorial*. *Cited page 71*

Chatelain, P., Krupa, A., and Navab, N. (2017). Confidence-Driven Control of an Ultrasound Probe. *IEEE Transactions on Robotics*, 33(6):1410–1424. *Cited page 64*

Chen, Y. C. and Walker, I. D. (1993). A consistent null-space based approach to inverse kinematics of redundant robots. In *IEEE International Conference on Robotics and Automation*, pages 374–381. *Cited page 50*

Cherubini, A., Passama, R., Crosnier, A., Lasnier, A., and Fraisse, P. (2016). Collaborative manufacturing with physical human–robot interaction. *Robotics and Computer-Integrated Manufacturing*, 40:1 – 13. *Cited page 64*

Cherubini, A., Passama, R., Meline, A., Crosnier, A., and Fraisse, P. (2013). Multimodal control for human-robot cooperation. In *IEEE International Conference on Intelligent Robots and Systems*, pages 2202–2207. *Cited page 18*

De Luca, A., Albu-Schaffer, A., Haddadin, S., and Hirzinger, G. (2006). Collision Detection and Safe Reaction with the DLR-III Lightweight Manipulator Arm. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1623–1630. *2 citations page 2 and 18*

De Luca, A. and Ferrajoli, L. (2008). Exploiting robot redundancy in collision detection and reaction. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3299–3305. *Cited page 18*

Drake, S. (1977). *Using Compliance in Lieu of Sensory Feedback for Automatic Assembly*. PhD thesis, MIT. *Cited page 15*

Dumas, R., Cheze, L., and Verriest, J.-P. (2007). Adjustments to mcconville et al. and young et al. body segment inertial parameters. *Journal of biomechanics*, 40:543–53. *Cited page 95*

Esteveny, L., Barbe, L., and Bayle, B. (2014). A novel actuation technology for safe physical human-robot interactions. In *IEEE International Conference on Robotics and Automation*, pages 5032–5037. *Cited page 16*

Faverjon, B. and Tournassoud, P. (1987). A local based approach for path planning of manipulators with a high number of degrees of freedom. In *IEEE International Conference on Robotics and Automation*. *Cited page 34*

Ferraguti, F., Secchi, C., and Fantuzzi, C. (2013). A tank-based approach to impedance control with variable stiffness. In *International Conference on Robotics and Automation*, pages 4948–4953. *Cited page 20*

Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G., and Diehl, M. (2014). qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363. *2 citations page 33 and 71*

Ficuciello, F., Villani, L., and Siciliano, B. (2016). Impedance Control of Redundant Manipulators for Safe Human-Robot Collaboration. *Acta Polytechnica Hungarica Journal of Applied Sciences*, 13(1):223–238. *Cited page 64*

Flacco, F. (2013). Optimal Redundancy Resolution with Task Scaling under Hard Bounds in the Robot Joint Space. In *IEEE International Conference on Robotics and Automation*, pages 3969–3975. *Cited page 30*

Flacco, F., Kroeger, T., De Luca, A., and Khatib, O. (2014). A Depth Space Approach for Evaluating Distance to Objects. *Journal of Intelligent & Robotic Systems*, 80(1):7–22. *2 citations page 17 and 29*

Flacco, F., Luca, A. D., and Khatib, O. (2012). Motion Control of Redundant Robots under Joint Constraints: Saturation in the Null Space. In *IEEE International Conference on Robotics and Automation*. *Cited page 30*

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33. *Cited page 2*

Haddadin, S., Albu-Schäffer, A., Frommberger, M., and Hirzinger, G. (2008a). The role of the robot mass and velocity in physical human-robot interaction - Part II: Constrained blunt impacts. In *IEEE International Conference on Robotics and Automation*, pages 1339–1345. *Cited page 11*

Haddadin, S., Albu-Schäffer, A., and Hirzinger, G. (2008b). The role of the robot mass and velocity in physical human-robot interaction - Part I: Non-constrained blunt impacts. In *IEEE International Conference on Robotics and Automation*, pages 1331–1338. *2 citations page 9 and 11*

Haddadin, S., De Luca, A., and Albu-Schäffer, A. (2017). Robot collisions: A survey on detection, isolation, and identification. *IEEE Transactions on Robotics*, 33(6):1292–1312. *Cited page 18*

Haddadin, S., Schaffer, A., and Hirzinger, G. (2007). Safety Evaluation of Physical Human-Robot Interaction via Crash-Testing. In *Robotics: Science and Systems Conference (RSS 2007)*, pages 395–407. *Cited page 11*

Haddadin, S., Suppa, M., Fuchs, S., Bodenmueller, T., Albu-Schäffer, A., and Hirzinger, G. (2011). Towards the robotic co-worker. *Robotics Research*, 70:261–282. *Cited page 64*

Hannaford, B. and Jee-Hwan, R. (2000). Time Domain Passivity Control of Haptic Interfaces. *IEEE Transactions on Robotics and Automation*, 18(1):1–39. *Cited page 19*

Hannaford, B. and Ryu, J.-H. (2002). Time Domain Passivity Control of Haptic Interfaces. *Transactions on Robotics and Automation*, 18(1):1–10. *Cited page 96*

Haque, M. A., Ahmad, M. O., Swamy, M. N. S., Hasan, M. K., and Lee, S. Y. (2013). Adaptive projection selection for computed tomography. *IEEE Transactions on Image Processing*, 22(12):5085–5095. *Cited page 8*

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions of systems science and cybernetics*, 4(2):100–107. *Cited page 1*

Hogan, N. (1984). Impedance Control : An Approach to Manipulation. In *American Control Conference*. *Cited page 19*

IEC (2017). Lbr iiwa med test certificate. https://www.kuka.com/-/media/kuka-downloads/imported/9cb8e311bfd744b4b0eab25ca883f6d3/lbr-med-cb-test-certificate.pdf. *Cited page 64*

ISO/TS-13849 (2015). *Safety of machinery – Safety-related parts of control systems.* International Organization for Standardization, Geneva, Switzerland. *Cited page 70*

ISO/TS-15066 (2016). *Robots and robotic devices - Collaborative robots.* International Organization for Standardization, Geneva, Switzerland. *2 citations page 10 and 13*

Jakopec, M., Rodriguez, F., Harris, S. J., Gomes, P., Cobb, J., and Davies, B. L. (2003). The Hands-On Orthopaedic Robot " Acrobot ": Early Clinical Trials of Total Knee Replacement Surgery. *IEEE Transactions on Robotics and Automation*, 19(5):902–911. *Cited page 6*

Jiang, B. C. and Gainer, C. A. (1987). A cause-and-effect analysis of robot accidents. *Journal of Occupational Accidents*, 9(1):27–45. *Cited page 1*

Jochen, H. and Zelinsky, A. (2003). Quantitative Safety Guarantees for Physical Human – Robot Interaction. *The International Journal of Robotics Research*, 22(7):479–504. *Cited page 18*

Joly, L. D. and Andriot, C. (1995). Imposing motion constraints to a force reflecting telerobot through real-time simulation of a virtual mechanism. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 357–362. *Cited page 40*

Jubien, A., Gautier, M., and Janot, A. (2014). Dynamic identification of the Kuka LightWeight robot: Comparison between actual and confidential Kuka's parameters. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, pages 483–488. *Cited page 65*

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., and Schaal, S. (2011). STOMP: Stochastic trajectory optimization for motion planning. In *IEEE International Conference on Robotics and Automation*, pages 4569–4574. *Cited page 35*

Kapoor, A., Li, M., and Taylor, R. H. (2006). Constrained Control for Surgical Assistant Robots. In *International Conference on Robotics and Automation*, pages 231–236. *2 citations page 34 and 40*

Khatib, O. (1986). Real time obstacle avoidance for manipulators and mobile robots. In *IEEE International Journal of Robotics and Research*, pages 90–98. *2 citations page 17 and 29*

Khatib, O. (1995). Inertial properties in robotic manipulation: An object-level framework. *International Journal of Robotics and Research*, 14(1):19–36. *Cited page 12*

Khatib, O. and Siciliano, B. (2008). *Springer handbook of robotics*. Springer. *Cited page 35*

Khosla, P. (1987). Choosing sampling rates for robot control. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*. *Cited page 34*

Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154. *2 citations page 63 and 71*

Koganezawa, K. (2005). Mechanical stiffness control for antagonistically driven joints. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1544–1551. *Cited page 16*

Kronander, K. and Billard, A. (2016). Stability Considerations for Variable Impedance Control. *IEEE Transactions on Robotics*, 32(5):1298–1305. *Cited page 19*

Kuindersma, S., Permenter, F., and Tedrake, R. (2014). An efficiently solvable quadratic program for stabilizing dynamic locomotion. In *IEEE International Conference on Robotics and Automation*, pages 2589–2594. *Cited page 34*

Kunz, T. and Stilman, M. (2012). Time-Optimal Trajectory Generation for Path Following with Bounded Acceleration and Velocity. In *Robotics: Science and Systems*, pages 1–7. *Cited page 35*

Laffranchi, M., Tsagarakis, N. G., and Caldwell, D. G. (2009). Safe human robot interaction via energy regulation control. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 35–41. *Cited page 18*

Lasota, P. A., Rossano, G. F., and Shah, J. A. (2014). Toward safe close-proximity human-robot interaction with standard industrial robots. In *IEEE International Conference on Automation Science and Engineering*, pages 339–344. *Cited page 18*

Lavalle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical report, Computer Science Dept., Iowa State University. *2 citations page 2 and 35*

Liégeois, A. (1977). Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics*, 7(12):868–871. *Cited page 28*

Liu, M., Tan, Y., and Paddois, V. (2016). Generalized hierarchical control. *Autonomous Robots*, 40(1):17–31. *Cited page 33*

Maciejewski, A. and Klein, C. (1985). Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments. *The International Journal of Robotics Research*, 4(3):109–117. *Cited page 29*

Mansfeld, N., Djellab, B., Rald, J., Beck, F., Ott, C., and Haddadin, S. (2017). Improving the Performance of Biomechanically Safe Velocity Control for Redundant Robots through Reflected Mass Minimization. In *IEEE International Conference on Intelligent Robots and Systems*, pages 5390–5397. *2 citations page 50 and 90*

Matthias, B. and Reisinger, T. (2016). Example Application of ISO / TS 15066 to a Collaborative Assembly Scenario Summary / Abstract. In *International Conference in Competitive Manufacturing*, pages 88–92. *Cited page 17*

Meguenani, A. (2018). *Safe Control of Robotic Manipulators in Dynamic Contexts*. PhD thesis, Sorbonnes Universitée. *Cited page 40*

Meguenani, A., Paddois, V., Da Silva, J., Hoarau, A., and Bidaud, P. (2017). Energy-based control for safe human-robot physical interactions. In *International Symposium on Experimental Robotics*, pages 809–818. *4 citations page vi, 19, 34, and 71*

Meguenani, A., Padois, V., and Bidaud, P. (2015). Control of robots sharing their workspace with humans: An energetic approach to safety. In *IEEE International Conference on Intelligent Robots and Systems*, pages 4678–4684. *Cited page 18*

Nguyen, K. D., Ng, T. C., and Chen, I. M. (2008). On Algorithms for Planning S-Curve Motion Profiles. *International Journal of Advanced Robotic Systems*, 5(1):99–106. *Cited page 35*

Oberer, S., Malosio, M., and Schraft, R. D. (2006). Investigation of robot human impact. In *International Symposium on Robotics*, volume 1956, page 87. *Cited page 9*

Oberer, S., Schraft, R. D., and Ipa, F. (2007). Robot-Dummy Crash Tests for Robot Safety Assessment. In *IEEE International Conference on Robotics and Automation*, pages 2934–2939. *Cited page 11*

Ott, C. (2008). *Cartesian Impedance Control: The Rigid Body Case*, pages 29–44. Springer Berlin Heidelberg, Berlin, Heidelberg. *Cited page 51*

Padois, V., Chiron, P., and Fourquet, J.-Y. (2004). Controlling dynamic contact transition for nonholonomic mobile manipulators. In *International Conference on Intelligent Robots and Systems*, pages 3817–3822. *Cited page 49*

Park, J. J., Haddadin, S., Song, J. B., and Albu-Sch??ffer, A. (2011). Designing optimally safe robot surface properties for minimizing the stress characteristics of Human-Robot collisions. In *IEEE International Conference on Robotics and Automation*, pages 5413–5420. *Cited page 15*

Penrose, R. (1955). A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413. *Cited page 28*

Poquet, C., Mozer, P., Vitrani, M. A., and Morel, G. (2015). An endorectal ultrasound probe comanipulator with hybrid actuation combining brakes and motors. *IEEE/ASME Transactions on Mechatronics*, 20(1):186–196. *Cited page 5*

Pratt, G. and Williamson, M. (1995). Series elastic actuators. In *International Conference on Intelligent Robots and Systems*, volume 1, pages 399–406. *Cited page 16*

Raiola, G., Cardenas, C. A., Tadele, T. S., de Vries, T., and Stramigioli, S. (2018). Development of a safety- and energy-aware impedance controller for collaborative robots. In *IEEE Robotics and Automation Letters*, pages 1237–1244. *Cited page 19*

Ratliff, N., Zucker, M., Bagnell, J. A., and Srinivasa, S. (2009). CHOMP: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation*, pages 489–494. *Cited page 35*

Ren, J., McIsaac, K. A., and Patel, R. V. (2006). Modified Newton's method applied to potential field-based navigation for mobile robots. *IEEE Transactions on Robotics*, 22(2):384–391. *Cited page 17*

Riwan, A., Giudicelli, B., Taha, F., Lazennec, J.-Y., Sahbani, A. andKilian, P., Jabbour, Z., VanRhijn, J., Louveau, F., Morel, G., Francoise, V., Armand, D., and Lavallée, S. (2011). Surgicobot project: Robotic assistant for spine surgery. *IRBM Ingéniérie et recherche Biomédicale*, 32(2):130–136. *Cited page 6*

Rubrecht, S. (2011). *Contributions à la commande de robots sous contraintes*. PhD thesis. *Cited page 29*

Rubrecht, S., Paddois, V., Bidaud, P., and De Broissia, M. (2010). Constraints compliant control: Constraints compatibility and the displaced configuration approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 677–684. *Cited page 40*

Rubrecht, S., Padois, V., Bidaud, P., De Broissia, M., and Da Silva Simoes, M. (2012). Motion safety and constraints compatibility for multibody robots. *Autonomous Robots*, 32(3):333–349. *Cited page 3*

Schreiber, G., Stemmer, A., and Bischoff, R. (2010). The fast research interface for the kuka lightweight robot. *Cited page 71*

Schweikard, A., Shiomi, H., and Adler, J. (2004). Respiration tracking in radiosurgery. *Medical Physics*, 31(10):2738–2741. *Cited page 7*

Shin, D., Sardellitti, I., Park, Y.-L., Khatib, O., and Cutkosky, M. (2010). Design and control of a bio-inspired human-friendly robot. *The International Journal of Robotics Research*, 29(5):571–584. *Cited page 16*

Siciliano, B. and Slotine, J.-J. (1991). A general framework for managing multiple tasks in highly redundant robotic systems. In *Advanced Robotics 'Robots in Unstructured Environments'*, pages 1211–1216. *Cited page 28*

Smits, R. (2018). KDL: Kinematics and Dynamics Library. `http://www.orocos.org/kdl`. *Cited page 74*

Stasse, O., Escande, A., Mansard, N., Miossec, S., Evrard, P., and Kheddar, A. (2008). Real-Time (Self)-Collision Avoidance Task on a HRP-2 Humanoid Robot. In *IEEE International Conference on Robotics and Automation*, pages 3200–3205. *Cited page 29*

States, J. D. (1969). The abbreviated and the comprehensive research injury scales. *SAE Transactions*, 78(4):2625–2634. *Cited page 9*

Stayman, J. W. and Siewerdsen, J. H. (2013). Task-Based Trajectories in Iteratively Reconstructed Interventional Cone-Beam CT. In *International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine*, pages 257–260. *Cited page 8*

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. (2017). OSQP: An operator splitting solver for quadratic programs. *ArXiv e-prints*. *Cited page 34*

Suita, K., Yamada, Y., Tsuchida, N., Imai, K., Ikeda, H., and Sugimoto, N. (1995). A failure-to-safety "Kyozon" system with simple contact detection and stop capabilities for safe human-autonomous robot coexistence. In *IEEE International Conference on Robotics and Automation*, pages 3089–3096. *Cited page 15*

Vanderborght, B., Bicchi, A., Burdet, E., Caldwell, D. G., Carloni, R., Catalano, M., Eiberger, O., Friedl, W., Ganesh, G., Garabini, M., Grebenstein, M., Grioli, G., Haddadin, S., Hoppner, H., Jafari, A., Laffranchi, M., Lefeber, D., Petit, F., Stramigioli, S., Tsagarakis, N., Damme, M. V., Ham, R. V., Visser, L. C., and Wolf, S. (2013). Variable impedance actuators: A review. *Robotics and Autonomous Systems*, 61(12):1601–1614. *Cited page 16*

Vanderborght, B., Verrelst, B., Ham, R. V., Damme, M. V., Lefeber, D., Duran, B. M. Y., and Beyl, P. (2006). Exploiting natural dynamics to reduce energy consumption by controlling the compliance of soft actuators. *The International Journal of Robotics Research*, 25(4):343–358. *Cited page 16*

Versace, J. (1971). A review of the severity index. In *SAE Technical Paper*. SAE International. *Cited page 10*

Vitrani, M. (2006). *Asservissement visuel à partir d'images échographiques. Application à la chirurgie intra-cardiaque*. These, Université Pierre et Marie Curie, Paris 6, ISIR. *Cited page 60*

Vogel, C., Walter, C., and Elkmann, N. (2013). A projection-based sensor system for safe physical human-robot collaboration. In *IEEE International Conference on Intelligent Robots and Systems*, pages 5359–5364. *Cited page 18*

Walker, I. D. (1994). Impact Configurations and Measures for Kinematically Redundant and Multiple Armed Robot Systems. *IEEE Transactions on Robotics and Automation*, 10(5):670–683. *2 citations page 12 and 49*

Wassink, M. and Stramigioli, S. (2007). Towards a novel safety norm for domestic robotics. In *IEEE International Conference on Intelligent Robots and Systems*. *Cited page 11*

Watson, P. C. (1976). Remote center compliance system. *Cited page 15*

Wolf, S. and Albu-Schäffer, A. (2013). Towards a Robust Variable Stiffness Actuator. In *IEEE International Conference on Intelligent Robots and Systems*, pages 5410–5417. *Cited page 16*

Wolfe, P. (1969). Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235. *Cited page 54*

Yamada, Y., Hirasawa, Y., Huang, S., Umetani, Y., and Suita, K. (1997). Human-robot contact in the safeguarding space. *IEEE/ASME Transactions on Mechatronics*, 2(4):230–236. *Cited page 18*

Zghal, H., Dubey, R., and Euler, J. (1990). Efficient gradient projection optimization for manipulators with multiple degrees of redundancy. In *IEEE International Conference on Robotics and Automation*, pages 1006–1011. *Cited page 50*

Ziegler, J. G. and Nichols, N. B. (1942). Optimum settings for automatic controllers. *Transaction of the A.S.M.E*, 64(1):759–765. *Cited page 74*

Appendices

Appendix A

Laser calibration

To measure the robot pointing error as defined in Equation (3.18), a laser module is mounted on the robot end-effector. A frame, $\mathcal{F}_s\left(\boldsymbol{X}_s, (\boldsymbol{i}_s, \boldsymbol{j}_s, \boldsymbol{k}_s)\right)$ is attached to the laser module, with $\boldsymbol{k}_s$ the direction of the laser beam and $\boldsymbol{X}_s = (x_s, y_s, z_s)$. The light emitted by the laser intersects with a quadrant photodiode surface on which a frame $\mathcal{F}_d\left(\boldsymbol{X}_d, (\boldsymbol{i}_0, \boldsymbol{j}_0, \boldsymbol{k}_0)\right)$ is attached, with $\boldsymbol{X}_d = (x_d, y_d, z_d)$ the centre of the photodiode detecting surface. This device records the position of the laser spot on its detection surface. The accurate position of this quadrant photodiode relatively to the robot base frame, as well as the transformation from the end-effector frame to the laser frame are two key elements in the determination of the pointing error. This appendix details the steps to follow to perform the calibrations necessary to determine this information.

A.1 Determination of the laser beam projection axis

A frame, $\mathcal{F}_0$, is attached to the robot base with $\mathcal{F}_0\left(\boldsymbol{X}_0, (\boldsymbol{i}_0, \boldsymbol{j}_0, \boldsymbol{k}_0)\right)$ with $\boldsymbol{X}_0 = (x_0, y_0, z_0)$ and $\boldsymbol{k}_0$ the axis normal to the detector plane. A frame, $\mathcal{F}_{ee}\left(\boldsymbol{X}_{ee}, (\boldsymbol{i}_{ee}, \boldsymbol{j}_{ee}, \boldsymbol{k}_{ee})\right)$ is attached to the robot end-effector. The aim of this section is to determine $\boldsymbol{k}_s$ with relation to $\boldsymbol{k}_0$. To do so, the rotation matrix, $R(\theta, \boldsymbol{u}) \in SO(3)$, describing the orientation of $\boldsymbol{k}_s$ with relation to $\boldsymbol{k}_{ee}$ is computed. This rotation matrix can be described as a rotation by angle θ about an axis in the direction $\boldsymbol{u}$. A frame, $\mathcal{F}_d$, is attached to the robot base. This position is undetermined at the beginning of the calibration. Only relative positions of the laser spot with relation to the detector sensor are used in this first step.

As an initial guess, the orientation of $\boldsymbol{k}_s$ is supposed to be the same as $\boldsymbol{k}_{ee}$. The robot end-effector is positioned such that $\boldsymbol{k}_{ee} \times \boldsymbol{k}_0 = \boldsymbol{0}$. The laser source is positioned

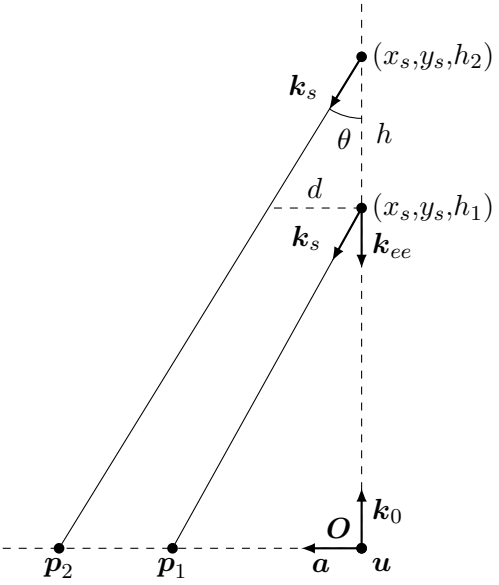


FIGURE A.1: Schematic presentation of the laser source positions and laser spot positions relatively to the robot base in the $(\boldsymbol{O}, (\boldsymbol{a}, \boldsymbol{k}_0))$ plane. $\boldsymbol{k}_s$ is the laser source real projection axis that needs to be calibrated

at a random point in Cartesian space: (x_s, y_s, h_1) above the laser detector. The position of the laser spot on the detector plane, $\boldsymbol{p}_1(x_{p_1}, y_{p_1})$, is recorded. Without modifying the laser frame orientation, the laser source is moved along $\boldsymbol{k}_0$ to a position (x_s, y_s, h_2). The distance between h_1 and h_2 is called h. The new position of the laser spot, $\boldsymbol{p}_2(x_{p_2}, p_{l_2})$ is recorded. With this information, it is possible to compute the axis of rotation $\boldsymbol{u}$, and the rotation angle θ necessary to transform $\boldsymbol{k}_{ee}$ to $\boldsymbol{k}_s$.

If the robot is not moving along $\boldsymbol{k}_s$, the laser spot moves along a line passing through the points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ along the robot base. The distance between these two points is called d and the direction vector of the line passing through these points is called $\boldsymbol{a}$. Figure A.1 depicts the position of the end-effector and the laser beam during this experiment in the $(\boldsymbol{O}, (\boldsymbol{a}, \boldsymbol{k}_0))$ plane with $\boldsymbol{O}\,(x_{ee}, y_{ee}, z_d)$. To describe the relative orientation of $\boldsymbol{k}_s$ with relation to $\boldsymbol{k}_{ee}$, one must perform a rotation, $R(\theta, \boldsymbol{u})$ along the axis

$$\boldsymbol{u} = \frac{\boldsymbol{a} \times \boldsymbol{k}_0}{\|\boldsymbol{a} \times \boldsymbol{k}_0\|}, \tag{A.1}$$

with a rotation angle

$$\theta = atan\frac{d}{h}. \tag{A.2}$$

The new laser direction $\boldsymbol{k}_s$ is

$$\boldsymbol{k}_s = R(\theta, \boldsymbol{u})\boldsymbol{k}_s. \tag{A.3}$$

Due to error in the measures, the determination of $\boldsymbol{u}$ and θ may not give a correct rotation matrix. To verify if $\boldsymbol{k}_s$ is the correct laser source direction, one can perform a motion along $\boldsymbol{k}_s$. The laser spot should not be moving. If this is not the case, $\boldsymbol{k}_s$ is used as the new laser direction and the first step is repeated. When the distance between $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ is sufficiently small, this step of the calibration procedure is done.

Algorithm 5 sums up the operations to perform to calibrate the projection axis.

Algorithm 5: Algorithm to determine the laser projection axis from the robot position and the laser spot relative position.

$(\boldsymbol{i}_s, \boldsymbol{j}_s, \boldsymbol{k}_s) = (\boldsymbol{i}_{ee}, \boldsymbol{j}_{ee}, \boldsymbol{k}_{ee})$
Position the robot laser frame parallel to its base frame
while $\|\boldsymbol{p}_1 - \boldsymbol{p}_2\| > \epsilon$ **do**
  Gather two points on the detector surface, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, at different laser source height h_1 and h_2, with the same laser orientation
  $\boldsymbol{a} = \frac{\boldsymbol{p}_1 - \boldsymbol{p}_2}{\|\boldsymbol{p}_1 - \boldsymbol{p}_2\|}$
  $\theta = atan\frac{\|\boldsymbol{p}_1 - \boldsymbol{p}_2\|}{h}$
  $\boldsymbol{u} = \frac{\boldsymbol{a} \times \boldsymbol{k}_0}{\|\boldsymbol{a} \times \boldsymbol{k}_0\|}$,
  Compute the rotation matrix $R(\boldsymbol{u}, \theta)$
  $\boldsymbol{k}_s = R(\boldsymbol{u}, \theta)\boldsymbol{k}_s$
end
return $\boldsymbol{k}_s$

This operation must be carefully performed as an error in the laser orientation will result in a large projection error. For example, a laser source located at 0.6 m from the detector plane with a 1° orientation error will project a laser spot at 10 mm from the desired position.

A.2 Determination of the laser source position offset

Once the laser projection axis has been determined, the position of the laser source, $\boldsymbol{X}_s(x_s, y_s, z_s)$, relatively to the end-effector, $\boldsymbol{X}_{ee}(x_{ee}, y_{ee}, z_{ee})$ must be calibrated. Since it is supposed that the laser beam projects light along a line in the $\boldsymbol{k}_s$ direction, the position z_s is not relevant here as it does not influence where the laser beam is projected. The laser frame is positioned so that $\boldsymbol{k}_s \times \boldsymbol{k}_0 = \boldsymbol{0}$. The robot then realises a rotation about the $\boldsymbol{k}_s$-axis. If the first step in Section A.1 has been correctly performed and there

is an offset in the laser source position, the projection of the laser beam should form a circle on the quadrant photodiode plane. The data gathered will be subject to noise, regardless of the acquisition method. A circle must be fitted with the obtained data. Given a set of measured (x,y) pairs, a method minimizing the sums of squared radial deviation is used[1]. This method gives an estimated position of the circle centre, noted (x_c, y_c), on the photodiode plane.

Without moving the robot from its current position $\boldsymbol{X}_s$, the projection of the laser spot should be on the circle. Given that the detector position relatively to the robot base is still unknown, let us note (x_p, y_p), the position of the laser spot relatively to the circle centre. The corrected laser source position is

$$\boldsymbol{X}_s = \begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix} + \begin{pmatrix} x_c - x_p \\ y_c - y_p \\ 0 \end{pmatrix} \tag{A.4}$$

Once again, this step should be repeated several times to obtain a good estimation of the laser frame position. This step is completed once the laser spot is located on the centre of the circle and the laser spot does not move when performing a rotation about the laser projection axis.

A.3 Determination of the quadrant-photodiode position

The last step of this calibration procedure requires to determine the quadrant-photodiode position relatively to the robot base. Indeed, the servoing of the pointing task requires to define a target position. This position should be ideally the centre of the quadrant-photodiode detection surface, $\boldsymbol{X}_d\,(x_d, y_d, z_d)$. The laser being correctly calibrated, it can now be used to determine the detector position.

The next step supposes that the detector surface is parallel to the robot base. The determination of the position of x_t and y_t requires to position the laser frame so that $\boldsymbol{k}_s \times \boldsymbol{k}_0 = \boldsymbol{0}$. Pointing the laser towards the centre of the detector directly gives $(x_d, y_d) = (x_s, y_s)$.

The next step consists in pointing the laser toward the detector and to record the position and orientation of the laser frame and the position of the laser spot relatively to the robot base. The parametric equation of a line passing through (x_s, y_s, z_s) along the vector $\boldsymbol{k}_s = (a, b, c)$ is

[1] https://fr.mathworks.com/matlabcentral/fileexchange/5557-circle-fit

$$\begin{cases} x = x_s + ta & \text{(A.5a)} \\ y = y_s + tb & \text{(A.5b)} \\ z = z_s + tc & \text{(A.5c)} \end{cases}$$

with t a scalar parameter.

The quadrant photodiode records the position of the laser spot x_p and y_p relatively to the centre of its detection surface. These positions are used in Equations (A.5a) and (A.5b) and yields $t_x = \frac{x_p - x_s}{a}$ and $t_y = \frac{y_p - y_s}{b}$. Ideally t_x should be equal to t_y but due to uncertainty in the measure it might not be the case. To minimize the error, it is best to choose $\bar{t} = \frac{t_x + t_y}{2}$. Using $\bar{t}$ in Equation (A.5c) yields

$$z_d = z_s + \bar{t}c. \tag{A.6}$$

For better results it is recommended to perform this step for different positions of the laser source to get a mean value of z_d.

Once this step is realised, the laser frame is correctly determined and the position of the laser detector relatively to the robot base is acquired.