ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE - DIN

CORSO DI LAUREA IN INGEGNERIA MECCANICA LM

TESI DI LAUREA

in

MECCANICA DEI ROBOT M

Robust Grasp with Compliant Multi-Fingered Hand

CANDIDATO:
Marco Speziali

RELATORE:
Prof. Ing. Vincenzo Parenti Castelli

CORRELATORI:
Prof. Ing. Oussama Khatib
Ing. Shameek Ganguly
Ing. Mikael Jorda

Anno accademico 2019/2020 - Sessione III

Your work is going to fill a large part of your life,
and the only way to be truly satisfied is to do what you believe is great work.
And the only way to do great work is to love what you do.

Steve Jobs

Abstract

As robots find more and more applications in unstructured environments, the need for grippers able to grasp and manipulate a large variety of objects has brought consistent attention to the use of multi-finger hands. The hardware development and the control of these devices have become one of the most active research subjects in the field of grasping and dexterous manipulation. Despite a large number of publications on grasp planning, grasping frameworks that strongly depend on information collected by touching the object are getting attention only in recent years. The objective of this thesis focuses on the development of a controller for a robotic system composed of a 7-dof collaborative arm + a 16-dof torque-controlled multi-fingered hand to successfully and robustly grasp various objects. The robustness of the grasp is increased through active interaction between the object and the arm/hand robotic system. Algorithms that rely on the kinematic model of the arm/hand system and its compliance characteristics are proposed and tested on real grasping applications. The obtained results underline the importance of taking advantage of information from hand-object contacts, which is necessary to achieve human-like abilities in grasping tasks.

Contents

Chapter 1

Introduction

"Robots are not people (Roboti nejsou lidé). They are mechanically more perfect than we are, they have an astounding intellectual capacity, but they have no soul. The creation of an engineer is technically more refined than the product of nature." Such is the definition of robots given in the prologue of the Czech drama "R.U.R." (Rossum's Universal Robots) by Karel Čapek. In this occasion, the Czech playwright coined the word *robot* deriving it from the term *robota* that means executive labour in Slav languages. The robot presented in the play is, however, far different from the well-known image of robots as complex mechanical devices. The latter was first presented a few decades later by Isaac Asimov. The Russian fiction writer conceived the concept of robots, in the 1940s, as of automatons with human appearance but devoid of feelings. Furthermore, he introduced the term robotics as the science devoted to the study of robots and he based such discipline on the three famous fundamental laws:

- A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- A robot must obey the orders given by human beings, except when such orders would conflict with the first law.
- A robot must protect its own existence, as long as such protection does not conflict with the first or second law.

Although the idea of robots as highly autonomous systems with human-like abilities is still a product of science fiction, the science of robotics is already a mature technology when it comes to industrial applications. It is, in fact, the norm to see robot manipulators being used in a huge number of industrial tasks such as: welding, assembly, packaging and labeling, assembly of printed circuit boards, product inspection, etc.
Despite the high capabilities of industrial robots, the above listed activities are generally performed in a *structured environment* where physical characteristics are clearly defined and known a priori. In this field, the concept of a robot could be then captured by the definition given by the Robot Institute of America: *a robot is a reprogrammable multifunctional manipulator designed to move materials, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks.*

Roboticists worldwide are currently aiming at a much more aspiring target: developing robots able to operate in unstructured environment. The goal is to introduce robots to hospitals, offices, construction sites and homes with significant societal, scientific, and economic impact. In these environments, robots can't rely anymore on simple *programmed motions* but there is the need to develop competent and practical systems capable of interacting with the environment and with humans.
A first step to achieve this goal is the development of robots able to share their workspace with humans and physically interact with them. We are moving closer to this target developing torque-controlled robots about which is given a brief overview in the next paragraphs.
Another fundamental point to be covered is the ability to grasp and manipulate objects of various dimensions and shapes. A domestic robot would be of little utility if not capable of interacting with the high number of different objects that can be found in a house. To achieve such abilities researchers are working on the development of highly dexterous robotic hands as explained in the following sections.

1.1 Collaborative robots

Traditional industrial robots are programmed to work at a distance from workers, they are heavy, fast, expensive and they request high expertise for being set up. These robots are installed to perform a single and repetitive task where the environment around them is completely known. Despite being vastly utilized in modern industries, traditional robots are hazardous to humans and require fencing or other barriers in order to isolate them from human contact. Despite the fact that these

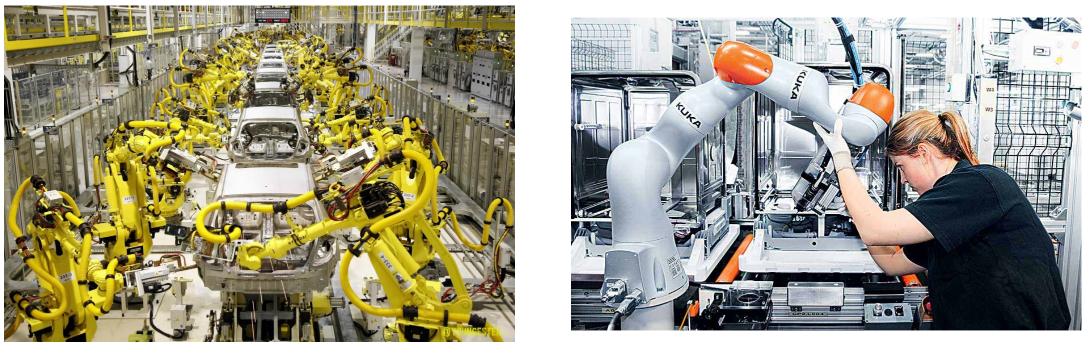

(a) Traditional industrial robots (b) Collaborative robots

Figure 1.1: The challenge of unstructured environments

robots play a key role in a large number of industries (Figure 1.1), we are trying to extend the presence of robots outside of their protective fences and use them as versatile coworkers helping humans in complex or physically demanding work. These robots have to be able to operate in unstructured, partially unknown and dynamic environments [1]. They will share their workspace with humans and avoid undesired collisions while handling intentional and accidental contact in a safe and robust way ([2],[3],[4]). Equipping robots with these skills is the global goal of physical human–robot interaction (pHRI) research [5].

With this goal in mind, in the last decade, several companies have developed and commercialized a new type of robotic manipulators: collaborative robots. These kinds of robots are lighter, characterized by rounded edges and, especially, are equipped with built-in force-torque sensors that enable them to interact with humans safely. This is not the only valuable characteristic of collaborative robots: they allow a fast set-up, easy programming, flexible deployment and they can provide a quicker return on investment (ROI) than their heavier, more dangerous industrial counterparts.

The idea of introducing robots able to collaborate with workers in the industries was introduced by J. Edward Colgate and Michael Peshkin [6], professors at Northwestern University in 1996. They called these kinds of robots *cobots* describing them as *"an apparatus and method for direct physical interaction between a person and a general-purpose manipulator controlled by a computer"*. These "cobots" assured human safety by having no internal source of motive power. The collaborative robots that are conquering the industries in the last years are more complex and advanced than the robots proposed by the Northwestern University professors. Their main feature is their reliance on torque-control schemes making them strongly different from the old industrial robots which are equipped with embedded position controllers.

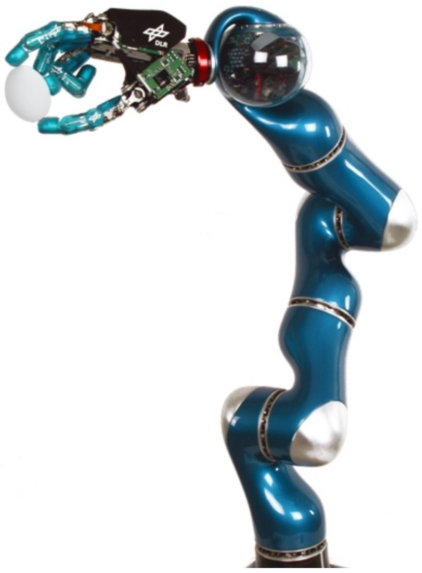


Figure 1.2: Collabrative robot DLR LBR III

This position control, realized at the joint level, doesn't allow to compensate for the nonlinear dynamic coupling of the system. Hence, the dynamic coupling effects have to be treated as a disturbance. On the other hand, a torque-control robot can handle this problem and allows better performance in position tracking, especially in compliant motion.

Until the early 21st century very few robots were developed with torque-control capabilities, although efforts have been made to extend torque-control algorithms to position control robot [7]. One of the first robots developed with torque-control

capability was released by KUKA in 2004. This robot, *KUKA LBR III*, was the outcome of a bilateral research collaboration with the Institute of Robotics and Mechatronics at the German Aerospace Center (DLR), Wessling. The latter started working on the development of light-weight robots in the 1993 ROTEX space shuttle mission. To enable the astronauts to train for the mission they needed a comparable robot on Earth. In this occasion the need for a small lightweight robot was born leading to the development of three generations of lightweight robots: LWR I, LWR II, and LWR III [8]. The first version, weighted only 18 kg with a load to weight ratio close to 1:2. Each actuator was equipped with a double planetary gearing with a 1:600 ratio and an inductive torque-sensing. In the second version, a harmonic drive was used instead and the torque sensing was substituted by a strain-gauge based torque measurement system, embedded into a full state measurement and feedback system (motor position, link position, joint torque). The third generation (LWR III, Figure 1.2) was equipped with motors and encoders developed by DLR, lightweight construction principles and advanced control concepts were applied.
After the technological transfer, KUKA further refined the technology [9], releasing the KUKA LBR 4 in 2008 and the serial product for industrial use LBR Iiwa in 2013 (Figure 1.3).

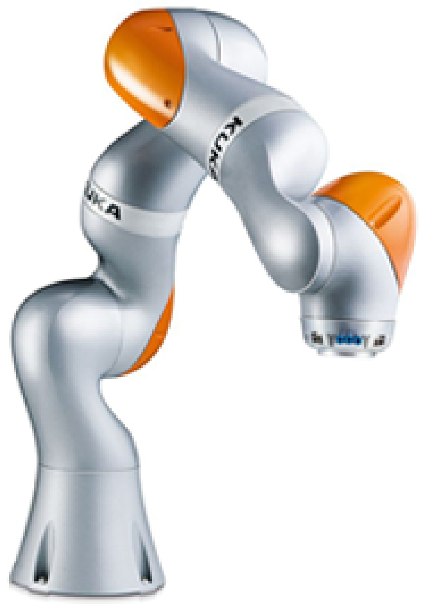


Figure 1.3: KUKA Iiwa

In the meanwhile, other robotics companies had started working on light-weight collaborative robots. The UR5 was released by Universal Robot in 2008. Several other versions were released in the following years and they are shown in Figure 1.4.

Also FANUC entered the collaborative robot market with the FANUC CR-35iA in 2015. ABB proposed in the same year YuMi, the first collaborative dual arm robot. Lately numerous other companies developed their own collaborative robots. Among them, it is worth citing FRANKA EMIKA GmbH that introduced a new

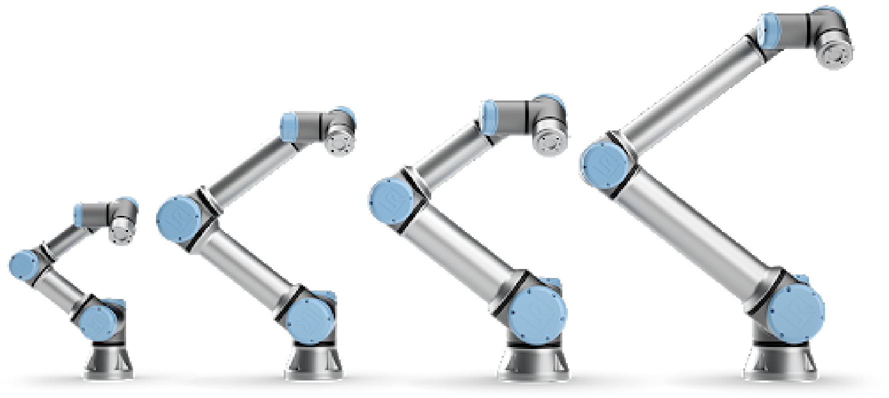


Figure 1.4: Universal robots

fully softrobotics-controlled lightweight robot [10] to the market, offering a low-cost collaborative arm with torque sensors in all 7 axes (Figure 1.5).

Figure 1.5: Panda robot, Franka Emika GmbH.

The element usually mounted on the last joint of the robots is called *end-effector* and it constitutes the interface between the robot and the environment. The end-effector is often composed of a simple two- or three-jaw gripper, tongs, remote compliance devices or other specific tools. In order to increase the flexibility of the robot, roboticists are developing multi-finger hands with the goal in mind of replicating the dexterity and flexibility of manipulation shown by human-beings.

1.2 Grasping and dexterous Manipulation with Multi-finger hands

The ability of humans to explore and manipulate objects played a big role in their thriving and their affirmation as the main species on the planet. The importance of hands in humans is further underlined by the amount of brain dedicated to our hands. This aspect is well shown in *The cortical homunculus* (Figure 1.6) , a distorted representation of the human body, where the size of his parts are proportional to the size of brain areas devoted to them.

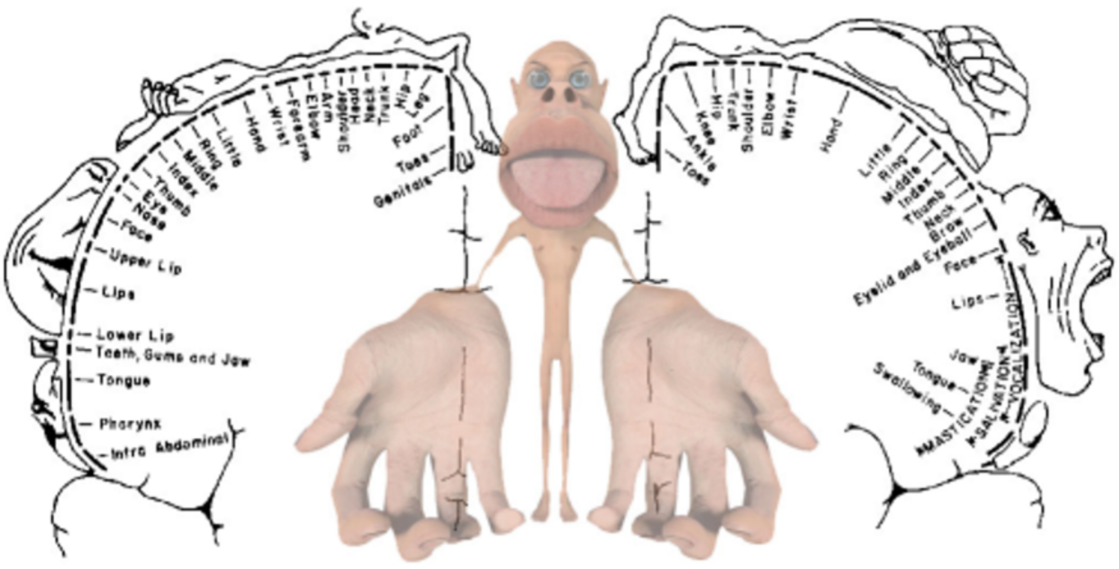


Figure 1.6: The Cerebral homunculus.

One of the first thing that stands out is the disproportion between the hands and the rest of the body. Interestingly, the fraction of the motor cortex dedicated to the control of hands is often used as a measure of the intelligence of a member of the mammalian family. It is no surprise that $30 - 40\%$ of the motor cortex in humans is dedicated towards the hands in contrast to $20 - 30\%$ for most other primates and 10% for dogs and cats. Transferring all those humans skills related to the use of hands is a challenging and inspiring goal for robotics.

Grasping and manipulate various objects is essential both in industrial and non-industrial environments. For example: assembly, pick and place movements, lifting dishes, glasses, toys are essential grasping scenarios.
The bibliography about the grasping problem is very rich and still in constant expansion. In the latest years, there has been a great amount of work around multi-finger hands both for their hardware design and for their direct applications in solving grasping problems. These hands were developed to give robots the ability to grasp objects of varying geometry and physical properties. Taking advantage of their potential in reconfiguring themselves for performing a variety of different grasps, it is possible to reduce the need for changing end-effector and to close the distance in replicating human abilities.

The first robotic hand designed for dexterous manipulation was the Salisbury hand (fig. 1.7). It has three three-jointed fingers, enough to control all six degrees of freedom of an object and the grip pressure.

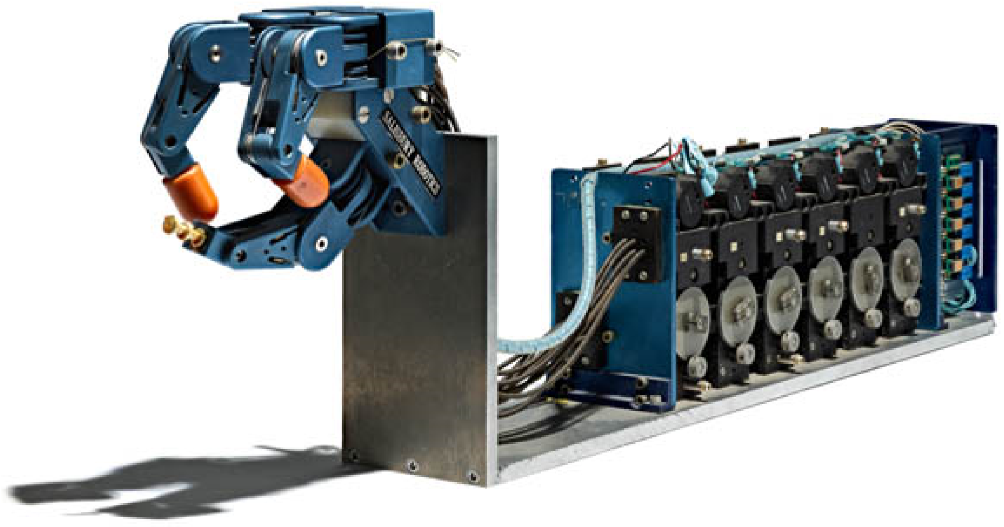


Figure 1.7: Salisbury hand

In the following years, the number of multi-finger hands developed by researchers and companies increased substantially finding applications in numerous fields. As proposed in [11] these areas can be split into three main categories:

- prosthetics and rehabilitation (assistive robotics and prosthetics).
- industrial (supervised manipulation, autonomous manipulation, and logistics).
- human–robot interaction (teleoperation, teleinteraction, social robotics, entertainment, and service robotics).

Although it may seem tempting to design a fully actuated anthropomorphic hands, given the large capabilities of human hands, this hasn't yet lead to an increase in performances when compared to non-anthropomorphic end-effector or partially actuated anthropomorphic hands. The results stemmed from numerous international competitions such as the Amazon Picking Challenge, the Robotic Grasping and Manipulation Competition and the DARPA Robotics Challenge where it was shown that the latter performed better in terms of dexterity underling how approaches aiming at simplified design can bring to practical benefits.
This is probably due to the complexity and problems that come along with the design of multi-finger hands underling how the design and control of artificial hands remain one of the hardest challenges in robotics ([12], [13]). Regardless, the motives for keeping researching in this field are consistent. The improvement in hardware and sensing has still great potential, furthermore, as stated in [12] we can identify the following reasons to support anthropomorphic designs:

- The end-effector can operate on a man-oriented environment, where tasks may be executed by the robot or by man as well, acting on items, objects or tools that have been sized and shaped according to human manipulation requirements.
- The end-effector can be teleoperated by man, with the aid of special-purpose interface devices (e.g. a data-glove), directly reproducing the operator's hand behavior.
- It may be specifically required that the robot has a human-like aspect and behavior, like for humanoid robots for purposes of entertainment, assistance, and so on.

As stated in [14] the three main functions of the human hands are to explore, to restrain objects (*fixturing*), and to manipulate objects (in-hand manipulation, *dexterous manipulation*). The first function, to explore, is related to the field of haptics. So far, the work in robot grasping has dealt meanly with the latter two abilities although the exploratory part is an essential element for successfully grasp and manipulate unknown objects especially when vision is impaired or completely absent.
The way the multi-finger hand will grasp the object is strictly correlated to the purpose of the grasping process itself. It is possible to identify two types of grasps [15]:

- Power grasps: The object may be held in a clamp formed by the partly flexed fingers and the palm, counter pressure being applied by the thumb lying more or less in the plane of the palm.
- Precision grasps: the object may be pinched between the flexor aspects of the fingers and the opposing thumb.

These two categories define the position of the fingers while holding an object. As the names suggest, power grasps are in favor of a tight and stable hold whereas a precision grasp could allow in-hand manipulation of the objects. In the following years, different types of static hand postures were analyzed and classified as a function of object and task properties. As far as robotics is concerned, one of the earliest taxonomy of grasping postures was developed by Cutkosky [16] and following used to label human grasps for a variety of tasks [17]. Subsequently, a large number of different taxonomies were defined. 30 of these were analyzed and combined into a single one in [18].

The grasping problem is not only limited to how the objects are held. In fact, we can describe the grasping process with the following steps:

- Grasp planning
- Approaching object
- Tactile exploration, Grasp and lift
- Manipulate

Other steps could be added to the previous list, someone could argue that also the releasing part of the object is an important point related to the grasping problem. During this thesis, only those four points are explored.

The first step relies on long-range vision sensors, non-contact sensors such as cameras and laser scanners, to detect the objects and select how to grasp it. A large amount of research work has been done on this first point. Once the object is localized a grasp position is identified in order to satisfy a set of criteria relevant for the grasping task. This problem is known as *grasp analysis*. As reviewed by [19] two different kind of approaches have emerged in the literature, *analytical* and *empirical* methodologies:

- *Analytical* methodologies are formulated as a constraint optimization problem over grasp quality measures. It relies on mathematical models of the interaction between the object and the hand.
- *Empirical* or *data-driven* approaches rely on sampling grasping candidates for an object and ranking them according to a specific metric. This process is usually based on some existing grasp experience that can be a heuristic or is generated in simulation or on a real robot.

The work on analytical methodologies is vast. A good review is given by Bicchi and Kumar [14] whereas a review on different quality metrics has been written by A. Roa and R. Suárez [20]. Although this approach works well in simulations it is not clear if the simulated environment resembles the real world well enough to transfer methods easily. Inaccuracies in the robotic systems and in the sensor outputs are always present, contact locations identified by the grasp planner could be not reachable or small errors could occur. Several attempts to design analytical grasp planning algorithms that account for a certain degree of inaccuracy have been made [21], [22]. Despite these efforts, a considerable number of publications showed how classic metrics are not good predictors for grasp success in the real world ([23], [24]).

Data-driven methodologies, on the other hand, started being consistently used only in the 21st century with the availability of GraspIt! [25]. These methods rely on machine learning approaches with the intent of letting the robot learn how to grasp by experience. Although, collecting examples is laborious, transferring a learned model to the real robot is trivial. In [26] it is given a review of data-driven approaches, dividing them in three categories: *known objects*, *familiar objects* and *unknown objects*.

The second step, approaching the object, is relatively straightforward. The movement of the end-effector towards the object to be grasped must take into consideration possible obstacles, in case of multi-finger hands, the position of the palm has to be optimized in case only the points of contact are given.

Most of the works done so far on grasping covered the first two steps. The grasping is then performed without taking advantage of feedback from tactile information. It has been shown that humans strongly rely on tactile information in order to grasp an object [27]. In particular, it has been explained how dexterous manipulation tasks can be decomposed into a sequence of action phases. The task is therefore composed of sub-goals, usually determined by specific mechanical events, where the information from tactile sensing plays a key role. For example, the contact between the digits and an object marks the end of the reach phase. Always in [27] the steps that compose a grasping task are defined as follows: reach, load, lift, hold, replace, unload, and release. This has expired Romano et al. [28] to design these basic controllers and running them in succession. The transition from one controller to another was triggered by a specific signal from the tactile sensors installed into a robotic hand. M. Kazemi et al. [29] proposed a similar procedure taking into consideration the contact between fingertips and the support surface and showing their key role in establishing a proper relative position of the fingers and the object throughout the grasp.

However, the importance of tactile information is not a new concept. Using tactile sensory cues for robotic actions was already proposed nearly three decades ago by Howe et al. [30]. During the last years, a big number of tactile sensors have been developed and the literature on the topic is abundant as reviewed in [31] and showed in [32]. Nevertheless, as stated previously, works that exploit information from touching the object are still relatively new. The technology around tactile sensors is still in a development process and most of them are not yet available outside the research labs in which they are developed. Only in the last years, with the emergence of robotic grippers and hands equipped with tactile and force sensors, more and more works proposed approaches where tactile sensor feedback was included ([33], [34], [35], [36], [37]).

Once the object has been grasped and lifted, there could be the need for changing his relative position with respect to the palm of the hand. This could be useful for assembly applications or for all those tasks where reliance on the arm movement to change the posture of the object wouldn't be convenient. Roboticists are trying to confer such ability, called *dexterous in-hand manipulation*, to multi-finger hands. Although several works have been done for planar tasks, the execution of dexterous manipulation tasks in three dimensions continues to be an open problem in robotics.

1.3 Outline of the thesis

As stated in the previous paragraphs the problem of *grasping* is wide and complex. To successfully grasp and manipulate objects is necessary to use both an arm (to approach and give macro-movements to the object itself) and an appropriate end-effector such as a multi-finger hand (to explore, restrain and manipulate the object). The objective of this thesis revolves around the development of a controller for an arm/hand robotic system in order to successfully and robustly grasp various objects. Nearly all the steps constituting a grasping task, listed in the previous section, are covered. Using a fully actuated multi-finger hand it is showed how it is possible to increase the robustness of the grasp through active interaction between the object and the arm/hand robotic system. Tacking advantage of the compliance of the multi-finger hand and the kinematic model of the arm/hand system it is, in fact, possible to give a sense of touch to the robotic system increasing his performances. The target of this thesis is twofold:

- Showing the importance of active interaction with the object in order to increase grasp success and robustness.
- Underling the benefits given by utilizing a torque-control multi-finger robotic hand during hand-object interactions.

The thesis is structured as follows:

Chapter 1 has already given a brief introduction about collaborative robots and multi-finger hands with the purpose of introducing broadly the hardware utilized in this thesis (well described in Chapter 3) and the *grasping* topic.

Chapter 2 is dedicated to the mathematical framework utilized to develop the arm/hand controller. Basic concepts about robotic control with particular attention to Operational-space control are presented. A consistent part of this chapter is dedicated to the mathematical background around the grasping problem and the force optimization algorithms adopted throughout this thesis to solve the *force distribution* problem.

Chapter 3 is dedicated to describing the arm/hand system utilized during the development of the thesis. The software tools utilized to develop the controller are shown together with the hardware components composed by a collaborative arm, a multi-finger fully actuated hand and an eye-in-hand RGBD camera.

Chapter 4 presents innovative methods developed during the thesis. They rely on the compliance capabilities of the hand and their effectiveness is demonstrated.

Chapter 5 constitutes the core of this thesis where the algorithms developed in Chapter 4 are applied to real grasping applications.

Chapter 6 includes conclusions. The results shown in Chapter 5 are further analyzed, points of strength and weakness of the new algorithms proposed are presented. Possible developments in this work are listed.

Chapter 2

Mathematical background

The content of this chapter comprises a mathematical description of the methodologies used throughout the thesis. A first section is dedicated to robotic motion control where the difference between joint-space and operational space control is explained, the latter is presented and a few descriptions of the algorithms utilized for robot control in redundant manipulators are given. The second part of the chapter is dedicated to the description of mathematical models and methods utilized in the field of *grasping*, different contact models are presented along with the concept of *grasp matrix*, *hand jacobian* and other important milestones from the grasping robotic discipline. The last section of this chapter is dedicated to the *force distribution* problem where the theory utilized in this thesis for selecting the forces to be applied with the fingers in order to guarantee a robust grasp is explained.

2.1 Robot control

A large portion of the work done for this thesis has dealt with the development of the controller for a robotic system composed of a 7-*dof* collaborative robot and a 16-*dof* torque-controlled robotic hand. The robotic system is described in Chapter 3 where a detailed explanation of the controller architecture is presented. The

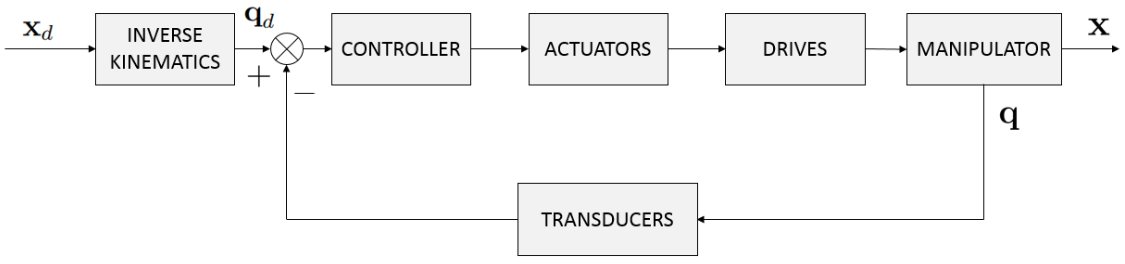


Figure 2.1: General scheme of joint space control

problem of controlling a manipulator can be formulated as that to determine the inputs to be sent to each joint motor in order to guarantee the execution of the commanded task while satisfying given transient and steady-state requirements. A task may be related only to motion in free space or to the execution of contact forces and movements in a constrained environment.

The task (end-effector motion and forces) is usually carried out and specified in the cartesian space (operational space), whereas control actions (joint actuator generalized forces) are performed in the joint space. This fact naturally leads to considering two kinds of general control schemes, namely, a *joint space control* scheme (fig. 2.1) and an *operational space control* scheme (Figure 2.2). In both schemes, the control structure has closed loops to exploit the good features provided by feedback, in the first case the outer loop is around the joint angular positions whereas in the operational space control the quantities confronted lie in the task space. The choice

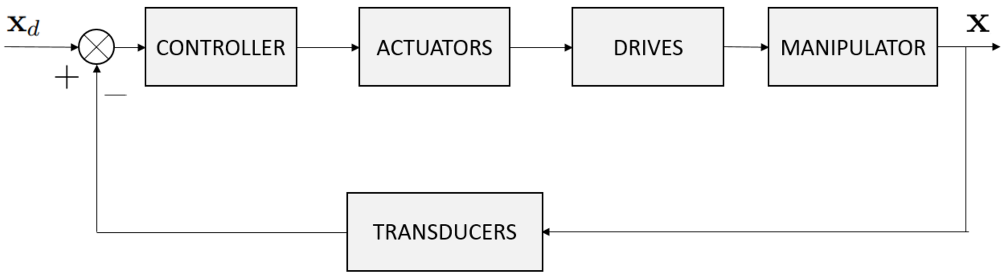


Figure 2.2: General scheme of operational space control

for the right controller is influenced by a lot of factors. The power of the calculator utilized to run the controller is a first aspect that could hinder the possibility of implementing real-time computationally expensive controllers. The mechanical hardware plays a big role as well. If the electric motor in each joint is equipped with a reduction gears of high ratios, the presence of gears tends to linearize system dynamics, decoupling the joints in view of the reduction of nonlinearity effects. However, joint friction, elasticity and backlash are introduced reducing performances of the controller. On the other hand, utilizing direct drives, the previously listed drawbacks could be avoided, but the weight of nonlinearities and couplings between the joints becomes relevant. Consequently, different control strategies have to be thought of to obtain high performance. The nature of the task plays a big role as well. In case the end-effector is in contact with the environment and precise forces have to be exchanged the choice of the controller falls on an active force control for which the operational space control is requested.

2.1.1 Joint space control

The joint space control problem could be divided into two steps. First, the desired motion $\mathbf{x}_d$ defined in the operational space is transformed into the corresponding motion $\mathbf{q}_d$ in the joint space through inverse kinematic [38]. Then, a joint space control scheme is implemented allowing the actual motion $\mathbf{q}$ to follow the desired angular position $\mathbf{q}_d$. This second step can be achieved in a multitude of ways. The controller could be *decentralized* considering nonlinear effects and coupling between joints as disturbances. To achieve better performances a *centralized* scheme could be implemented such as the inverse dynamic control.
In this scheme, the manipulator dynamic model is used to deal with the inertial coupling and to compensate for centrifugal, Coriolis, and gravity forces. This technique is based on the theory of nonlinear dynamic decoupling [39], or the so called

computed torque method. Given the equation of motion described in joint space:

$$A(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) = \Gamma \tag{2.1}$$

The inverse dynamic control is achieved by choosing the following control structure:

$$\Gamma = \hat{A}(\mathbf{q})\Gamma^{*} + \hat{\mathbf{b}}(\mathbf{q}, \dot{\mathbf{q}}) + \hat{\mathbf{g}}(\mathbf{q}) \tag{2.2}$$

Where $\hat{A}(\mathbf{q})$, $\hat{\mathbf{b}}(\mathbf{q}, \dot{\mathbf{q}})$ and $\hat{\mathbf{g}}(\mathbf{q})$ represent the estimates of the kinetic energy matrix $A(\mathbf{q})$, the vector of centrifugal and Coriolis forces $\mathbf{b}(\mathbf{q}, \dot{\mathbf{q}})$ and the vector of gravity forces $\mathbf{g}(\mathbf{q})$.
A scheme of the controller is shown in Figure 2.3 where the inverse kinematic step is omitted:

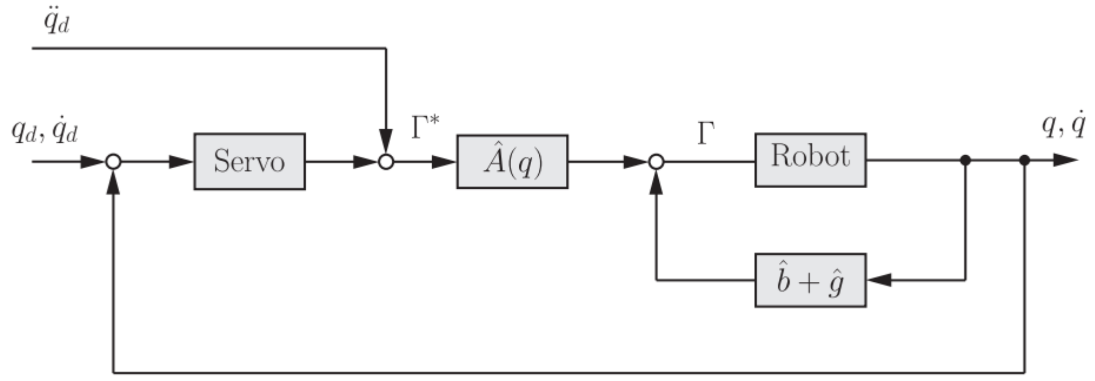


Figure 2.3: Joint space control structure

At this point the vector $\mathbf{\Gamma}^*$ becomes the input of the decoupled system where a simple PID control structures could be selected or, to deal with model uncertainties, robust or adaptive controllers could be implemented. One of the main downsides of the joint space framework is that operational space variables $\mathbf{x}$ are controlled in an open-loop fashion although they are the real goal of the task. What is more, when the end-effector is in contact with the environment a joint space control scheme is not suitable and an operational space control approach has to be chosen. Humans strongly rely on contacts and compliant motion to perform nearly every kind of task. The adoption of a operational space control is therefore obvious thinking in term of unsructured environment applications.

2.1.2 Operational space control

Task specification for motion and contact forces, force sensing feedback and dynamics are related to the end-effector's motion. A description of the dynamic behavior of the end effector, essential for controlling end-effector's motion and applied forces, is not well achieved through joint space dynamic models. The adoption of an operational space approach is therefore requested. The objective of such a scheme is to control motion and contact forces using control forces that act directly at the task space level. To achieve this kind of scheme it is necessary to develop a model able to describe the dynamic behavior at the point on the effector where the task is specified (*operational point*).

The use of the forces generated at the end-effector to control motions leads to a natural integration of active force control. In the following paragraphs, the operational space control framework is presented, although the natural implementation of active force control is not treated in this thesis the reader can see [40], [41] or [42] for works on force control. For a more detailed analysis, [43] can be read for *impedance control*, [44] for *compliance control* and [45] for hybrid force/position control.

2.1.2.1 Effector Equations of Motion

When the dynamic response or impact force at some point on the end-effector or manipulated object is of interest, the inertial properties involved are those evaluated at that point, termed the operational point. Attaching a coordinate frame to the end-effector at the operational point and using the relationships between this frame and the reference frame attached to the manipulator base provide a description, $\mathbf{x}$, of the configuration, i.e. position and orientation, of the effector.
The kinetic energy of the holonomic system is a quadratic form of the generalized operational velocities

$$T(\mathbf{x}, \dot{\mathbf{x}}) = \frac{1}{2}\dot{\mathbf{x}}^T \Lambda(\mathbf{x})\dot{\mathbf{x}} \tag{2.3}$$

where $\Lambda(\mathbf{x})$ is the kinetic energy matrix which describes the inertial properties of the end-effector. The equation of motion can be obtained through the Lagrangian formalism:

$$\frac{d}{dt}(\frac{\boldsymbol{\partial} L}{\partial \dot{\mathbf{x}}}) - \frac{\boldsymbol{\partial} L}{\boldsymbol{\partial} \mathbf{x}} = \mathbf{F} \tag{2.4}$$

where $L(\mathbf{x}, \dot{\mathbf{x}})$ is the Lagrangian function defined as:

$$L(\mathbf{x}, \dot{\mathbf{x}}) = T(\mathbf{x}, \dot{\mathbf{x}}) – U(\mathbf{x}) \tag{2.5}$$

$U(\mathbf{x})$ is the gravitational potential energy whereas $\mathbf{F}$ is the operational force vector. Defining $\mathbf{p}(\mathbf{x})$ as:

$$\mathbf{p}(\mathbf{x}) = \nabla U(\mathbf{x}) \tag{2.6}$$

and being $\boldsymbol{\mu}(\mathbf{x}, \dot{\mathbf{x}})$ the vector of centrifugal e Coriolis forces, the equation of motion in operational space is:

$$\Lambda(\mathbf{x})\ddot{\mathbf{x}} + \boldsymbol{\mu}(\mathbf{x}, \dot{\mathbf{x}}) + \mathbf{p}(\mathbf{x}) = \mathbf{F} \tag{2.7}$$

It is interesting identifying the relationship beetween the elements of the joint space equation of motion 2.1 and those that appear in the operational space one 2.7. The connection beetween $\Lambda(\mathbf{x})$ and $A(\mathbf{x})$ is easily identifiable equating the two quadratic forms of the kinetic energy:

$$\frac{1}{2}\dot{\mathbf{q}}^T A(\mathbf{q})\dot{\mathbf{q}} = \frac{1}{2}\dot{\mathbf{x}}^T \Lambda(\mathbf{x})\dot{\mathbf{x}} \tag{2.8}$$

Considering the kinematic model:

$$\dot{\mathbf{x}} = J\dot{\mathbf{q}} \tag{2.9}$$

where J is the jacobian matrix it is possible to obtain the following relationship:

$$A(\mathbf{q}) = J^T(\mathbf{q})\Lambda(\mathbf{x})J(\mathbf{q}) \tag{2.10}$$

As far as the Coriolis forces $\mathbf{b}(\mathbf{q},\dot{\mathbf{q}})$ and $\boldsymbol{\mu}(\mathbf{x},\dot{\mathbf{x}})$ are concerned, the relatonship can be found starting from the expansion of 2.4:

$$\boldsymbol{\mu}(\mathbf{x},\dot{\mathbf{x}}) = \dot{\Lambda}(\mathbf{x})\dot{\mathbf{x}} - \nabla T(\mathbf{x},\dot{\mathbf{x}}) \tag{2.11}$$

And taking advantage of 2.10:

$$\dot{\Lambda}(\mathbf{x})\dot{\mathbf{x}} = J^{-T}(\mathbf{q})\dot{A}(\mathbf{q})\dot{\mathbf{q}} - \Lambda(\mathbf{q})\mathbf{h}(\mathbf{q},\dot{\mathbf{q}}) + \dot{J}^{-T}(\mathbf{q})A(\mathbf{q})\dot{\mathbf{q}} \tag{2.12}$$

$$\nabla T(\mathbf{x},\dot{\mathbf{x}}) = J^{-T}(\mathbf{q})\mathbf{l}(\mathbf{q},\dot{\mathbf{q}}) + \dot{J}^{-T}(\mathbf{q})A(\mathbf{q})\dot{\mathbf{q}} \tag{2.13}$$

where

$$\mathbf{h}(\mathbf{q},\dot{\mathbf{q}}) = \dot{J}(\mathbf{q})\dot{\mathbf{q}} \tag{2.14}$$

$$l_i(\mathbf{q},\dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^T A_{q_i}(\mathbf{q})\dot{\mathbf{q}} \qquad (i = 1, ..., n) \tag{2.15}$$

where we indicated with A_{q_i} the partial derivatived with respect to the i^{th} joint coordinate.
As well known $\mathbf{b}(\mathbf{q},\dot{\mathbf{q}})$ is the vector of centrifugal and Coriolis forces, leading to:

$$\mathbf{b}(\mathbf{q},\dot{\mathbf{q}}) = \dot{A}(\mathbf{q})\dot{\mathbf{q}} - \frac{1}{2}\begin{pmatrix} \dot{\mathbf{q}}^T A_{q_1}(\mathbf{q})\dot{\mathbf{q}} \\ \dot{\mathbf{q}}^T A_{q_2}(\mathbf{q})\dot{\mathbf{q}} \\ . \\ . \\ . \\ \dot{\mathbf{q}}^T A_{q_n}(\mathbf{q})\dot{\mathbf{q}} \end{pmatrix} = \dot{A}(\mathbf{q})\dot{\mathbf{q}} - \mathbf{l}(\mathbf{q},\dot{\mathbf{q}}) \tag{2.16}$$

and

$$\boldsymbol{\mu}(\mathbf{x},\dot{\mathbf{x}}) = J^{-T}(\mathbf{q})\mathbf{b}(\mathbf{q},\dot{\mathbf{q}}) - \Lambda(\mathbf{q})\mathbf{h}(\mathbf{q},\dot{\mathbf{q}}) \tag{2.17}$$

Equating the functions expressing the gravity potential energies and considering the definition of the Jacobian matrix the following relationship holds:

$$\mathbf{p}(\mathbf{x}) = J^{-T}(\mathbf{q})\mathbf{g}(\mathbf{q}) \tag{2.18}$$

All the previously written equations allow to write:

$$J^{-T}[A(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{b}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q})] = \Lambda(\mathbf{x})\ddot{\mathbf{x}} + \boldsymbol{\mu}(\mathbf{x},\dot{\mathbf{x}}) + \mathbf{p}(\mathbf{x}) \tag{2.19}$$

Leading to the dynamic case of the force/torque relationship whose derivation from the virtual work principle assumes static equilibrium:

$$\boldsymbol{\Gamma} = J^T(\mathbf{q})\mathbf{F} \tag{2.20}$$

The equations priviously obtained could be rearranged in a more friendly computational way. $\mathbf{b}(\mathbf{q},\dot{\mathbf{q}})$ can be written in the form:

$$\mathbf{b}(\mathbf{q},\dot{\mathbf{q}}) = B(\mathbf{q})[\dot{\mathbf{q}}\dot{\mathbf{q}}] \tag{2.21}$$

Where $B(\mathbf{q})$ is an n x $n(n+1)/2$ matrix given by:

$$\begin{pmatrix} b_{1,11} & 2b_{1,12} & ... & 2b_{1,1n} & b_{1,22} & 2b_{1,13} & ... & 2b_{1,2n} & ... & b_{1,nn} \\ b_{2,11} & 2b_{2,12} & ... & 2b_{2,1n} & b_{2,22} & 2b_{2,13} & ... & 2b_{2,2n} & ... & b_{2,nn} \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ b_{n,11} & 2b_{n,12} & ... & 2b_{n,1n} & b_{n,22} & 2b_{n,13} & ... & 2b_{n,2n} & ... & b_{n,nn} \end{pmatrix} \tag{2.22}$$

$b_{i,jk}$ are the Christoffel symbols given as a function of the partial derivatived of the joint space kinetic energy matrix $A(\mathbf{q})$ with respect to the generalized coordinated $\mathbf{q}$ by:

$$b_{i,jk} = \frac{1}{2}(\frac{\partial a_{ij}}{\partial q_k} + \frac{\partial a_{ik}}{\partial q_j} - \frac{\partial a_{jk}}{\partial q_i}) \tag{2.23}$$

and

$$[\dot{\mathbf{q}}\dot{\mathbf{q}}] = \begin{bmatrix}\dot{q}_1^2 & \dot{q}_1\dot{q}_2 & \dot{q}_1\dot{q}_3...\dot{q}_1\dot{q}_n & \dot{q}_2^2 & \dot{q}_2\dot{q}_3...\dot{q}_2\dot{q}_n...\dot{q}_n^2\end{bmatrix}^T \tag{2.24}$$

The vector $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}})$ can be similarly written as:

$$\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = H(\mathbf{q})[\dot{\mathbf{q}}\dot{\mathbf{q}}] \tag{2.25}$$

Given these new relationships the vector of Coriolis and centrifugal forces assumes the following expression:

$$\boldsymbol{\mu}(\mathbf{x}, \dot{\mathbf{x}}) = [J^{-T}(\mathbf{q})B(\mathbf{q}) - \Lambda(\mathbf{q})H(\mathbf{q})][\dot{\mathbf{q}}\dot{\mathbf{q}}] \tag{2.26}$$

It is now possible to summarize the relationship between the components of the joint space dynamic model with those of the operational space dynamic ones:

$$\Lambda(\mathbf{x}) = J^{-T}A(\mathbf{q})J^{-1}(\mathbf{q})$$

$$\boldsymbol{\mu}(\mathbf{x}, \dot{\mathbf{x}}) = [J^{-T}(\mathbf{q})B(\mathbf{q}) - \Lambda(\mathbf{q})H(\mathbf{q})][\dot{\mathbf{q}}\dot{\mathbf{q}}]$$

$$\mathbf{p}(\mathbf{x}) = J^{-T}(\mathbf{q})\mathbf{g}(\mathbf{q})$$

2.1.2.2 Dynamic Decoupling

Motion control of the manipulator in operational space lends itself easily to dynamic decoupling. Selecting the following structure for the controller:

$$\mathbf{F} = \hat{\Lambda}(\mathbf{x})\mathbf{F}^* + \hat{\boldsymbol{\mu}}(\mathbf{x}, \dot{\mathbf{x}}) + \hat{\mathbf{p}}(\mathbf{x}) \tag{2.27}$$

where $\hat{\Lambda}(\mathbf{x})$,$\hat{\boldsymbol{\mu}}(\mathbf{x}, \dot{\mathbf{x}})$ and $\hat{\mathbf{p}}(\mathbf{x})$ are the estimates of $\Lambda(\mathbf{x})$,$\boldsymbol{\mu}(\mathbf{x}, \dot{\mathbf{x}})$ and $\mathbf{p}(\mathbf{x})$ that can be found in 2.7. Considering 2.7 and multiply by $\Lambda^{-1}(\mathbf{x})$ we obtain:

$$I_{m0}\ddot{\mathbf{x}} = G(\mathbf{x})\mathbf{F}^* + \boldsymbol{\eta}(\mathbf{x}, \dot{\mathbf{x}}) + \mathbf{d}(t) \tag{2.28}$$

with I_{m_0} identity matrix of dimension m_0 equal to the length of the vector $\mathbf{x}$ and

$$G(\mathbf{x}) = \Lambda^{-1}(\mathbf{x})\hat{\Lambda}(\mathbf{x}) \tag{2.29}$$

$$\boldsymbol{\eta}(\mathbf{x}, \dot{\mathbf{x}}) = \Lambda^{-1}(\mathbf{x})[\tilde{\boldsymbol{\mu}}(\mathbf{x}, \dot{\mathbf{x}}) + \tilde{\boldsymbol{p}}(\mathbf{x})] \tag{2.30}$$

where

$$\tilde{\boldsymbol{\mu}}(\mathbf{x}, \dot{\mathbf{x}}) = \hat{\boldsymbol{\mu}}(\mathbf{x}, \dot{\mathbf{x}}) - \tilde{\boldsymbol{\mu}}(\mathbf{x}, \dot{\mathbf{x}}) \tag{2.31}$$

$$\tilde{\boldsymbol{p}}(\mathbf{x}) = \hat{\boldsymbol{p}}(\mathbf{x}) - \mathbf{p}(\mathbf{x}) \tag{2.32}$$

while $\mathbf{d}(t)$ is a component that includes unmodeled disturbances. It is easily noticeable how with a perfect non linear dynamic decoupling and zero disturbances the end-effector becomes equivalent to a single unit mass moving in the m_0 dimensional space:

$$I_{m_0}\ddot{\mathbf{x}} = \mathbf{F}^* \tag{2.33}$$

2.1.2.3 Trajectory tracking

In case a desired motion of the end-effector is specified, a linear dynamic behavior can be obtained by selecting:

$$\mathbf{F}^* = I_{m_0}\ddot{\mathbf{x}} - k_v(\dot{\mathbf{x}} - \dot{\boldsymbol{x}}_{\boldsymbol{d}}) - k_p(\mathbf{x} - \boldsymbol{x}_{\boldsymbol{d}}) \tag{2.34}$$

where the subscript d stands for the desired values dictated by the trajectory planning. The previous dynamic decoupling and motion control result in the following end-effector closed loop behavior:

$$I_{m_0}\ddot{\boldsymbol{\epsilon}}_x + k_v\dot{\boldsymbol{\epsilon}}_x + k_p\boldsymbol{\epsilon}_{\boldsymbol{x}} = 0 \tag{2.35}$$

with k_p and k_v gains for the position and the velocity and

$$\epsilon_x = \mathbf{x} - \boldsymbol{x}_{\boldsymbol{d}} \tag{2.36}$$

2.1.3 Redundancy

Being m the degrees of freedom of the end-effector and n the degrees of freedom of the manipulator, the latter is said to be *redundant* if $n > m$. The extent of the manipulator redundancy is $(n-m)$. In this definition, redundancy is a characteristic of the manipulator. Another kind of redundancy is the so called *task redundancy.* A manipulator is said to be redundant with respect to a task if the number of independent parameters m_{task} needed to describe the task configuration is smaller than n. Generally speaking, redundancy is therefore a relative concept, it holds with respect to a given task.

Redundancies can be exploited in several ways such as to avoid collision with obstacles (in Cartesian space) or kinematic singularities (in joint space), stay within the admissible joint ranges, increase manipulability in specified directions, minimize energy consumption or needed motion torques, etc. Although the number of benefits is substantial the increase in mechanical complexity (more links, transmission, actuators, sensors etc.), costs and complexity of control algorithms have to be taken into consideration.

2.1.3.1 Redundant Manipulators Kinematics

From a *kinematics point of view*, the core problem of dealing with redundancy is the resolution of equation 2.9 where the Jacobian $J(\mathbf{q})$ has more columns than rows. Three different ways of tackling the redundancy resolution problem are:

- Jacobian-based methods.
- Null-space methods.
- Task augmentation methods.

With the first method a solution is chosen, among the infinite possible, minimizing a suitable (possibly weighted) norm. The equation 2.9 can be solved using a weighted pseudoinverse $J(\mathbf{q})^{\#}_W$ of $J(\mathbf{q})$:

$$\dot{\mathbf{q}} = J(\mathbf{q})^{\#}_W\dot{\mathbf{x}} \tag{2.37}$$

where

$$J(\mathbf{q})^{\#}_{W} = W^{-1}J(\mathbf{q})^{T}(J(\mathbf{q})W^{-1}J(\mathbf{q})^{T})^{-1} \tag{2.38}$$

minimizing the weighted norm:

$$\parallel \dot{\mathbf{q}} \parallel^{2}_{W} = \dot{\mathbf{q}}^{T}W\dot{\mathbf{q}} \tag{2.39}$$

A simple choice for W could be $W = I$ where I is the identity matrix. In this case $J(\mathbf{q})^{\#}_{W}$ is coincident with the pseudo-inverse of $J(\mathbf{q})$, minimizing the norm $\parallel \dot{\mathbf{q}} \parallel^{2} = \dot{\mathbf{q}}^{T}\dot{\mathbf{q}}$.

A null-space methodology add a further component $\dot{\mathbf{q}_0}$ to the previous solution, projecting it into the null-space of $J(\mathbf{q})$:

$$\dot{\mathbf{q}} = J(\mathbf{q})^{\#}\dot{\mathbf{x}} + (I - J(\mathbf{q})^{\#}J(\mathbf{q}))\mathbf{q}_0 \tag{2.40}$$

The choice of $\dot{\mathbf{q}_0}$ could lie on the gradient of a differentiable objective function $H(\mathbf{q})$. While executing the time-varying task $\mathbf{x}(t)$ the robot tries to increase the value of $H(\mathbf{q})$. The choice of $H(\mathbf{q})$ could revolve around the increase in manipulability:

$$H_{man}(\mathbf{q}) = \sqrt{det[J(\mathbf{q})J^{T}(\mathbf{q})]} \tag{2.41}$$

Minimizing the distance from the middle points of the joint ranges:

$$H_{range}(\mathbf{q}) = -\frac{1}{2N}\sum_{i=1}^{N}(\frac{q_i - q_{med,i}}{q_{max,i} - q_{min,i}})^2 \tag{2.42}$$

Avoiding obstacles maximizing the minimum distance to Cartesian obstacles:

$$H_{obs}(\mathbf{q}) = \min_{\substack{a \in robot \\ b \in obstacles}} \parallel a(\mathbf{q}) - b \parallel^2 \tag{2.43}$$

In the task augmentation method, the redundancy is reduced or eliminated by adding further auxiliary tasks and then solving with one of the two methods previously presented.

In the case of operational space schemes, the inverse kinematic doesn't have to be resolved. The redundancy still plays an important role as stated in the next paragraphs.

2.1.3.2 Redundant Manipulators Dynamics

For spatial robots $m = 6$. In the case of a *redundant manipulator* the dynamic behavior of the entire system is not completely describable by a dynamic model using operational coordinates. The dynamic behavior of the end-effector itself can still be described by equation 2.7. In this case, this equation could be seen as a "projection" of the system dynamics into the operational space. A set of operational coordinates, describing only the end-effector position and orientation, is obviously insufficient to completely specify the configuration of a manipulator with more than six degrees of freedom. Therefore, the dynamic behavior of the entire system cannot be described by a dynamic model using operational coordinates. This can be seen looking at equation 2.20. This fundamental relationship becomes incomplete for redundant manipulators that are in motion. As previously stated, kinematically speaking a redundancy corresponds to the possibility of displacements in the null space associated with a generalized inverse of the Jacobian matrix without altering the configuration of the end effector. From a dynamic point of view, it is interesting to find a joint torque vector that could be applied without affecting the resulting forces at the end effector. 2.20 could be then rewritten as:

$$\mathbf{\Gamma} = J^T(\mathbf{q})\mathbf{F} + [I - J^T(\mathbf{q})J^{T\#}(\mathbf{q})]\mathbf{\Gamma}_0 \tag{2.44}$$

where $J^{T\#}$ is a generalized inverse of J^T and $\mathbf{\Gamma}_0$ is an arbitrary generalized joint torque vector projected in the null space of J^T through the projector matrix $[I - J^T(\mathbf{q})J^{T\#}(\mathbf{q})]$. It can be shown how among the infinite possible inverses for J^T only one is consistent with the system dynamics.
Multiplying 2.1 by $J(\mathbf{q})A^{-1}(\mathbf{q})$, using the relationship $\ddot{\mathbf{x}} - \dot{J}(\mathbf{q})\dot{\mathbf{q}} = J(\mathbf{q})\ddot{\mathbf{q}}$ and 2.44 lead to:

$$\begin{gathered}\ddot{\mathbf{x}} + (J(\mathbf{q})A^{-1}(\mathbf{q})\mathbf{b}(\mathbf{q},\dot{\mathbf{q}}) - \dot{J}(\mathbf{q})\dot{\mathbf{q}}) + J(\mathbf{q})A^{-1}(\mathbf{q})\mathbf{g}(\mathbf{q}) = \\ (J(\mathbf{q})A^{-1}(\mathbf{q})J^T(\mathbf{q}))\mathbf{F} + J(\mathbf{q})A^{-1}(\mathbf{q})[I - J^T(\mathbf{q})J^{T\#}(\mathbf{q})]\mathbf{\Gamma}_0\end{gathered} \tag{2.45}$$

Underling the relationship between $\mathbf{F}$ and $\ddot{\mathbf{x}}$. It can be seen how the acceleration at the operational point is not affected by $\mathbf{\Gamma}_0$ only if:

$$J(\mathbf{q})A^{-1}(\mathbf{q})[I - J^T(\mathbf{q})J^{T\#}(\mathbf{q})]\mathbf{\Gamma}_0 = 0 \tag{2.46}$$

The generalized inverse of $J(\mathbf{q})$ that satisfies equation 2.46 is said to be *dynamically consistent*. Furthermore this matrix is unique and is given by:

$$\bar{J}(\mathbf{q}) = A^{-1}(\mathbf{q})J^T(\mathbf{q})\Lambda(\mathbf{q}) \tag{2.47}$$

Rewriting the equation of motion in the operational space formulation for redundant manipulators:

$$\Lambda(\mathbf{x})\ddot{\mathbf{x}} + \boldsymbol{\mu}(\mathbf{x},\dot{\mathbf{x}}) + \mathbf{p}(\mathbf{x}) = \mathbf{F} \tag{2.48}$$

it can be noted how the resulting equation is of the same form as 2.7 established for non-redundant manipulators. However, in case of redundancy, the inertial properties are affected not only by the end-effector configuration but also by the manipulator posture:

$$\boldsymbol{\mu}(\mathbf{q},\dot{\mathbf{q}}) = \bar{J}^T(\mathbf{q})\mathbf{b}(\mathbf{q},\dot{\mathbf{q}}) - \Lambda(\mathbf{q})\dot{J}(\mathbf{q})\dot{\mathbf{q}} \tag{2.49}$$

$$\mathbf{p}(\mathbf{q}) = \bar{J}^T(\mathbf{q})\mathbf{g}(\mathbf{q}) \tag{2.50}$$

It is interesting noting how equation 2.48 is the projection of 2.1 by $\bar{J}^T(\mathbf{q})$:

$$\bar{J}^T(\mathbf{q})(A(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{b}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q})) = \bar{J}^T\mathbf{\Gamma} \quad \longrightarrow \quad \Lambda(\mathbf{x})\ddot{\mathbf{x}} + \boldsymbol{\mu}(\mathbf{x},\dot{\mathbf{x}}) + \mathbf{p}(\mathbf{x}) = \mathbf{F}$$

2.2 Grasping

As briefly summarized in Chapter 1 the branch of robotics that deals with the grasping problem is very rich and wide. Here a mathematical model capable of predicting the behavior of the hand and object under the various loading conditions is presented. Several publications, reviews and books are available on the topic such as [46],[13],[47].

2.2.1 Definition of Parameters

The main elements that allow a basic description of the grasping theory are shown in Figure 2.4, where the finger 1 is shown not in contact to better reveal the presence of the reference frame C_i defined later. We suppose punctual contacts between the last

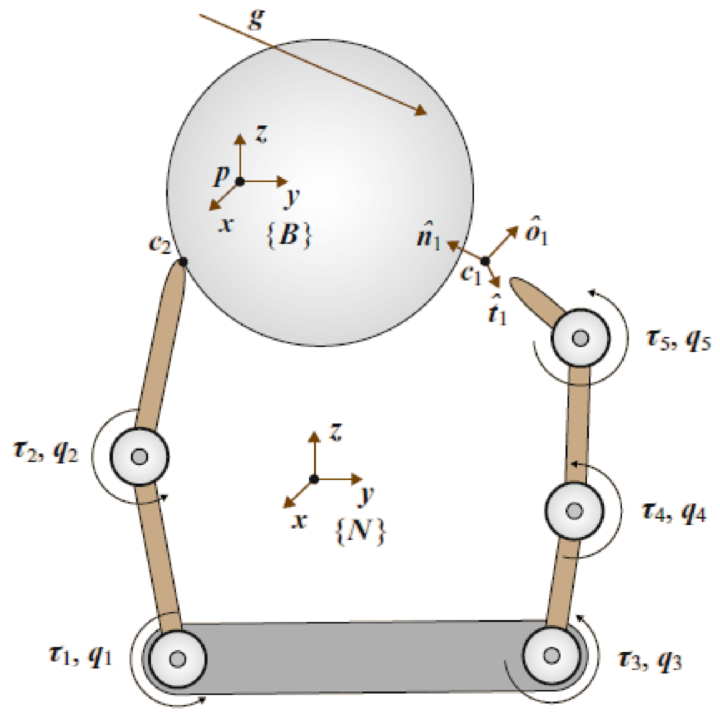


Figure 2.4: Parameter definitions

link of each finger and the object. Set $\{N\}$ the inertial frame fixed in the workspace. The frame $\{B\}$ with center p is fixed to the object and his position from the center of $\{N\}$ is identifiable by the vector $\mathbf{p} \in \mathbb{R}^3$. The center of $\{B\}$ could be selected arbitrarily. Usually, to simplify dynamic analysis, it is chosen in the center of mass of the object. $\mathbf{r}_i \in \mathbb{R}^3$ is the vector that connects the center of $\{B\}$ with the the contact point i^{th}. In each contact point i a reference frame C_i, with axes

$$\begin{bmatrix} \hat{\mathbf{n}}_i & \hat{\mathbf{t}}_i & \hat{\mathbf{o}}_i \end{bmatrix} \tag{2.51}$$

is defined such that $\hat{\mathbf{n}}_i$ is orthogonal to the contact tangent plan, directed toward the object. The orientation of each $\{C_i\}$ w.r.t $\{N\}$ is therefore given by:

$$R_{NC_i} = \begin{bmatrix} \hat{\mathbf{n}}_i & \hat{\mathbf{t}}_i & \hat{\mathbf{o}}_i \end{bmatrix} \tag{2.52}$$

Let denote the vector of joint displacement by

$$\mathbf{q} = \begin{bmatrix} q_1 & \dots & q_{n_q} \end{bmatrix}^T \tag{2.53}$$

where n_q is the number of joints. Similarly the joint torques are defined by

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_1 & \dots & \tau_{n_q} \end{bmatrix}^T \tag{2.54}$$

$\mathbf{u} \in R^{n_u}$ is a vector describing the position and orientation of $\{B\}$ w.r.t. $\{N\}$, n_u is equal to the number of parameters utilized to describe the orientation of $\{B\}$ plus those for the position.

$$\boldsymbol{\nu} = \begin{bmatrix} \mathbf{v}^T & \boldsymbol{\omega}^T \end{bmatrix}^T \tag{2.55}$$

the twist of the object described in $\{N\}$ where $\mathbf{v}$ is the velocity of the center of $\{B\}$ w.r.t. $\{N\}$ and $\boldsymbol{\omega}$ is the angular velocity of the object w.r.t. $\{N\}$. Let $\mathbf{f}_e$ and $\mathbf{m}_e$ be the force applied to the object in p and the applied moment. These two elements allow to define the external wrench applied to the object:

$$\mathbf{g} = \begin{bmatrix} \mathbf{f}_e^T & \mathbf{m}_e^T \end{bmatrix}^T \tag{2.56}$$

2.2.2 Kinematics

Two are the important matrices in grasping theory:

- The *grasp matrix* G whose transpose maps the object twist $\boldsymbol{\nu}$ to the contact frames.
- The *hand jacobian* J_h maps the joint velocities to the twist of the hand expressed in the contact frames.

Let $\boldsymbol{\omega}_{obj}^N$ and $\mathbf{v}_{i,obj}^N$ be respectively the angular velocity of the object and the velocity of the object point coincident with the origin of $\{C_i\}$ w.r.t. $\{N\}$. The following expressions hold:

$$\boldsymbol{P_i} = \begin{pmatrix} I & 0 \\ S(\mathbf{r}_i) & I \end{pmatrix} \tag{2.57}$$

$$\begin{pmatrix} \mathbf{v}_{i,obj}^N \\ \boldsymbol{\omega}_{obj}^N \end{pmatrix} = \boldsymbol{P_i}^T \boldsymbol{\nu} \tag{2.58}$$

where the function $S(\mathbf{x})$ gives the skew symmetric form of the vector $\mathbf{x} \in \mathbb{R}^3$:

$$S(\mathbf{x}) = \begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix} \tag{2.59}$$

and $\boldsymbol{\nu}$ is the object twist referred to $\{N\}$ defined in equation 2.55. It is therefore possible to write the object twist referred to $\{C_i\}$:

$$\boldsymbol{\nu}_{i,obj} = \begin{pmatrix} R_{NC_i}^T & 0 \\ 0 & R_{NC_i}^T \end{pmatrix} \begin{pmatrix} \mathbf{v}_{i,obj}^N \\ \boldsymbol{\omega}_{obj}^N \end{pmatrix} = \hat{R}_{NC_i}^T \begin{pmatrix} \mathbf{v}_{i,obj}^N \\ \boldsymbol{\omega}_{obj}^N \end{pmatrix} \tag{2.60}$$

Combining 2.58 and 2.60 is now possible to define the partial grasp matrix $\tilde{G}_i^T$:

$$\tilde{G}_i^T = \hat{R}_{NC_i}^T P_i^T \tag{2.61}$$

leading to:

$$\mathbf{v}_{i,obj} = \tilde{G}_i^T \boldsymbol{\nu} \tag{2.62}$$

The derivation of the partial hand jacobian associated with the contact i^{th} is similar to what has just been done for the partial grasp matrix. Being $J_{N,i}$ the jacobian matrix that maps the joint velocities into the following vector:

$$\begin{bmatrix} \mathbf{v}_{i,hnd}^N \\ \boldsymbol{\omega}_{i,hnd}^N \end{bmatrix} = J_{N,i}\dot{\mathbf{q}} \tag{2.63}$$

where $\mathbf{v}_{i,hnd}^N$ is the translational velocity of the point of finger in contact with the object w.r.t $\{N\}$ and $\boldsymbol{\omega}_{i,hnd}^N$ is the angular velocity of the last link of the finger i w.r.t $\{N\}$. As done previously is now possible to change the reference to $\{C_i\}$:

$$\boldsymbol{\nu}_{i,hnd} = \hat{R}_{NC_i}^T \begin{pmatrix} \mathbf{v}_{i,hnd}^N \\ \boldsymbol{\omega}_{hnd}^N \end{pmatrix} \tag{2.64}$$

obtaining the partial hand jacobian $\hat{J}_i$:

$$\boldsymbol{\nu}_{i,hnd} = \hat{J}_i\dot{\mathbf{q}} \tag{2.65}$$

$$\hat{J}_i = \bar{R}_{NC_i}^T J_{N,i} \tag{2.66}$$

Repeating the same procedure for all the n_c contacts and stacking twists of the hand and objects all together into the vectors $\boldsymbol{\nu}_{c,hand}$ and $\boldsymbol{\nu}_{c,obj}$ it is possible to obtain the *complete grasp matrix* $\tilde{G}$ and the *complete hand jacobian* $\tilde{J}$:

$$\boldsymbol{\nu}_{c,obj} = \tilde{G}\boldsymbol{\nu} \tag{2.67}$$

$$\boldsymbol{\nu}_{c,hnd} = \tilde{J}\dot{\mathbf{q}} \tag{2.68}$$

where

$$\tilde{G}^T = \begin{pmatrix} \tilde{G}_1^T \\ . \\ . \\ . \\ \tilde{G}_{n_c}^T \end{pmatrix} \qquad \tilde{J}^T = \begin{pmatrix} \tilde{J}_1^T \\ . \\ . \\ . \\ \tilde{J}_{n_c}^T \end{pmatrix} \tag{2.69}$$

2.2.3 Contact Modelling

Depending on the shape and material of the part of the finger in contact with the object, three contact models have been widely used:

- point contact without friction (*PwoF*);
- hard finger (*HF*);
- soft finger (*SF*).

Each model equates specific components of the contact twists of the hand and of the object. Analogously, the corresponding components of the contact force and moment are also equated. This last step is done without considering the constraints imposed by friction models and contact unilaterality. The first model, PwoF, supposes a very

small region of contact and a null coefficient of friction. In this case, only the normal components of the translational velocity and force are transmitted to the object.
An HF model is used when the coefficient of friction between the finger and the object is significant but the region of contact is not extended enough to transmit a moment around the normal of the contact. Now, all three translational velocity and forces components are transmitted through the contact.
The last model (SF) is used when the region of contact is big enough to allow the transmission of a moment around the contact normal. Respect to the previous case, also the angular velocity and moment components around the contact normal are transmitted.
The matrix H_i, for the contact i^{th} is defined so as to represent mathematically the previous model:

$$H_i(\boldsymbol{\nu}_{i,hnd} - \boldsymbol{\nu}_{i,obj}) = \mathbf{0} \tag{2.70}$$

The matrix H_i changes both in dimension and value in function of the type of models:

$$H_{i,PwoF} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad H_{i,HF} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$H_{i,SF} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{2.71}$$

Identifying the matrix H_i for all the n_c contacts allows to construct the H matrix defined as:

$$H = \textbf{Blockdiag}(H_1, ..., H_{n_c}) \tag{2.72}$$

which gives:

$$H(\boldsymbol{\nu}_{c,hnd} - \boldsymbol{\nu}_{c,obj}) = \mathbf{0} \tag{2.73}$$

Combining 2.73 with 2.67 and 2.68 leads to:

$$\begin{pmatrix} J_h & -G^T \end{pmatrix} \begin{pmatrix} \dot{\mathbf{q}} \\ \boldsymbol{\nu} \end{pmatrix} = \mathbf{0} \tag{2.74}$$

where the grasp matrix and hand jacobian are:

$$G^T = H\tilde{G}^T \tag{2.75}$$

$$J_h = H\tilde{J} \tag{2.76}$$

This leads to the important relationship defined as the *fundamental grasping constraint* that relates velocities of the finger joints to velocities of the object [46]:

$$J_h\dot{\mathbf{q}} = \boldsymbol{\nu}_{cc,hnd} = \boldsymbol{\nu}_{cc} = \boldsymbol{\nu}_{cc,obj} = G^T\boldsymbol{\nu} \tag{2.77}$$

Where $\boldsymbol{\nu}_{cc}$ is the vector that contains only the components of the twists that are transmitted by the contact.

2.2.4 Dynamics and Equilibrium

In the previous section, the grasp kinematics was described. Here it is shown how both the hand Jacobian and the grasp matrix play an important role in the equations that describe the dynamic of the hand-object system. The dynamic equations are subjected to the constraint 2.74 and they can be written as follow:

$$M_{hnd}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{b}_{hnd}(\mathbf{q}, \dot{\mathbf{q}}) + J_h^T \boldsymbol{c} = \boldsymbol{\tau}_{app} \tag{2.78}$$

$$M_{obj}(\mathbf{u})\dot{\mathbf{v}} + \mathbf{b}_{obj}(\mathbf{u}, \mathbf{v}) - G\boldsymbol{c} = \boldsymbol{g_{app}} \tag{2.79}$$

where the vector $\mathbf{c}_i$ is:

$$\boldsymbol{c} = \begin{bmatrix} \boldsymbol{c_1}^T & ... & \boldsymbol{c_{n_c}}^T \end{bmatrix}^T \tag{2.80}$$

$\mathbf{c}_i$ (named *wrench intensity vector*) contains the force and moment components transmitted through the contact i^{th} to the object, expressed in the contact frame $\{C_i\}$. $M_{hnd}(\mathbf{q})$ and $M_{obj}(\mathbf{u})$ are the inertia matrix respectively of the hand and object, $\mathbf{b}_{hnd}(\mathbf{q}, \dot{\mathbf{q}})$ and $\mathbf{b}_{obj}(\mathbf{u}, \mathbf{v})$ are the velocity-product terms, $\boldsymbol{g_{app}}$ is the external wrench applied to the object and $\boldsymbol{\tau}_{app}$ is the vector of joint torques. It can be notice how $G_i\mathbf{c}_i = \tilde{G}_i H_i \mathbf{c}_i$ is the wrench applied to the object through the i^{th} contact. Imposing the kinematic constraints imposed by the contact models the following relationship holds:

$$\begin{pmatrix} J_h^T \\ -G \end{pmatrix} \mathbf{c} = \begin{pmatrix} \boldsymbol{\tau} \\ \mathbf{g} \end{pmatrix} \tag{2.81}$$

where:

$$\begin{aligned} \boldsymbol{\tau} &= \boldsymbol{\tau}_{app} - M_{hnd}(\mathbf{q})\ddot{\mathbf{q}} - \mathbf{b}_{hnd}(\mathbf{q}, \dot{\mathbf{q}}) \\ \boldsymbol{g} &= \boldsymbol{g}_{app} - M_{obj}(\mathbf{u})\dot{\mathbf{v}} - \mathbf{b}_{obj}(\mathbf{u}, \dot{\mathbf{v}}) \end{aligned} \tag{2.82}$$

It is worth noticing how both the hand Jacobian and the grasp matrix are important for kinematic and dynamic analysis of the grasping problem. The relationship previously obtained are well organized in Figure 2.5.

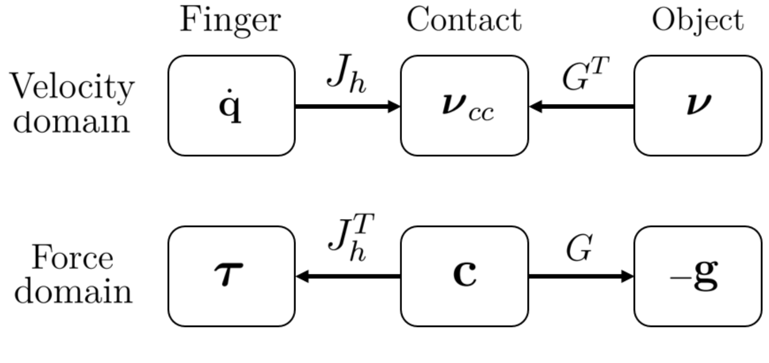


Figure 2.5: Relationship for a multi-finger grasp

2.3 Force distribution

A key problem in robot manipulation is the choice of the forces to apply with the gripper in order to restrain and manipulate the object properly and avoid slippage. This is often referred to as the *force distribution problem.* This has attracted much attention in the past few years being a common problem with other robotic areas. It is possible to categorize the forces applied to the object in two kinds [14]:

- *Internal forces*, also sometimes called the *interaction forces* or the *squeeze forces*, are the contact forces lying in the nullspace of the grasp matrix.
- *Equilibrating forces*, also called *manipulating forces*, are forces that lie in the range space of the grasp matrix and allow to equilibrate the external wrenches and eventually to manipulate the object.

The fingers are required to act in unison so as to apply to the object the exact forces to exert a desired wrench to the object. The choices of the forces have to satisfy specific constraints:

- Fingers can only exert positive or pushing forces.
- Joint efforts must not exceed hardware limits.
- Contact forces must be inside the friction cone associated with the contact.

The first two requirements could be described as linear constraints while the latter is a nonlinear Second-order Conic (SOC) constraint through which any slippage between the finger and the object is avoided. The methods to solve this optimization problem are numerous and many works can be found in the literature. Kerr and Roth proposed in [48] to approximate the cone of friction using a set of planes tangent to the cone itself. This allows linearizing the constraint, looking at the force distribution problem as a Linear Program (LP). On one hand, linearizing the friction-cone constraint leads to an easily solvable optimization problem, on the other hand, inaccuracies are introduced unless many linear constraints are used to approximate each cone, increasing the computational burden. The main downside of this approach is given by the discontinuous behavior of the resulting force profiles in response to an infinitesimal change in robot configuration. If the end-effector is held steady on the border between two different operating regions this could lead to serious instabilities and unacceptable behaviors. A different approach is proposed by Ross et al. in [49]. The authors observed how the cone of friction cone-constraint was analogous to require the positive definiteness of a particular matrix P. This allowed them to formulate the force optimization problem as a Semi-Definite Program (SDP), which is a convex optimization problem. Ross. et al. further analyze the method proposed in [49] adopting, to resolve the SDP problem, a gradient flow solution method in [50] and Dikin-type algorithm in [51]. A major disadvantage of the previous works is the computational complexity required to the solver. The grasping-force optimization problem was solved with a machine learning approach in [52], where the authors minimized a quadratic objective function subject to nonlinear friction-cone constraints using a recurrent neural network. In this thesis, the approach proposed by Borgstrom et al. is adopted [53]. As detailed described in the following section they utilized a Weighted Barrier Function (WBF) force distribution method, *to minimize a weighted sum of two barrier functions to efficiently compute force distributions for redundantly-actuated parallel manipulators subject to non-linear friction constraints.*

2.3.1 The Virtual Linkage

Before analyzing the force distribution algorithm utilized in this thesis a way to give a geometrical description to the internal forces is described. This method was proposed by Khatib and William in [54] where a model for characterizing internal forces and moments during multi-grasp manipulation is proposed. The first application was directed towards multi arm manipulation, the analogy with multi-finger hands is straightforward. The authors suggested representing the manipulated object with a closed-chain mechanism, called *the virtual linkage.*

A virtual linkage associated with an n-grasp manipulation task is a mechanism

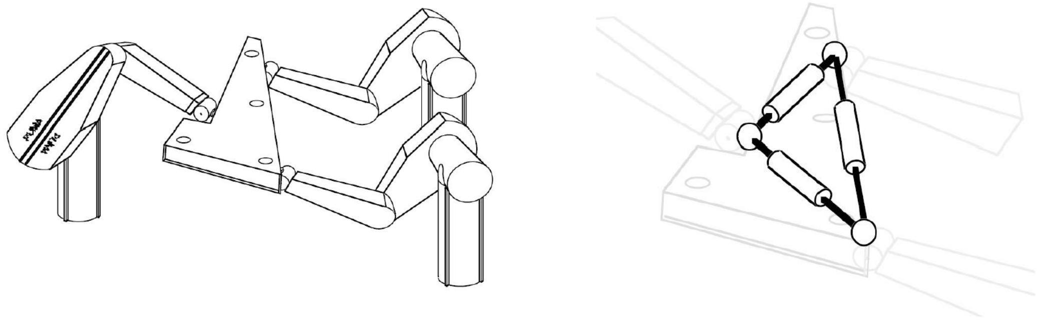


Figure 2.6: The virtual linkage model

with $6(n-1)$ degree-of-freedom. The actuated joints identify the object's internal forces and moments. Forces and moments applied at the grasp points of a virtual linkage cause joint forces and torques at its actuators. When these actuated joints are subjected to the opposing forces and torques, the virtual linkage becomes a statically determinate locked structure. The internal forces and moments in the object are then characterized by these forces and torques. Considering an object manipulated by three arms (or three fingers) the kinematic structure of the virtual linkage is shown in Figure 2.6. It is composed of three actuated prismatic joints, connected by passive revolute joints, which form a 3-dof closed-chain mechanism. To model internal moments a spherical joint with three actuators is placed at each grasp point. In the case of multi-finger grasping with HF contact model the moments transmitted to the object are null, therefore the three spherical joints are treated as passive. Before describing the details of the virtual linkage model, it is worth mentioning how the approach and the elements utilized are really similar to the one described in the previous chapter. In the following paragraphs, each component is expressed with respect to the inertial frame $\{N\}$, no contact frame is defined as it was done in section 2.2.1. The relationship between the elements obtained in the following and those utilized in the previous section is analyzed later. Considering Figure 2.7, $\mathbf{e}_{ij}$ is the unit vector along the link from the i^{th} and the j^{th} contact points and, as already defined in 2.2.1, $\mathbf{r}_i$ is the vector from point 0, analogous to point p, to the contact point i. Let $\mathbf{f}_i$ be the force applied at the grasp point i expressed w.r.t. the

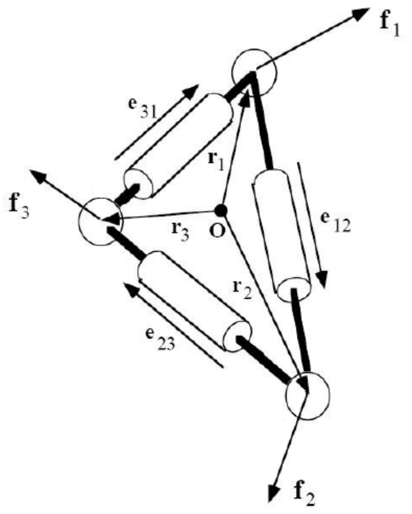


Figure 2.7: Virtual linkage parameters

inertial frame and **f** the vector defined as following:

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_1 \\ \mathbf{f}_1 \end{bmatrix} \tag{2.83}$$

It is worth noting that $\mathbf{f} \neq \mathbf{c}$ given that the first vector is composed of the forces $\mathbf{f}_i$ applied at the contact i^{th} expressed w.r.t. $\{N\}$ and not w.r.t the reference frame $\{C_i\}$ defined in 2.51. As stated before we can decompose the forces applied into internal **t** and manipulating forces $\boldsymbol{f_m}$. The following equation holds:

$$\mathbf{f} = E\mathbf{t} + \boldsymbol{f_m} \tag{2.84}$$

$$E = \begin{bmatrix} -\mathbf{e}_{12} & 0 & \mathbf{e}_{31} \\ \mathbf{e}_{12} & -\mathbf{e}_{23} & 0 \\ 0 & \mathbf{e}_{23} & -\mathbf{e}_{31} \end{bmatrix} \tag{2.85}$$

Solving for **t** gives:

$$\mathbf{t} = \bar{E}(\mathbf{f} - \mathbf{f}_m) \tag{2.86}$$

where

$$\bar{E} = (E^T E)^{-1} E^T \tag{2.87}$$

and

$$\mathbf{t} = \bar{E}\mathbf{f} \tag{2.88}$$

given that $\mathbf{f}_m$ doesn't contribute by definition to the internal forces. Closely to what was done in the previous section, we define the matrix W as:

$$W = \begin{bmatrix} I_3 & I_3 & I_3 \\ \hat{\mathbf{r}_1} & \hat{\mathbf{r}_2} & \hat{\mathbf{r}_3} \end{bmatrix} \tag{2.89}$$

where $\hat{\mathbf{r}_i}$ is $S(\mathbf{r}_i)$ with S function defined in 2.59.
Given the matrix W the following equation holds:

$$\begin{bmatrix} \mathbf{f}_r \\ \mathbf{m}_r \end{bmatrix} = W\mathbf{f} \tag{2.90}$$

where $\mathbf{f}_r$ and $\mathbf{m}_r$ are the external forces and moments applied to the object. At this stage it is possible to define a square matrix called *grasp description matrix* G_f so that:

$$F_0 = G_f \mathbf{f} \tag{2.91}$$

$$F_0 = \begin{bmatrix} \mathbf{f}_r \\ \mathbf{m}_r \\ \mathbf{t} \end{bmatrix} \tag{2.92}$$

Given the definition of G_f also the following equations hold:

$$G_f G_f^{-1} = \begin{bmatrix} W \\ \bar{E} \end{bmatrix} \begin{bmatrix} \bar{W} & E \end{bmatrix} = I_9 \tag{2.93}$$

where

$$\bar{W} = W^T (W W^T)^{-1} \tag{2.94}$$

From these relationships, we can also show that:

$$\mathbf{f}_m = \bar{W} \begin{bmatrix} \mathbf{f}_r \\ \mathbf{m}_r \end{bmatrix} \tag{2.95}$$

It is now shown the relationships between the elements defined in this section and those proposed in section 2.2. The relations are given for a three-finger grasp with HF contact model, the extension to n fingers is straightforward. Let R be the following matrix:

$$R = \begin{bmatrix} R_{NC_1} & 0 & 0 \\ 0 & R_{NC_2} & 0 \\ 0 & 0 & R_{NC_3} \end{bmatrix} \tag{2.96}$$

Where R_{NC_i} is defined in 2.52. It is possible to rewrite the vector $\mathbf{c}$ as sum of two components, internal and equilibrating forces, analogously to what was done for the vector $\mathbf{f}$.

$$\mathbf{c} = \mathbf{c}_i + \mathbf{c}_e \tag{2.97}$$

where

$$\mathbf{c}_p = G^{\#} \mathbf{F} \tag{2.98}$$

being $\mathbf{F} = -\mathbf{g}$ so that:

$$G\mathbf{c} = \mathbf{F} \tag{2.99}$$

and $G^{\#}$ a generalize inverse of the grasp matrix defined in 2.75. Whereas $\mathbf{c}_e$ lies in the nullspace of G. Let N be a matrix spanning the nullspace of G the following holds:

$$\mathbf{c} = G^{\#}\mathbf{F} + N\mathbf{t} \tag{2.100}$$

The relationships among the elements previously introduced are as follow:

$$G = WR \tag{2.101}$$

$$N = R^T E \tag{2.102}$$

$$G^{\#} = R^T W^{\#} \tag{2.103}$$

$$N^{\#} = E^{\#} R \tag{2.104}$$

Equation 2.102 is going to be used in the following section.

2.3.2 WBF Force Distribution Method

In the following paragraphs, the Weighted Barrier Function (WBF) formulation developed in [53] is explained, the formulas are slightly modified in order to introduce the virtual linkage model. It is worth noting how the overall structure of the method remains the same but the internal forces assume a geometrical meaning. Before doing so a mathematical description of the constraints listed in the previous section is given. Remembering equation 2.99:

$$G\mathbf{c} = \mathbf{F} \tag{2.105}$$

where G is the grasp matrix and c is the vector of contact forces seen with respect to the corresponding contact frames. The first relevant constraint to equation 2.105 is given by the torque limits in each joint actuator. Let $\boldsymbol{\tau}^L$ and $\boldsymbol{\tau}^U$ be the lower and upper joint torque limits the linear constraint is described by:

$$\boldsymbol{\tau}^L \leq J_h^T \mathbf{c} \leq \boldsymbol{\tau}^U \tag{2.106}$$

where J_h is the hand Jacobian defined in 2.76. To avoid slippage the forces applied to the object have to lie into the cone of friction associated with each contact. This defines a SOC constraint defined as follows:

$$\mu_i c_{i_{norm}} \geq \sqrt{c_{i_x}^2 + c_{i_y}^2}, \qquad i = 1, 2, ...n \tag{2.107}$$

where μ_i is the coefficient of friction associated with the contact i^{th} and $c_{i_{norm}}$, c_{i_x} and c_{i_y} are the three components of the vector $\mathbf{c}_i$, the force transmitted by the contact i as shown in Figure 2.8. The grasp optimization problem can be described

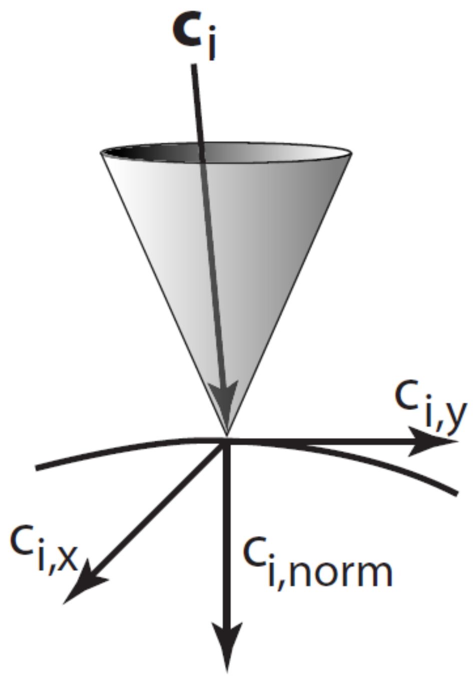


Figure 2.8: Friction cone and force $\mathbf{c}_i$

as the following:

$$\begin{aligned}
&\overset{\star}{\mathbf{c}} = argminf(\mathbf{c}) \\
&s.t.\ \ G\mathbf{c} = \mathbf{F} \\
&\boldsymbol{\tau}^L \leq J_h^T \mathbf{c} \leq \boldsymbol{\tau}^U \\
&\mu_i c_{i_{norm}} \geq \sqrt{c_{i_x}^2 + c_{i_y}^2} \qquad i = 1 \cdots n
\end{aligned} \tag{2.108}$$

where $f(\mathbf{c})$ is the objective function to be minimized. The choice of $f(\mathbf{c})$ has to take into consideration the desire of having forces close to the axes of the friction cone as well as having joint torques as distant as possible from the upper and lower limits.

2.3.2.1 Polyedhral and SOC barrier functions

Let $Q \in R^u$ be a polyhedral set defined as:

$$Q = \{\mathbf{x} \| A\mathbf{x} \leq \mathbf{b}\} \tag{2.109}$$

where $\mathbf{x}$, A and $\mathbf{b}$ represent a set of linear inequality constraints on $\mathbf{x}$. Such linear constraints is usually represented by a barrier function defined as:

$$\Phi(\mathbf{x}) = -\sum_{i=1}^{v} ln(r_i) \tag{2.110}$$

where r_i is the i^{th} element of the vector $\mathbf{r}$ defined as:

$$\mathbf{r} = \mathbf{b} - A\mathbf{x} \tag{2.111}$$

For the linear constraint previously defined the gradient $\nabla\Phi_{lin}(\mathbf{x}) \in R^u$ and the Hessian matrix $\nabla^2\Phi_{lin}(\mathbf{x})$ are defined as follow:

$$\begin{aligned}
\nabla\Phi_{lin}(\mathbf{x}) &= A^T\mathbf{d} \\
\nabla^2\Phi_{lin}(\mathbf{x}) &= A^T(\mathbf{diag}(\mathbf{d}))^2 A
\end{aligned} \tag{2.112}$$

where the i^{th} component of the vector $\mathbf{d}$ is given by $\frac{1}{r_i}$.

In SOCPs is a common practice to split the generic vector $\mathbf{y}$ into a scalar y_0 and a subvector $\hat{\mathbf{y}}$. In particular if the vector $\mathbf{y} = [y_0 \| \hat{\mathbf{y}}]^T$ satisfies $\mu y_0 \geq \|\hat{\mathbf{y}}\|$ it means that $\mathbf{y}$ lies within the Lorentz cone defined by the parameter μ and it is written as $\mathbf{y} \succeq_{soc} 0$. Considering β vectors we define:

$$\tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_\beta \end{bmatrix} \tag{2.113}$$

The barrier function for the conic constraint is then defined:

$$\Phi_{soc}(\tilde{\mathbf{y}}) = -\sum_{i=1}^{\beta} ln((\mu_i y_{i0})^2 - \|\tilde{\mathbf{y}}_i\|^2) \tag{2.114}$$

As done for the linear constraint the gradient and the Hessian of $\Phi_{soc}(\tilde{\mathbf{y}})$ is provided. The gradient of the single vector $\mathbf{y}_i$ is given by:

$$\nabla\Phi_{soc}(\mathbf{y}_i) = \frac{2}{g_i} R_{\mu_i}\mathbf{y}_i \tag{2.115}$$

where:

$$R_{\mu_i} = \begin{bmatrix} -\mu_i^2 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{2.116}$$

and

$$g_i = (\mu_i y_{i0})^2 - \|\tilde{\mathbf{y}}_i\|^2 \tag{2.117}$$

The gradient $\Phi_{soc}(\tilde{\mathbf{y}})$ is given by:

$$\Phi_{soc}(\tilde{\mathbf{y}}) = \begin{bmatrix} \nabla\Phi_{soc}(\mathbf{y}_i) \\ \vdots \\ \nabla\Phi_{soc}(\mathbf{y}_\beta) \end{bmatrix} \tag{2.118}$$

On the other hand the Hessian matrix of $\Phi_{soc}(\mathbf{y}_i)$ is given by:

$$\nabla^2\Phi_{soc}(\mathbf{y}_i) = \nabla\Phi_{soc}(\mathbf{y}_i)\nabla^T\Phi_{soc}(\mathbf{y}_i) + \frac{2}{g_i}R_{\mu_i} \tag{2.119}$$

leading to:

$$\nabla^2\Phi_{soc}(\tilde{\mathbf{y}}_i) = \underset{i=1\ldots\beta}{\mathbf{Blockdiag}}(\nabla^2\Phi_{soc}(\mathbf{y}_i)) \tag{2.120}$$

2.3.2.2 Force Distribution using the WBF

The force distribution problem consists of solving an optimization problem to determine the forces to apply with each finger. In the previous section, a linear and a conic constraint and the barrier function associated with them are described. In the specific case of grasping the linear inequality constraints $A\mathbf{x} \leq \mathbf{b}$ has the following components:

$$A = \begin{bmatrix} J_h^T \\ -J_h^T \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} \tau^U \\ -\tau^L \end{bmatrix} \tag{2.121}$$

The nonlinear friction-cone constraint can be written as $\mathbf{c}_i \succeq_{soc} 0, \quad i = 1\cdots n$. The grasp force optimization can be defined as follow:

$$\overset{\star}{\mathbf{c}} = \underset{\mathbf{c}}{argmin}\,\Phi(\mathbf{c}) \quad s.t.\ G\mathbf{c} = \mathbf{F} \tag{2.122}$$

where:

$$\Phi(\mathbf{c}) = \Phi_{soc}(\mathbf{c}) + \alpha\Phi_{lin}(\mathbf{c}) \tag{2.123}$$

The objective function $\Phi(\mathbf{c})$ consists solely of barrier terms weighted relative to each other through the α parameter. For large α joint efforts are penalized more severely and lower grasp forces are expected, whereas the risk of slippage is higher. Using small values of α larger joint efforts are favored keeping the forces as close as possible to the axis of the correspondent friction-cone. The optimization problem can be further simplified considering equation 2.100, here remembered:

$$\mathbf{c} = G^{\#}\mathbf{F} + N\mathbf{t} \tag{2.124}$$

where N is defined in 2.102 and $\mathbf{t}$ is the vector of internal forces. The problem described in 2.122 becomes:

$$\overset{\star}{\mathbf{t}} = \underset{\mathbf{t}}{argmin}\, \Phi(\mathbf{c}_p + N\mathbf{t}) \tag{2.125}$$

reducing substancially the dimension of he optimization vector. Also the gradiant and the Hessian will change to reflect the change in variables:

$$\begin{aligned} \nabla\Phi(\mathbf{t}) &= N^T(\nabla\Phi_{soc}(\mathbf{c}) + \alpha\nabla\Phi_{lin}(\mathbf{c})) \\ \nabla\Phi^2(\mathbf{t}) &= N^T(\nabla^2\Phi_{soc}(\mathbf{c}) + \alpha\nabla^2\Phi_{lin}(\mathbf{c}))N \end{aligned} \tag{2.126}$$

Being both the gradient and Hessian analytically defined the optimization problem can be solved using the Newton method:

$$\mathbf{t}_{k+1} = \mathbf{t}_k - \gamma\nabla^2\mathbf{\Phi}_k^{-1}\nabla\mathbf{\Phi}_k \tag{2.127}$$

where $\gamma \leq 1$ to avoid leaving the feasible region at each Newton step. The Newton method has quadratic convergence when close to the solution, however, the first guess has to be inside the feasible space or the solution diverges. When this space is large a first guess is easily identifiable taking advantage of the geometric meaning of the vector $\mathbf{t}$. In other cases, a Phase-1 solver can be used to bring the initial guess inside the feasible region and use the Newton method subsequently.

Chapter 3

Arm/hand system

The objective of this thesis revolves around the development of a controller for an arm/hand robotic system in order to successfully and robustly grasp various objects. In this chapter, the main hardware components that constitute the arm/hand robotic system are illustrated. A brief scheme of the controller is shown and the various algorithms utilized to control the system are explained.

3.1 Hardware

The robotic system utilized in this thesis is composed by a 7-dof collaborative arm and a 16-dof torque control robotic hand. An RGBD camera is attached to the last link of the arm (fixed with respect to the hand's palm). The CAD assembly of the robot is shown in Figure 3.1.

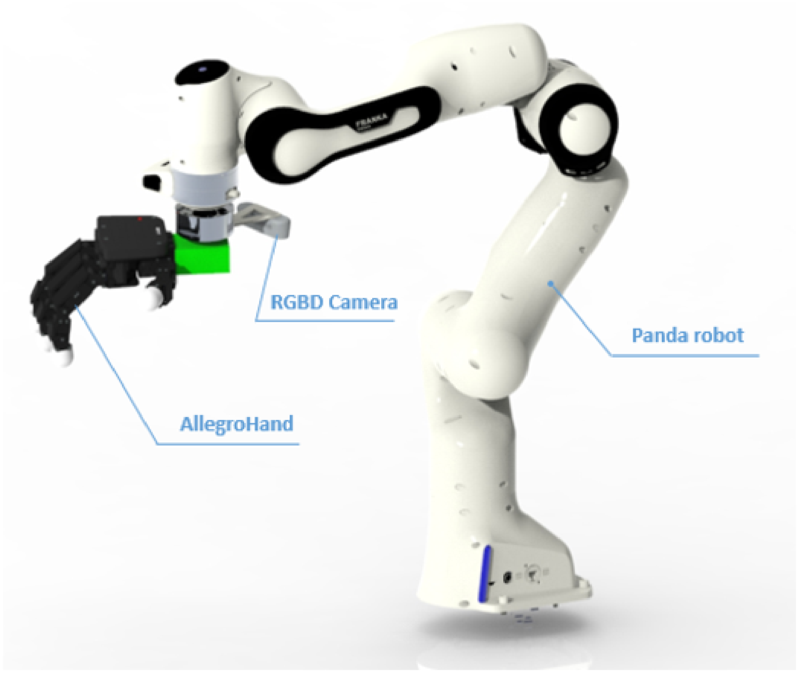
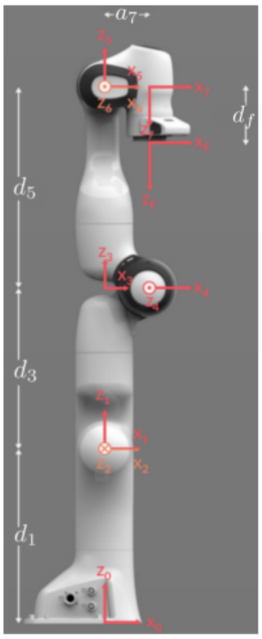


Figure 3.1: Arm/Hand assembly

3.1.1 Arm

The Franka Emika Panda robot is a manipulator that has been attracting a large interest in the robotics and industrial communities. The high usability and relatively low price among the torque-controlled manipulators make it a real competitor in the cobot market. The kinematic parameters according to the modified Denavit Hartenberg convetion are shown in Figure 3.2.



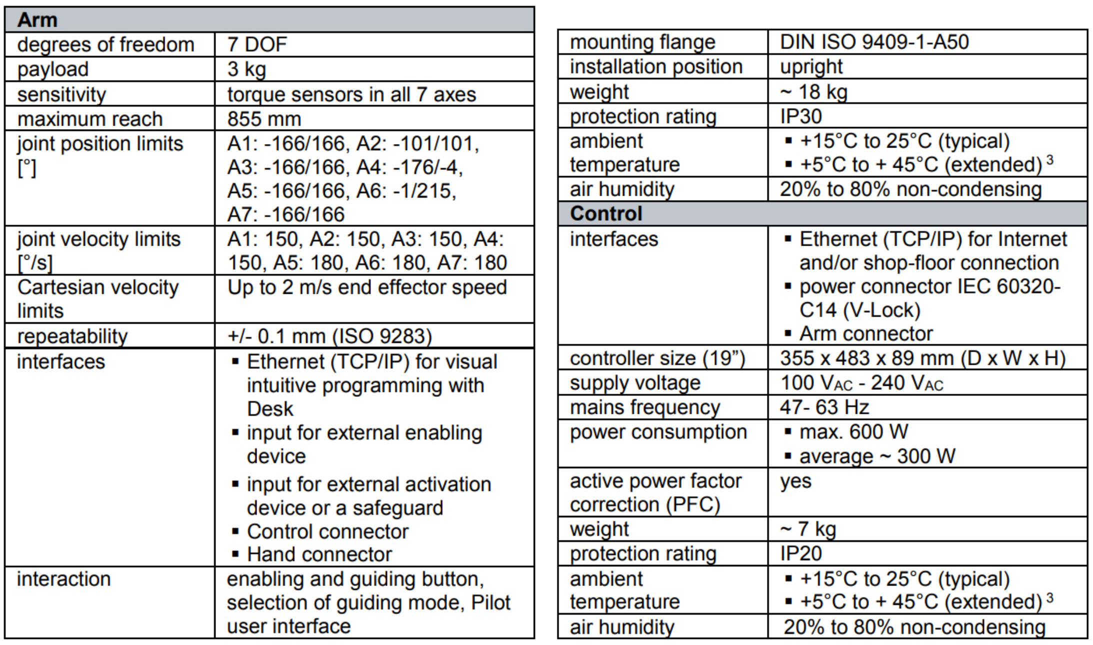
i	a_i	α_i	d_i	θ_i
1	0	0	d_1	q_1
2	0	$-\pi/2$	0	q_2
3	0	$\pi/2$	d_3	q_3
4	a_4	$\pi/2$	0	q_4
5	a_5	$-\pi/2$	d_5	q_5
6	0	$\pi/2$	0	q_6
7	a_7	$\pi/2$	0	q_7
8	0	0	d_f	0

Figure 3.2: Denavit-Hartenberg parameters. In the figure, d1 = 0.333 m, d3 = 0.316 m, d5 = 0.384 m, df = 0.107 m, a4 = 0.0825 m, a5 = −0.0825 m, a7 = 0.088 m.

The robot has 7 revolute joints, each mounting a torque sensor, and a total weight of 18 kg, with a payload up to 3 kg. Further details are shown in Figure 3.3.
It is possible to communicate with the robot through the Franka Control Interface (FCI), which provides the current status of the robot and enables its direct control with an external workstation PC connected via Ethernet. The communication frequency is set at $1kHz$ and it provides the following data:

- Measured joint data, such as the position, velocity and link side torque sensor signals.
- Estimation of externally applied torques and wrenches.
- Various collision and contact information.
- Dynamics: inertia matrix, Coriolis and centrifugal vector and gravity vector.

The robot can be controlled in different modalities. The user can choose to utilize an external controller, providing to the Franka driver only a desired vector $\boldsymbol{\tau}$ of joint torques (gravity compensation is performed by the driver). Otherwise an internal controller can be utilized, selecting a position-mode (by giving a desired joint $\mathbf{q}_d$ or cartesian $\mathbf{x}_d$ position vector) or a velocity-mode (by sending a desired joint $\dot{\mathbf{q}}_d$ or cartesian $\dot{\mathbf{x}}_d$ velocity vector). When an internal controller is utilized the user can



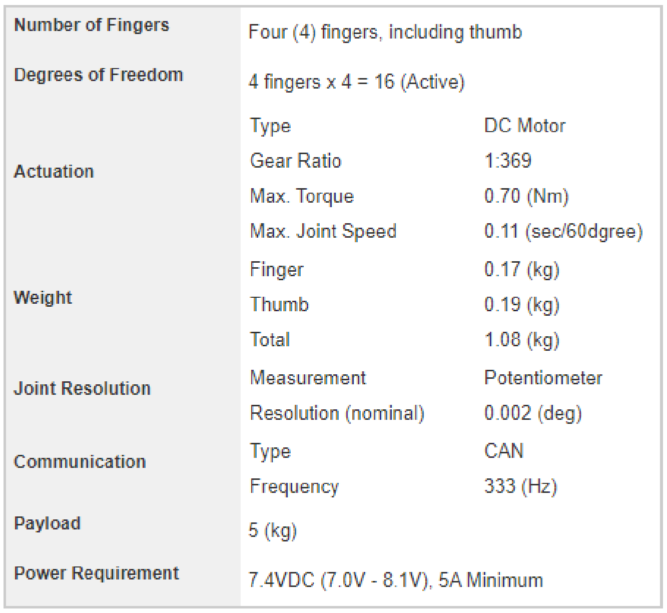
Arm	
degrees of freedom	7 DOF
payload	3 kg
sensitivity	torque sensors in all 7 axes
maximum reach	855 mm
joint position limits [°]	A1: -166/166, A2: -101/101, A3: -166/166, A4: -176/-4, A5: -166/166, A6: -1/215, A7: -166/166
joint velocity limits [°/s]	A1: 150, A2: 150, A3: 150, A4: 150, A5: 180, A6: 180, A7: 180
Cartesian velocity limits	Up to 2 m/s end effector speed
repeatability	+/- 0.1 mm (ISO 9283)
interfaces	▪ Ethernet (TCP/IP) for visual intuitive programming with Desk ▪ input for external enabling device ▪ input for external activation device or a safeguard ▪ Control connector ▪ Hand connector
interaction	enabling and guiding button, selection of guiding mode, Pilot user interface

mounting flange	DIN ISO 9409-1-A50
installation position	upright
weight	~ 18 kg
protection rating	IP30
ambient temperature	▪ +15°C to 25°C (typical) ▪ +5°C to + 45°C (extended) [3]
air humidity	20% to 80% non-condensing
Control	
interfaces	▪ Ethernet (TCP/IP) for Internet and/or shop-floor connection ▪ power connector IEC 60320-C14 (V-Lock) ▪ Arm connector
controller size (19")	355 x 483 x 89 mm (D x W x H)
supply voltage	100 V_{AC} - 240 V_{AC}
mains frequency	47- 63 Hz
power consumption	▪ max. 600 W ▪ average ~ 300 W
active power factor correction (PFC)	yes
weight	~ 7 kg
protection rating	IP20
ambient temperature	▪ +15°C to 25°C (typical) ▪ +5°C to + 45°C (extended) [3]
air humidity	20% to 80% non-condensing

Figure 3.3: Franka Panda specifications

choose between Joint or Cartesian impedance control, with the possibility to modify the proportional and damping gains. A detailed description of the FCI is given at [55].

3.1.2 Hand



Number of Fingers	Four (4) fingers, including thumb	
Degrees of Freedom	4 fingers x 4 = 16 (Active)	
Actuation	Type	DC Motor
	Gear Ratio	1:369
	Max. Torque	0.70 (Nm)
	Max. Joint Speed	0.11 (sec/60dgree)
Weight	Finger	0.17 (kg)
	Thumb	0.19 (kg)
	Total	1.08 (kg)
Joint Resolution	Measurement	Potentiometer
	Resolution (nominal)	0.002 (deg)
Communication	Type	CAN
	Frequency	333 (Hz)
Payload	5 (kg)	
Power Requirement	7.4VDC (7.0V - 8.1V), 5A Minimum	

Figure 3.4: AllegroHand specifications

The robotic hand utilized in this thesis is Allegrohand, an anthropomorphic hand with four fingers. The low cost makes it ideal for robotics research and industrial applications. Each finger has four joints for a total of 16 degrees of freedom and it uses humanoid robotic hand technology developed by the Korea Institute of Industrial Technology (KITECH). A list of feature is proposed in Figure 3.4 whereas its general dimensions are reported in Figure 3.5. The DC motors are open-loop torque

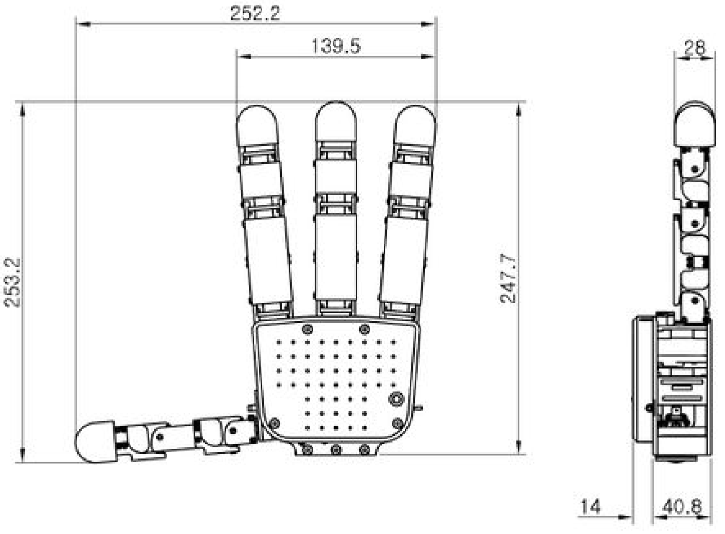


Figure 3.5: AllegroHand dimensions

controlled, the hand is not equipped with force/torque sensors and for this thesis no tactile sensors where utilized. A proper kinematic model of the hand is given at [56]. The tip of each finger has a hemispherical shape and the contact model adopted is of HF type. The vector of torques to compensate for the gravity is provided by the BHand library downloadable at [56]. However, the orientation of the hand has to be given as explained in section 3.2. The AllegroHand is connected to the arm through a 3d-printed interface component.

3.1.3 Vision

In order to detect objects so as to give a pre-grasp configuration, an Intel RealSense D435 was used and attached at the palm of the hand. This device is equipped with two IR-cameras, a laser IR dot pattern projector, an RGB camera and a vision chip that calculates the depth video stream from the two IR-camera video streams. The camera is suitable for outdoor applications and in sunlit conditions. The D435 camera is part of the D400 series by Intel, it has a diagonal field of view that reaches 100° and a resolution of up to 1280×720. The range of operation extends up to 10m while the basic frame rate is 30 frames per second but it can be increased up to 90 frames per second (fps) at lower resolutions. Thanks to its reliance on stereoscopic vision the camera works even without the dot pattern although at the expense of reduced accuracy and with more blank spaces in the depth image. The user is supported by the cross-platform RealSense SDK 2.0, an open-source library that simplifies the application development.

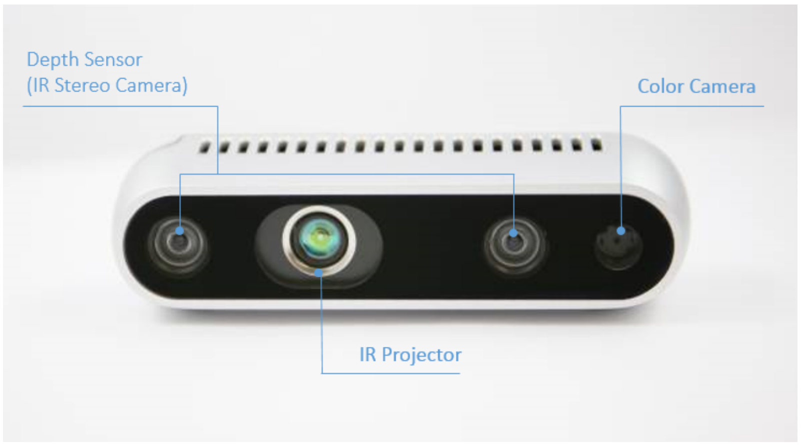


Figure 3.6: Realsense RGBD Camera D435

3.2 Software

In this section a scheme of the robotic system controller is shown. The development of the controller relied on Simulation and Active Interfaces (SAI) developed at Stanford University in collaboration with Google. This is an open-source framework developed to meet the need to integrate control and physically realistic simulation under one umbrella. A brief description of the capabilities of SAI are in 3.2.2. The controller is mainly written in $C++$ whereas the user interface and the vision algorithms are coded in Python. The controller is PC based and the operating system used is Linux Ubuntu where a real-time kernel is installed.

3.2.1 Contol Architecture

The architecture of the controller is shown in Figure 3.7. All the components of the

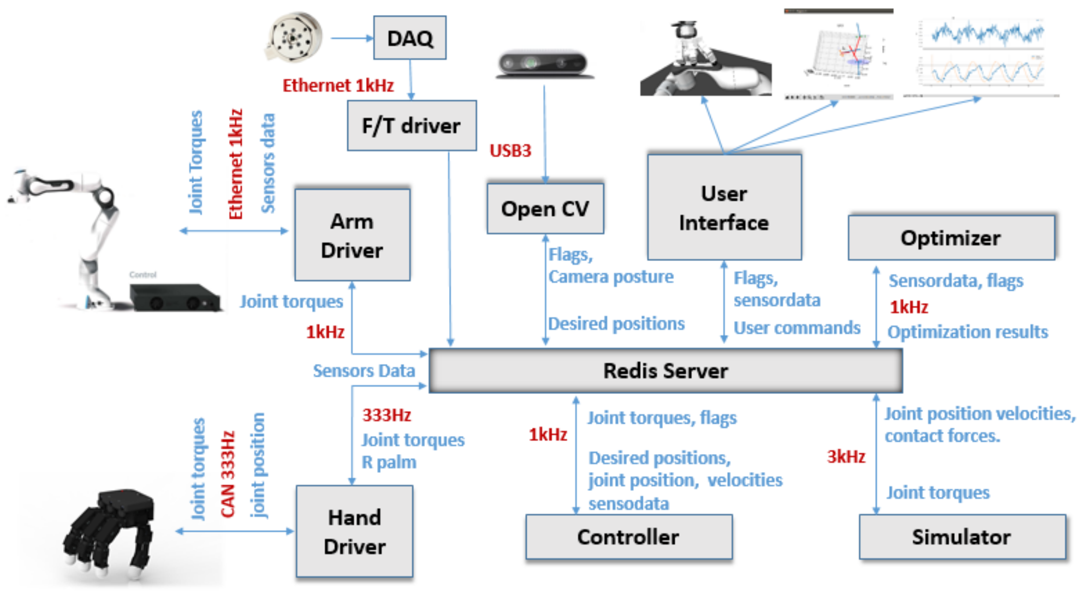


Figure 3.7: Controller Architecture

arm/hand system controller are in communication with a Redis database running on Linux. This is an in-memory key-value database that maps keys to types of values. Differently from other kinds of storage systems, Redis supports both strings and also abstract data types [57].

The *Arm Driver* is an interface component between the main controller and the Franka Control Interface. Safety checks on maximum torques, joint limits and position of the end-effector are run every controller cycle preventing the robot to operate in dangerous conditions.

The *Hand Driver* was downloaded by [56] and it handles the CAN communication (333Hz) with the robotic hand. The gravity joint torques components for the 16-dof of the hand are computed at this stage and they are then added to the torques received by the main controller. Several modifications were applied to the original driver to allow a Redis communication, filtering signals and especially to take into consideration the orientation of the palm so to modify the gravity compensation torques appropriately.

The main *controller* is entirely written in $C++$ and its development plays a major part in this thesis. Here the main control algorithms are run and the joint desired torques are computed. The Sai2 library was used which comprises a few well known and open-source libraries such as a rigid-body dynamic library (rbdl), the Reflexxes library for trajectory generation, the Eigen library for linear algebra. A brief description of the Sai2 library is given later whereas the algorithms implemented in the main controller are shown (only from a conceptual point of view) in Chapter 4.

The *simulator* relies on Sai2 and allows to simulate the robot model, kinematics, and dynamics. This facilitates the development of the controller, saving time and avoiding the use of untested algorithms on the real hardware. Both for the controller and for the simulator a single URDF description of the entire robotic system was written as part of this thesis.

The *optimizer* is a thread written in C++ during the development of this thesis that runs in real-time at a frequency of 1kHz and in which the algorithms shown in section 2.3 were implemented.

To help the development of the algorithms shown in Chapter 4, a simple *user interface*, composed by a couple of functions, was written in Python, allowing to give visual feedback to the user of several key aspects of the algorithms.

The block called *Open Cv* comprises the programs written to interact with the RGBD camera. Its use is explained in Chapter 4.

In order to test some key aspects of the methodologies developed in this thesis a force/torque sensor was integrated into the control architecture.

3.2.2 Sai2

Driving multi degree-of-freedom robot mechanism in unstructured environments in which a potential interaction with humans has to be taken into account is a complex and delicate task. To design new robotic platforms and to test new control strategies a dynamic multi-contact multi-body simulation framework was developed at the Stanford Robotic Lab (Stanford University) in order to have a set of tools to simulate the dynamic behavior and control of virtual robots. The potential of simulating the robotic system allows cheap and fast testing of new control algorithms, it can be helpful in designing new robotic platforms and testing motion planning. What is more, a simulation environment could lead to enormous benefits as a training platform for humans and AI. As far as the first is concerned, to instinctively interact with the simulated environments, the user can operate commands through the use of haptic interfaces. The SAI framework is currently used to develop and simulate a wide variety of mechanisms from mobile robots to human-like structures. The controllers are then developed and initially tested within the simulation framework before being ported to the real platforms. Figure 3.8 shows a functional scheme of the use of a simulation framework in the development of new control strategies. During the development of this thesis the sai2 framework was highly integrated into

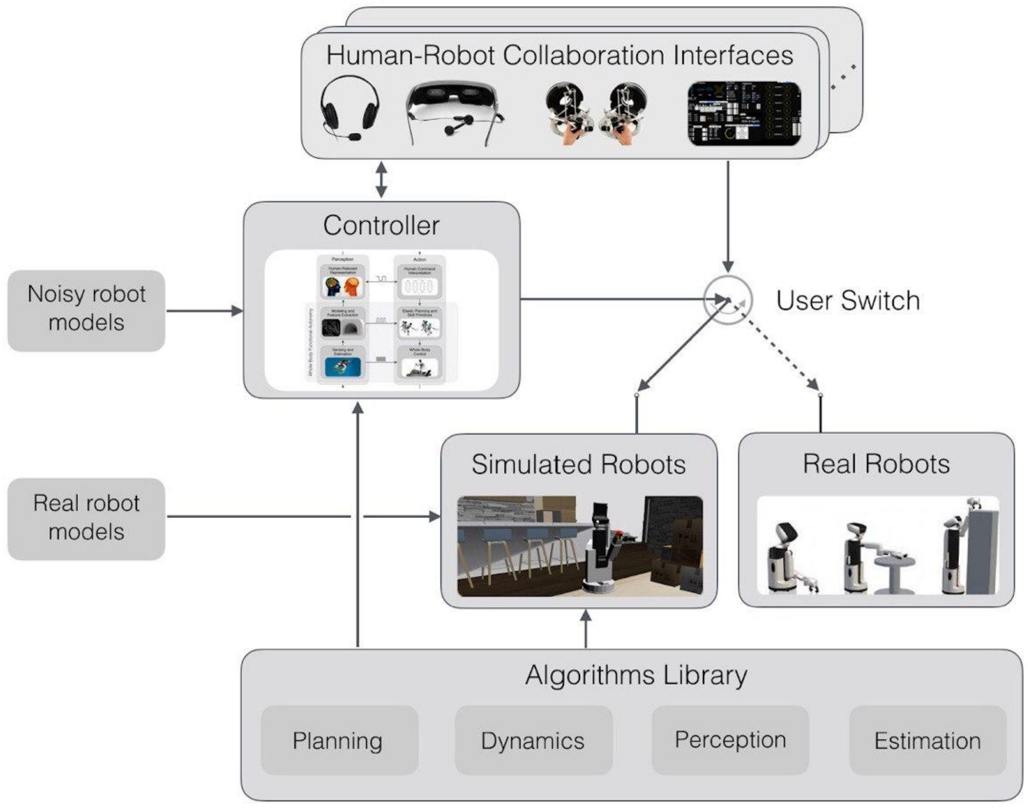


Figure 3.8: Sai2 (Simulation and Active Interfaces)

the control architecture. The possibility to switch from a simulated environment to the real hardware (with the change of only a few flags inside the controller) was implemented helping the testing of the new algorithms studied and implemented throughout the thesis. In addition, the main sai2 modules were utilized to help and accelerate the evolution of the controller. The core modules are listed in the following:

- Sai2-urdfreader: this module allows to build a robot model starting from a URDF (Unified Robot Description Format) of the robotic system presented as an XML file.
- Sai2-model: this is one of the main modules of the sai2 library. It comprises several functions wrapped around the RBDL open-source library to facilitate the computation of kinematic and dynamic parameters of the model (Analytic jacobians, geometric jacobian, position and rotations of links with respect to the world frame, mass matrix, ecc. ecc.).
- Sai2-simulation: this module comprises a physics engine where the interaction of several robot models and objects can be simulated.
- Sai2-graphics: this module is based on the Chai3 library and allows visualization and scene rendering for showing the simulation results.
- Sai2-common: A library that implements a set of utility functionalities to simplify robot simulation and control such as the filer module (to filter signals using the Eigen library), the force/sensor module (to simulate a force sensor when using sai2-simulation) and the Redis module (Redis functions with Eigen objects to communicate data between programs)
- Sai2-primitives: a control library that provides an implementation of basic tasks and primitives for a robot. Mainly based on the Operational Space framework, it offers the possibility to be used with the Reflexxes library for trajectory generation (Type II and IV).

Chapter 4

Robust Grasp Algorithms

It is well-known that we, as human beings, strongly rely on touching and interacting with the object in the process of performing a grasping task. This helps us to gather important information to improve what at first is only a vision and experience-based knowledge of the object we want to grasp. In this thesis, it is shown how an active interaction between a multi-finger torque-controlled robotic hand and the object is essential to improving the way the robot is going to perform the grasping task. Despite not using tactile sensors or force-torque sensors, it is shown that interacting with the object can increase the amount of information utilized by the hand to perform the grasp. This work can, therefore, be seen as an attempt to underline the importance of exploring the object and actively take advantage of the clues gathered after an active contact interaction between the hand and the object itself. In this chapter, two methods that rely on the torque-control capability of the hand are illustrated. The first method, here called *approaching detection* helps the arm/hand system to detect when the hand got in contact with the object. It is reminded that no force sensors or tactile sensors are installed. The second method gives a good estimation of the contact normals and it is here called *normal detection* method. As it is explained in the coming sections, knowing the contact normals helps to select the right forces during the *force distribution* resolution.

4.1 Experimental setup

Before describing the algorithms, a description of the system utilized to test said algorithms is presented. Panda Franka Emika robot are utilized. A kinematic model of the two robots is written, allowing the position and orientation of the second robot end-effector relative to the arm/hand robotic system to be known. A picture of the system is shown in Figure 4.1. Both robots are equipped with a force/torque (F/T) sensor. The F/T sensor installed in the arm/hand robot is named *sensor 1* and, as explained in the next paragraphs, it will not be used during the experiments. The F/T *sensor 2*, installed in the second arm, gives information about the forces applied by the hand to the black target (plastic black bar) attached to the last link of the second robot. In Figure 4.1, two arrows are shown. The green one is obtained by direct kinematic computation of the second arm and it represents the normal of the black target. The blue arrow is the approximated normal given by the *normal detection* method as described in Section 4.3 A single controller was used to control both robots. The controller architecture is similar to what explained in Figure 3.7

where a second robot, its driver and another F/T sensor have to be incorporated.

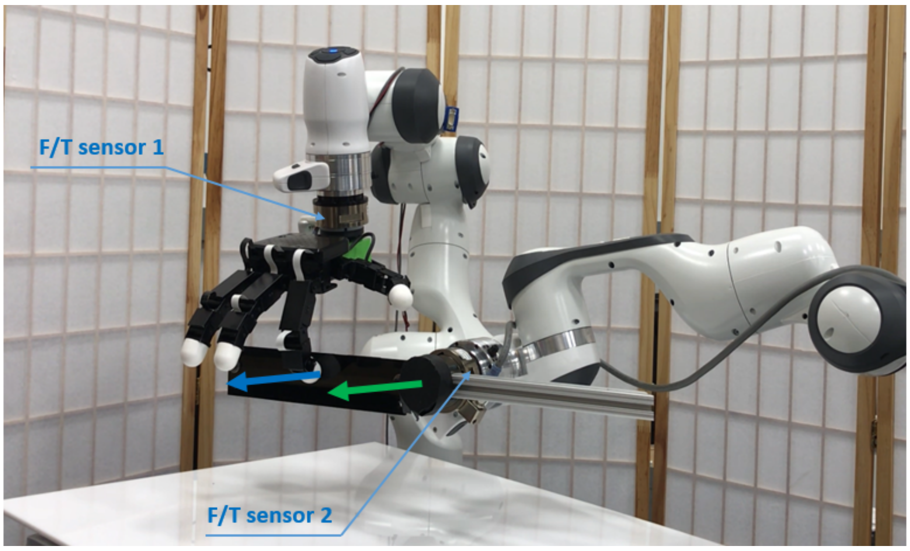


Figure 4.1: Experimental setup

4.2 Approaching Detection

This first method plays a key role in leading the robotic system to understand when the object is touched and to put an end to the approaching movement of the palm. Not having either a tactile sensor nor a force sensor installed in the finger the method has to rely only on the kinematic model of the hand and on its compliant characteristics. Moving the arm and compensating for the gravity effects on the fingers, it is possible to detect the first touch applying miniscule forces to the object (smaller than 0.2N). As stated in Chapter 3 and how it can be seen by Figure 4.1, a force/torque sensor was installed between the arm and the hand. The analytical dynamic model of the hand is not given, despite being embedded in the hand driver to calculate the gravity compensation torque. This does not allow the compensation for the forces and moments that arise during the arm movement due to the inertia of the hand and its changing in orientation with respect to the gravity vector. For this reason, the signal coming from the force/torque sensor 1 was not utilized. The method has to rely only on the sensor data from the joint encoders and the kinematic model of the arm/hand system. Despite this limitation, the hand is able to detect the touch with great precision and delicacy.

4.2.1 Method Description

Let $\{Q\}$ be a reference frame fixed w.r.t the palm of the hand and $\{K_i\}$ be the reference frame fixed with respect to the last link of the i^{th} finger. The tip of each finger of the AllegroHand have a hemispherical shape and $\{K_i\}$ is located for convenience at its center. The vector $\mathbf{b}$ connects $\{Q\}$ to $\{K_i\}$ while $\mathbf{d}$ is a constant vector with origin in $\{Q\}$ and equal to the vector $\mathbf{b}$ at time $t = 0$.
The fingers are kept in gravity compensation mode and the hand is moved close to the object as shown in Figure 4.2. Once the i^{th} finger touches the object, the

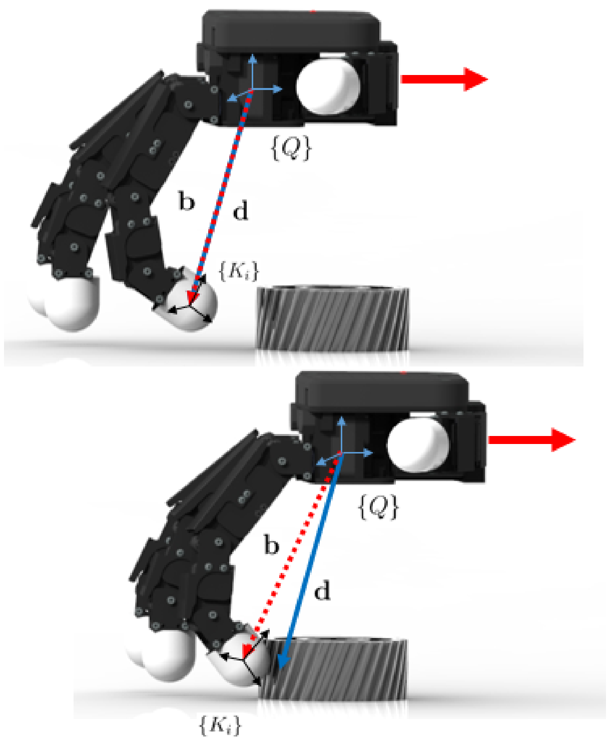


Figure 4.2: *Approaching detection* method

combination of a good backdrivability and the torque-control characteristics of the hand lead to a compliant touch without disturbing the object. The hand keeps moving and, as the tip of the finger is touching the object, the relative position of $\{K_i\}$ w.r.t. $\{Q\}$ changes with time. As shown in Figure 4.2, the vectors $\mathbf{b}$ and $\mathbf{d}$ are no longer the same. The controller computes the norm of the vector $\mathbf{l} = \mathbf{b} - \mathbf{d}$ and when it exceeds a specific threshold the contact is detected and the movement of the palm is stopped.

4.2.2 Results

The method is tested using the setup shown in Figure 4.1. The hand was moved towards the black target, mounted at the last link of the second arm, until the finger detected a touch. The objective of these experiments was that of selecting the best approach velocity and the $||\mathbf{l}||$ threshold before applying the methodologies on real objects. Several different thresholds were tested. The main concern is to be able to detect a touch while minimizing object disturbance. Here three graphs are presented in which different thresholds of 1 mm, 2.5 mm and 30 mm were used. The upper part of each picture shows the norm of the vector $\mathbf{l}$ in time, the lower graph shows the norm of the force detected by the F/T sensor 2 installed in the second robot.

In the first two graphs (fig. 4.3,4.4) thresholds of 1 mm and 2.5 mm and an approaching speed of 0.01m/s were used. A touch is detected as soon as $||\mathbf{l}||$ overcomes the red dotted line. To avoid false positives, given by a noisy reading of the finger position, the touch is detected only when $||\mathbf{l}||$ is greater than the threshold for more than a few milliseconds. It is interesting to observe that the way the palm is stopped is not optimized. This leads to the overshoots shown in the upper graphs. Consid-

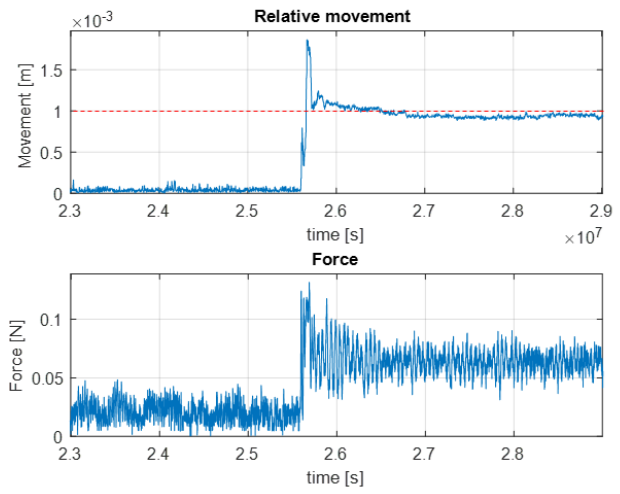


Figure 4.3: *Approaching detection*, 1mm threshold

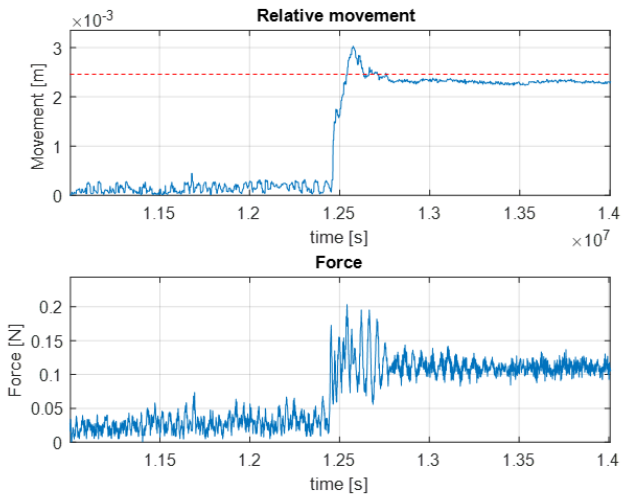


Figure 4.4: *Approaching detection*, 2.5mm threshold

ering the signal collected by the F/T sensor 2, the forces applied to the target are small. With this method, it is, therefore, possible to detect the first touch applying to the object forces smaller than 0.2N.

A higher threshold was tested and set at 30mm. From the graphs in Figure 4.5, it is evident that only the force requested to overcome the friction in the finger is applied to the target and it does not reach values over 0.5N. A threshold of 3 mm was chosen as appropriate for real object applications.

A study of the impact force as a function of the palm velocity approach was conducted. The hand was moved towards the target at different velocities and the maximum force detected by the F/T sensor 2 was memorized. It is observed that the contact is impulsive and no characterization of the F/T sensor was included. For this reason, the maximum force detected could be an overshoot given by the dynamic behavior of the sensor, going beyond the real maximal force applied to the target. It was chosen to consider these peaks as real forces without applying any filter to the sensor signal. In Figure 4.6 the maximum forces detected, varying the

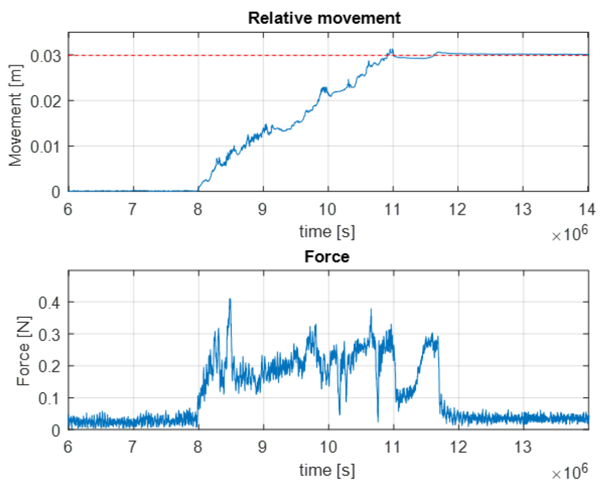


Figure 4.5: *Approaching detection*, 30mm threshold

approaching velocity, are illustrated.

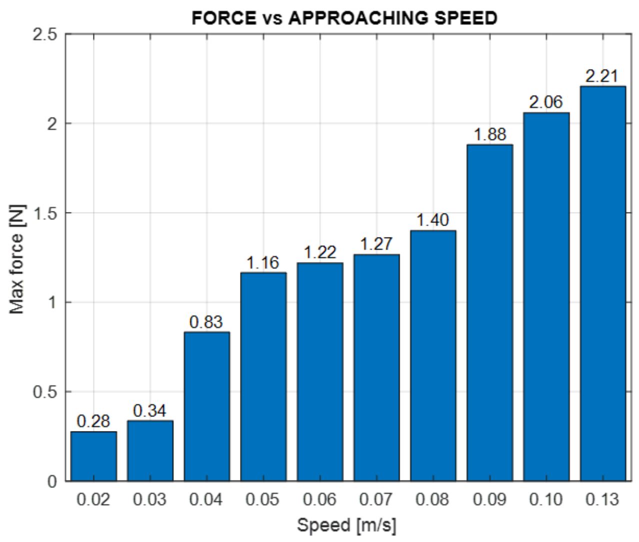


Figure 4.6: *Approaching detection*, different velocities

4.3 Normal detection

In resolving the *force distribution* problem the constraints listed in Section 2.3 have to be met to avoid slippage and excessive motor torques. To be able to apply the WBF algorithm the reference frames $\{C_i\}$ have to be identified. To do so the following information has to be known:

- Position of the contact i.
- Direction of the normal $\hat{\mathbf{n}}_i$.

Without using force or tactile sensors, detecting the point of contact is not trivial. When a finger i is in contact with an object the only information obtainable by the kinematic model of the arm/hand system is the position of the reference frame $\{K_i\}$. In this section, it is shown that it is possible to approximate the point of contact and the direction of $\hat{\mathbf{n}}_i$, resulting in a good resolution of the *force distribution* problem. What is more, the position of the contact i and the vector $\hat{\mathbf{n}}_i$ are not independent. Once the former is known, the hemispherical shape of the finger allows detecting the vector $\hat{\mathbf{n}}_i$, which is parallel to the line that connects the center of the hemisphere to the point of contact found. Hence, the problem reduces to finding only the position of the contact i.

4.3.1 Method description

As a starting point for this method, it is supposed that the finger i is in contact with the object, the position of $\{K_i\}$ is known and it is identified by the point $k_{i,0}$, whose position is written w.r.t. $\{N\}$. However, at this moment in time, there are no ways to determine what part of the finger is in contact with the object, hence, it is not possible to define the reference frame $\{C_i\}$. To be able to define these reference frames, the method consists in touching the object several times around the initial point. Every time the object is touched, the position of $\{K_i\}$ is recorded, providing a collection of N points $k_{i,j}$. The plane t that best fits these N points is parallel to the contact plane c at a distance of R, where R is the radius of the hemisphere. This idea is shown in Figure 4.7a for a plane surface and in Figure 4.7b for a random surface (in a 2D environment). A 3D representation of this concept is shown in Figure 4.8. To collect N different $k_{i,j}$ points the finger has to touch the

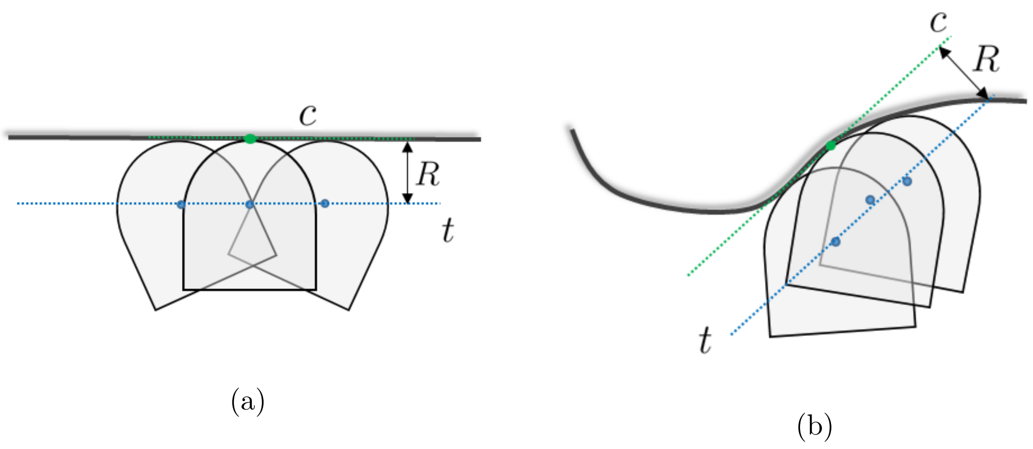


Figure 4.7: *Normal detection* method, 2d representation

object around the initial position other $N-1$ times, quickly and without disturbing the object. Every time, the finger is moved away from the object and it is then pushed back against it, in a position close to the previous one. To describe how this is achieved, the following quantities are defined.

- Let $\{L_i\}$ be the reference frame shown in Figure 4.9 where a schematic of the finger i is illustrated (depending on whether the finger used is the thumb or not). The reference frame $\{L_i\}$ is composed of the three unit vectors $\hat{\mathbf{l}}_{1i}$, $\hat{\mathbf{l}}_{2i}$ and $\hat{\mathbf{l}}_{3i}$ where $\hat{\mathbf{l}}_{3i}$ is parallel to the axis of the first joint and $\hat{\mathbf{l}}_{2i}$ is parallel to the axis of the last joint.

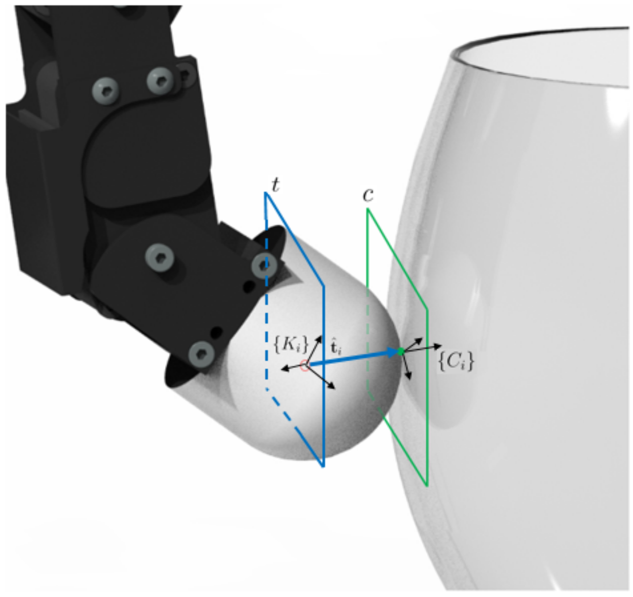


Figure 4.8: *Normal detection* method, 3d representation

- As defined previously, $k_{i,0}$ is the position of the $\{K_i\}$ reference frame w.r.t the world frame $\{N\}$ when the finger is in contact with the object for the first time.
- The plane p lies at a distance s_i from the point $k_{i,0}$ (moving away from the object) and it is parallel to the plane formed by $\hat{\mathbf{l}}_{2i}$ and $\hat{\mathbf{l}}_{3i}$.
- As soon as $k_{i,0}$ is recorded, N different points $(p_{i,0}, p_{i,1}, ..., p_{i,N-1})$ that lie on the plane p are defined. In particular, $p_{i,0}$ and $p_{i,N-1}$ coincides and are exactly s_i mm away from the $k_{i,0}$ point. Their position w.r.t. the world frame is $[k_{i,0} + \mathbf{s}_i]$ where the vector $\mathbf{s}_i$ is parallel to $\hat{\mathbf{l}}_1$, is expressed in $\{N\}$ and its module is s_i. The other $p_{i,j}$ points are located on a circle of radius r_i with center in $p_{i,0}$ (right side of fig. 4.10).

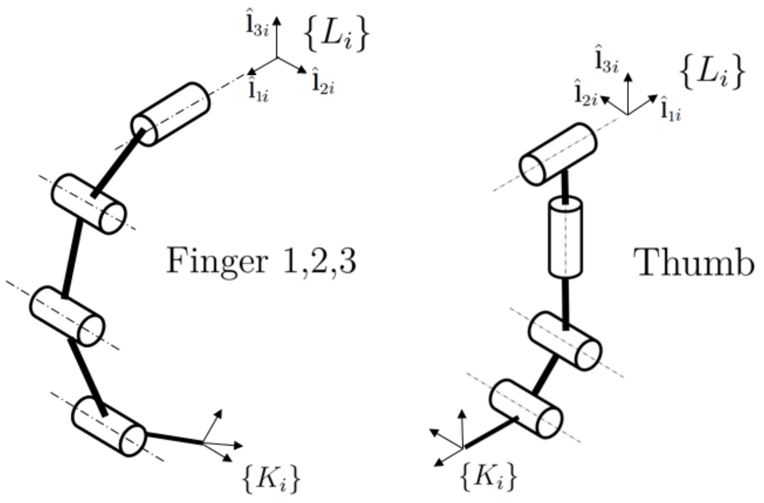


Figure 4.9: Fingers kinematic schemes

With this quantities defined, the method can be summarized in the following steps:

- The finger is moved away from the object so that $\{K_i\}$, firstly located at $k_{i,j}$, finds itself at $p_{i,j+1}$.
- The finger is moved back towards the object. After having touched the object the new position $k_{i,j+1}$ is found.
- The first two steps are repeated until $\{K_i\}$ finds itself at $p_{i,N-1}$ and then, touching the object, the $k_{i,N-1}$ is found. It is worth noting how $k_{i,N-1}$ is generally different from $k_{i,0}$, although they are generally really close to each others.
- N different $k_{i,j}$ points have already been collected and the plane t_i is calculated.
- The point of contact that lies on the surface of the tip of the finger C_i is found and its position w.r.t $\{N\}$ is $[k_{i,N-1} + R * \hat{\mathbf{t}}_i]$ where $\hat{\mathbf{t}}_i$ is the unit vector perpendicular to the plane t (fig. 4.8).

The method can be better understood looking at Figure 4.10. $p_{i,j}$ are identified by the red points and are uniquely defined after $k_{i,0}$ is collected. As previously said, $p_{i,0}$ and $p_{i,N-1}$ can be found starting from $k_{i,0}$ through the vector $\mathbf{s}_i$, yellow in fig 4.10. The plane t is found by best fitting the $k_{i,j}$ points, shown in blue. For this reason the generic $k_{i,j}$ does not necessarily lie on t and the latter has no reason for being parallel to p. The light green points represent the real points in which the object is touched and can not be determined. Only an approximation of the last real point of contact C_i (dark green) is, in fact, given by the method. The method was tested

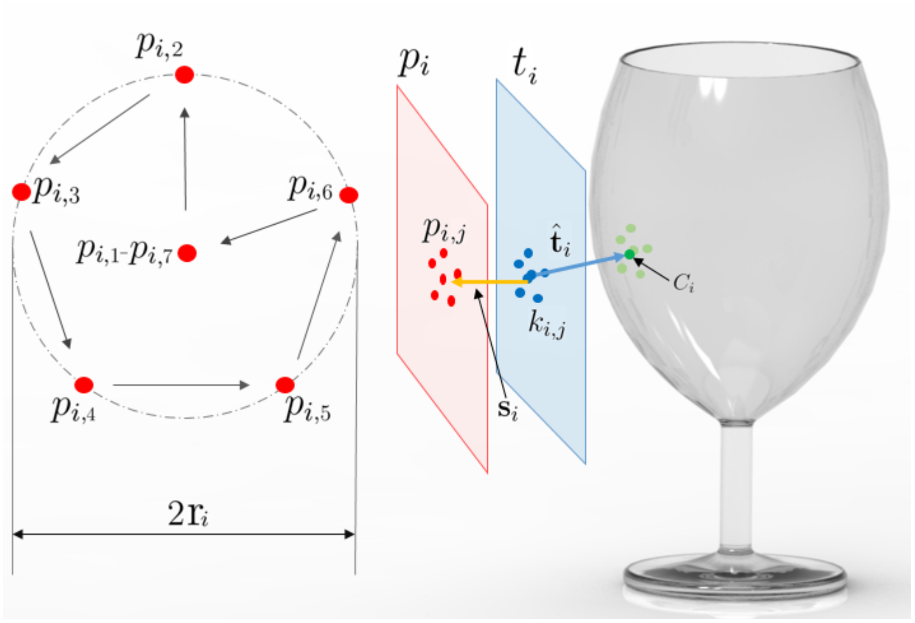


Figure 4.10: Pattern adopted to allow for touching the object N times

utilizing plane surfaces where the contact normal $\hat{\mathbf{n}}_r$ was known. The normal $\hat{\mathbf{n}}_i$ to the plane t_i allows finding the vector $\hat{\mathbf{t}}_i$ and it is coincident with the normal of the contact plane c. The value of $\hat{\mathbf{n}}_i$ was therefore confronted with the real value $\hat{\mathbf{n}}_r$. Before showing the results of such tests a description of the control algorithms used to move the finger in each step of the *normal detection* method are analyzed.

4.3.2 Internal methodologies

In this subsection the following aspects of the method are explained:

- Movement of the finger from $k_{i,j}$ to $p_{i,j+1}$
- Movement of the finger from $p_{i,j}$ to touch the object.
- Computing of the plane t_i.

As far as the first point is concerned, a joint space control approach was adopted. In this case, the objective is moving the finger away from the hand and bringing the frame $\{K_i\}$ to the position $p_{i,j}$. In spite of the good backdrivability of the hand, the friction is still really high. Using a joint space control allows for the use of higher gains, the static friction in each joint is more easily overcome and even small movements of the end-effector are achieved. As explained in Chapter 2 the inverse kinematic has to be solved to transform the desired position of $\{K_i\}$ into the desired joint angles. To avoid dealing with redundancies a joint is blocked and each finger is considered as a 3R manipulator. The analytical solution (without considering

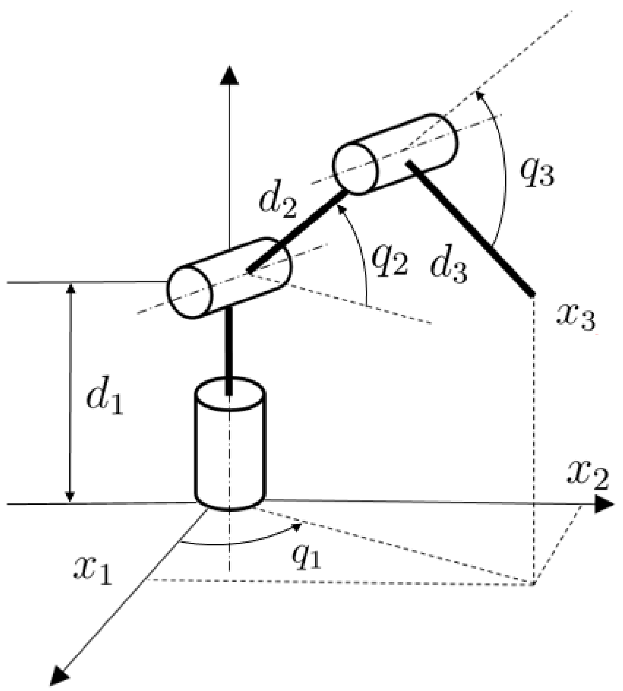


Figure 4.11: Scheme of a 3R manipulator

singularities) for the 3R manipulator shown in figure 4.11 is given as follows:

$$\begin{aligned} x_1 &= c_1(d_2c_2 + d_3c_{23}) \\ x_2 &= s_1(d_2c_2 + d_3c_{23}) \\ x_3 &= d_1 + d_2s_2 + d_3s_{23} \end{aligned} \tag{4.1}$$

are the equations that describe the kinematic of the manipulator. From these equation it is possible to obtain:

$$\begin{aligned} c_3 &= (x_1^2 + x_2^2 + (x_3 - d_1)^2 - d_2^2 - d_3^2)/(2d_2d_3) \\ \pm s_3 &= \pm\sqrt{1 - c_3^2} \end{aligned} \tag{4.2}$$

resulting in two possible solutions for q_3

$$\begin{aligned} q_3^{\{+\}} &= ATAN2\{s_3, c_3\} \\ q_3^{\{-\}} &= -q_3^{\{+\}} \end{aligned} \tag{4.3}$$

Calculating the possible values assumed by q_1 is straightforward:

$$\begin{aligned} c_1 &= x_1/\pm\sqrt{x_1^2 + x_2^2} \\ s_1 &= x_2/\pm\sqrt{x_1^2 + x_2^2} \end{aligned} \tag{4.4}$$

that gives other two solutions for q_1:

$$\begin{aligned} q_1^{\{+\}} &= ATAN2\{x_2, x_1\} \\ q_1^{\{-\}} &= ATAN2\{-x_2, -x_1\} \end{aligned} \tag{4.5}$$

Looking at the shape of the finger it is possible to chose the right values both for q_1 and q_3. The value of q_2 is then given by the following equation:

$$q_2 = ATAN2\{s_2, c_2\} \tag{4.6}$$

where s_2 and c_2 can be found by solving the system:

$$\begin{pmatrix} d_2 + d_3c_3 & -d_3s_3 \\ d_3s_3 & d_2 + d_3c_3 \end{pmatrix} \begin{pmatrix} c_2 \\ s_2 \end{pmatrix} = \begin{pmatrix} c_1x_1 + s_1x_2 \\ x_3 - d_1 \end{pmatrix} \tag{4.7}$$

The analytical inverse kinematic proposed above is easily applicable to all the fingers of the hand. In case the finger controlled is the thumb the first joint is blocked, otherwise the last joint of the finger is blocked. In the controller implementation the movement of the reference frame $\{K_i\}$ was divided in two steps: supposing the current position is $k_{i,j}$ the reference frame $\{K_i\}$ was brought to $p_{i,j+1}$ forcing a passage through the point $p_{i,j}$ limiting the risk to touch the object with the finger during this motion.

After the finger is brought into position $p_{i,j+1}$, it has to be moved back in contact with the object. The following aspects must be achieved:

- The N points $k_{i,j}$ have to be spread as uniformly as possible.
- The forces applied to the object have to be as low as possible.

The choice for the controller should fall into an operational space framework, where the force applied by the finger to the object could be easily controllable. As stated before this control scheme is problematic when the forces are really low given the high static friction in the joints. It was decided to apply a force to the end-effector controlling the torque in a single joint of each finger, leading to a 1-dof force controller. For example, considering the index finger, a torque of 0.07N/m was applied at the second joint (starting with a decreasing ramp to win the static friction). This results in a movement of the frame $\{K_i\}$ around a circle with axis coincident with the second joint axis, until the object is touched (it is worth noting that the short arc covered by the tip of the finger is closed to a straight movement parallel to the vector $\mathbf{s}_i$). Being the distance from this second joint to the tip of the finger roughly 100mm the force applied to the object is smaller than $1/100^{th}$ of Newton. In reality, the finger tends to accelerate during the motion towards the object leading to an impulsive contact. For this reason, the actual perturbation of the object is higher as shown in the next section. However, being the plane p_i generally set a few mm away from the initial point $k_{i,0}$, the kinetic energy accumulated by the finger during its motion towards the object is low and the impact with the object is really delicate.

Once the finger has touched the object for the last time and the $k_{i,N-1}$ is detected the finger is kept at that position and the plane t is fit among the N $k_{i,j}$ points. The plane is found using a Principal Components Analysis (PCA). The centroid m for the finger i is defined as follow:

$$m = \frac{1}{N}\sum_{w=1}^{N} k_{i,w} \tag{4.8}$$

The matrix $\mathbf{Y}$ is defined as follow:

$$\mathbf{Y} = \begin{bmatrix} y_1 & y_2 & \cdots & y_N \end{bmatrix} \tag{4.9}$$

where

$$y_j = x_j - m \tag{4.10}$$

The SVD of the matrix $\mathbf{Y} = UEV^*$ is computed and the normal $\hat{\mathbf{n}}_i$ is given by the last column vector of U.

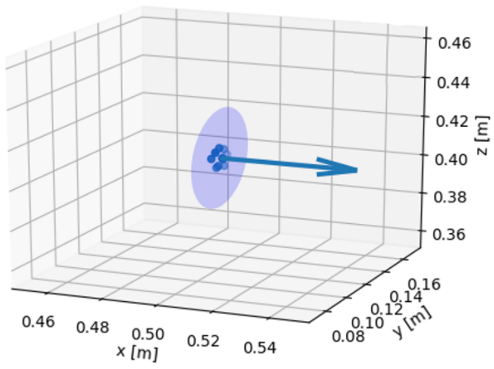
4.3.3 Tests Results

To test the effectiveness of the method the experimental setup described in Section 4.1 was used. The hand was moved against the black target shown in Figure 4.1. His normal $\hat{\mathbf{n}}_r$ is computed through the direct kinematic of the second robot. The hand is moved towards the target like what was done for the *approaching detection* method. Once the first contact is detected the finger carries on probing the surface and the algorithm gives the approximated normal $\hat{\mathbf{n}}$ (shown as a blue arrow). The parameters that characterize the method are the following (the subscript i, used in the previous section to underline the fact that all the values were referred to the single i^{th} finger, is omitted):

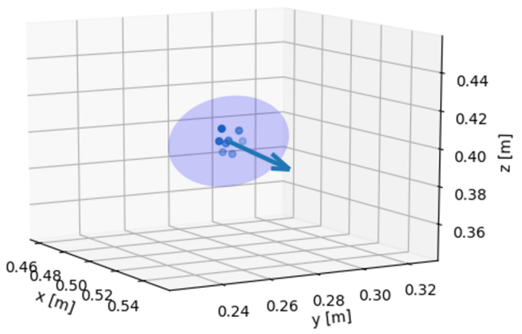
- Value of s.
- Value of r.
- Number of points of contact N.

- Intensity of the torque applied to the active joint to touch the finger.

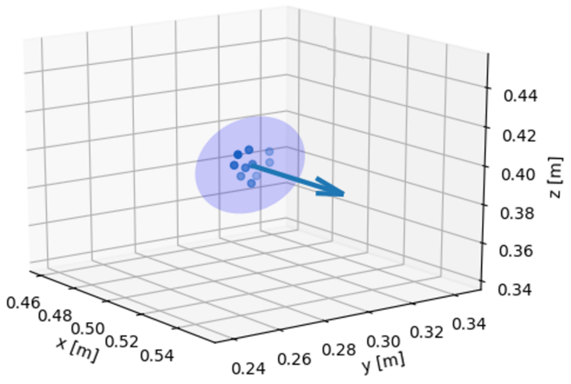
As far as the latter is concerned, as stated previously, a torque of 0.07Nm was selected. This is the lowest torque which gives a movement of the finger no matter what the configuration of the finger itself is. Using lower torques could help to reduce the forces applied to the objects but, given the static friction in the finger, they could fail in moving the finger compromising the method. The value of s is chosen as a function of the r value. The latter plays a key role in the success of the *normal detection* method. Low values of r could hinder the normal approximation: the detection of the plane t could be really affected by noises and imprecision in the collection of the $k_{i,j}$ points. On the other hand, excessive values of r could lead to a bad approximation of the normal surface, especially when the curvature of the object surface is high. A high N value helps in getting rid of noises and imprecision but it increases the time needed to detect the normal. In particular each finger is able to complete the *normal detection* method in around 2 seconds for $N = 6$, 3 seconds when $N = 8$ and 3.6 seconds for $N = 10$.
A real example of the collection of $k_{i,j}$ points for different values of r is shown in Figure 4.12. To select the best combination of parameters several tests were done.



(a) $r = 4$mm, $N = 10$



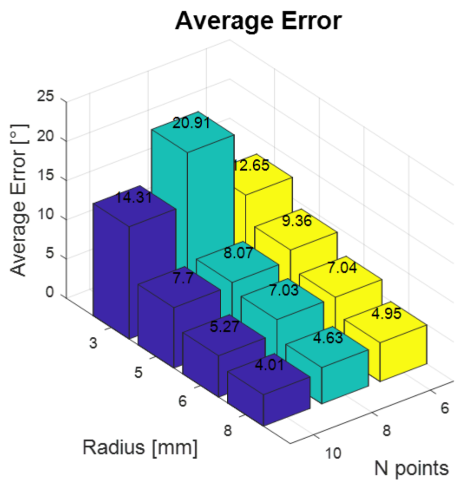
(b) $r = 6$mm, $N = 8$



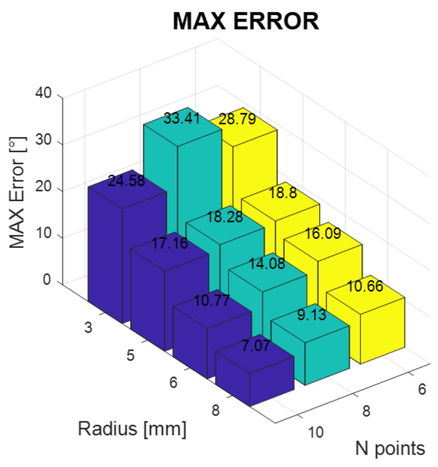
(c) $r = 8$mm, $N = 10$

Figure 4.12: $k_{i,j}$ points collected with the approximated normal $\hat{\mathbf{n}}$

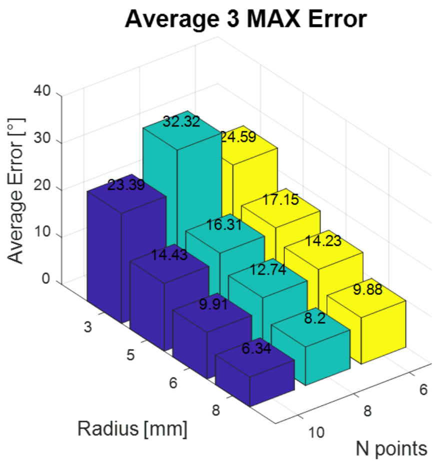
The error expressed in degrees between the approximated normal $\hat{\mathbf{n}}$ and $\hat{\mathbf{n}}_r$ is computed utilizing different combination of N and r. The results are shown in Figure 4.13. For each combination of parameters, 25 tests were done for a total of 300 samples collected. Among all the combinations, a $r = 6$mm and $N = 8$ were chosen



(a) Average error



(b) Maximum error



(c) Average of the 3 highest error encountered

Figure 4.13: Results of the *normal detection* method as a function of the parameters r and N.

as the parameters to be utilized on real objects. This leads to an average error of 7 degrees with a maximum error below 15 degrees. It is worth noting that the error decreases when the number of points N is increased. Being the surface explored flat the normal detection performs better for higher values of r. It is worth noting that for $r = 3$ the $k_{i,j}$ points are too close to each other to give a good approximation of the plane t leading to really high errors in the normal detection.

Another aspect worth exploring is the influence of the angular misalignment α between the surface normal $\hat{\mathbf{n}}_r$ and the vector $\mathbf{s}_i$. This misalignment tends to be close to the angles between the plan p_i and the plane t_i. It is remembered, in fact, that the hand has no prior information about the object surface normal, therefore the definition of the plane p, on which the points $p_{i,j}$ are defined, is only a function of the configuration of the hand. This concept is shown in Figure 4.14 where the plane p, the vector s_i and the real normal $\hat{\mathbf{n}}_r$ are shown in three different configurations. 65 tests were conducted with the parameters chosen (r = 6mm and $N = 8$), identifying the approximation error as a function of the angle α. The results are shown in Figure 4.15, the error tends to increase when α is higher but it maintains

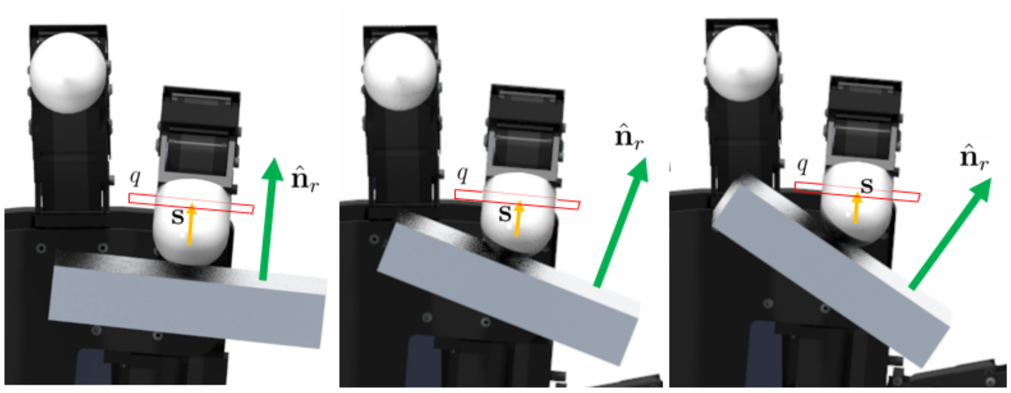


Figure 4.14: Different configuration of the i^{th} finger with respect to $\hat{\mathbf{n}}_r$

acceptable values. What is more, the average error obtained with these samples is really close to the previous tests. The last aspect of the *normal detection* method

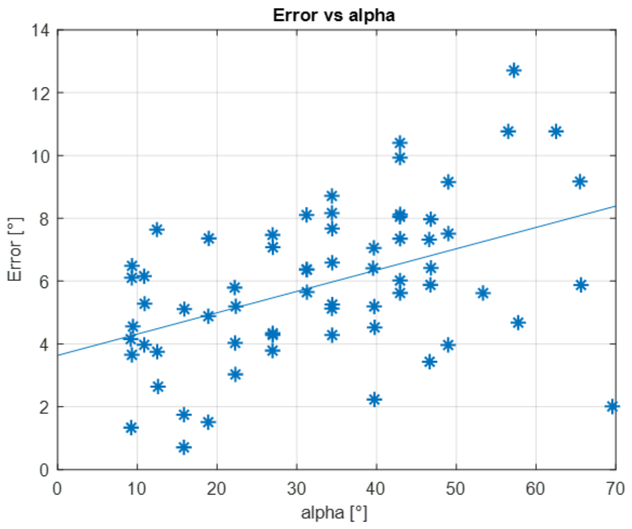


Figure 4.15: Detection error as a function of α

studied was the amount of force applied to the object during each touch. As done for the *approaching detection* method the F/T sensor mounted on the second arm was used. The results are shown in Figure 4.16. The approach detection method led to a first touch detection in correspondence of the red dotted line. At this point the $k_{i,0}$ position is recorded and the finger touches the object other 7 times. Each spike in the force recorded by the F/T sensor 2 corresponds to the moment in which the finger touches the black target. The forces detected are below 1N. What is more the high peaks in the force signal could be caused by the dynamic behavior of the F/T sensor and the real forces applied to the object could be lower. As explained in the following section the experiments performed on the real objects underline the delicacy of the method.

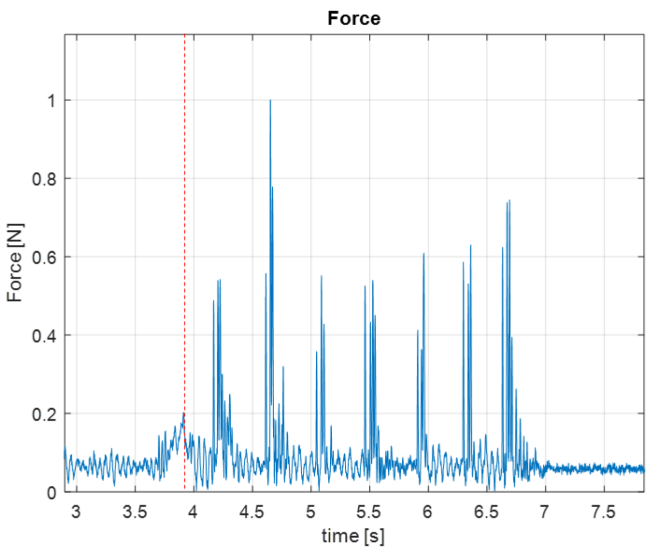


Figure 4.16: Forces applied to the object when the *normal detection* method is applied in sequence to the *approaching detection* method.

4.3.4 Real objects results

After having chosen the best parameters to be utilized, the *normal detection* method was applied to real objects. Detecting the point of contact for each finger leads to the definition of each contact reference frame $\{C_i\}$. The WBF algorithm is then computed to calculate the best forces to apply with each finger. The calculation is done by the thread called *optimizer* introduced in Chapter 3. The data is transmitted through the Redis server and the theory used is described in Chapter 2. In this section, the *normal detection* algorithm applied on real objects is studied. The results of the WBF algorithms are shown. Applications of the two methods previously explained, in a complete grasping framework is studied in the next chapter.

Only three fingers of the AllegroHand were used. The *normal detection* method is applied as soon as all the fingers are in contact with the object. At this point in time, the finger i^{th} touches the object N times and the point of contact C_i is detected. The other fingers are kept in position avoiding any movement of the object during the *normal detection* algorithm performed by the finger i. When the object is not too light the index and the middle fingers could probe the object at the same time. The delicacy of the method ensures that the object is not moved during each touch performed by the fingers. In Figure 4.17, 4.18, 4.19, and 4.20 four different examples are given. In each figure the object is shown, the three normals, approximated by the *normal detection* method, are shown together with the $k_{i,j}$ points collected. The forces computed by the WBF algorithm are shown in red and they are applied on the contact point C_i. After the *normal detection* method is performed by the three fingers and the grasping forces are computed, an operational point is defined in each C_i and an impedance force control (projected in the null space of the pose task defined for the palm) is implemented.

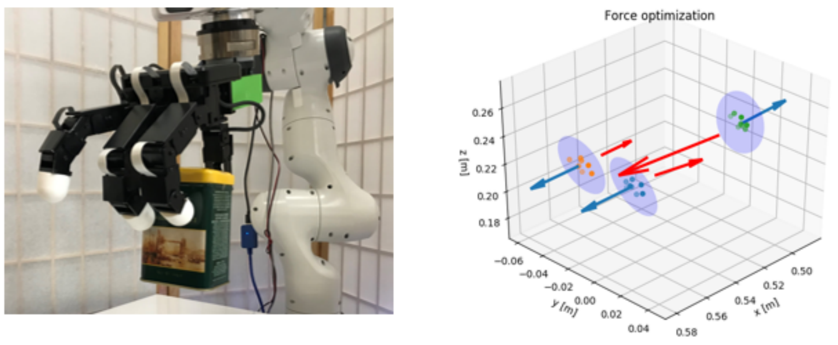


Figure 4.17: *Normal detection* results example 1

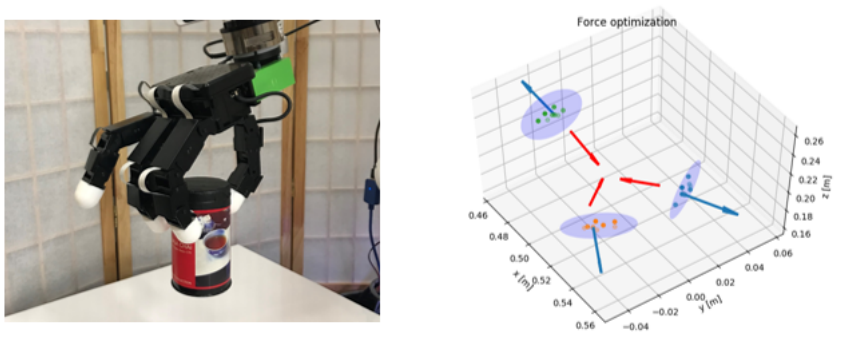


Figure 4.18: *Normal detection* results example 2

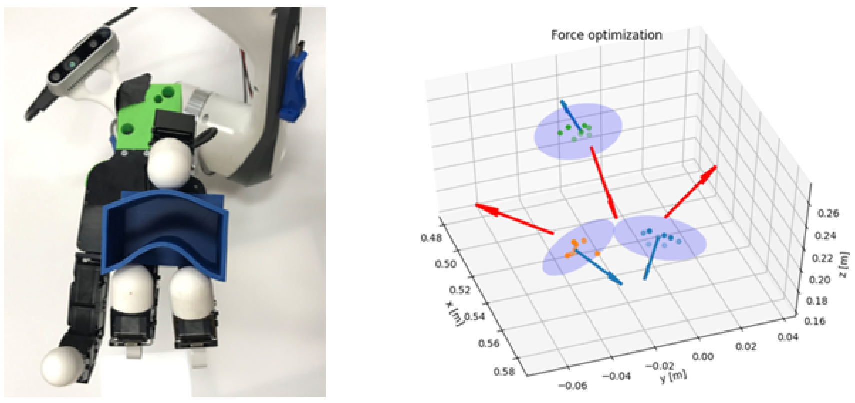


Figure 4.19: *Normal detection* results example 3

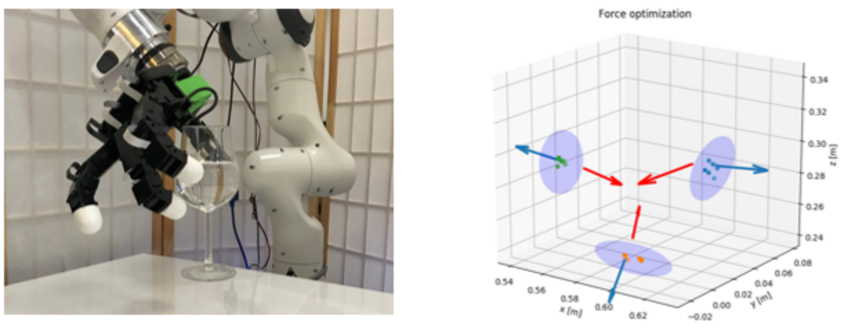


Figure 4.20: *Normal detection* results example 4

Chapter 5

Grasping and Dexterous Manipulation Applications

In this chapter, the two methods described in Chapter 4 are applied to real grasping tasks. The first section of this chapter is dedicated to the description of a grasping policy that takes advantage of the *approching detection* method. The second part of the chapter proposes the utilization of the *normal detection* method to improve an already well-performing grasping algorithms, Unigrasp. The latter utilizes a neural network to detect the grasping points from a point cloud of the object. The hand is brought into contact with those points and the grasping task is improved through the detection of the normals and the application of the forces given by the WBF method. The last part of this chapter shows how the WBF method has real-time capabilities and allows for carrying out an in-hand manipulation task, slightly simplified. Obviously, all the examples discussed are possible thanks to the torque-control characteristics of the hand.

5.1 Initial position adjustment

The *approaching detection* method is the key element of this first application. A set of 8 different glasses (fig. 5.1) are grasped several times. An offset in the real position of the axis of each glass is imposed and the behavior of various grasping algorithms is studied. To bring the finger in contact with the object two different approaches are used:

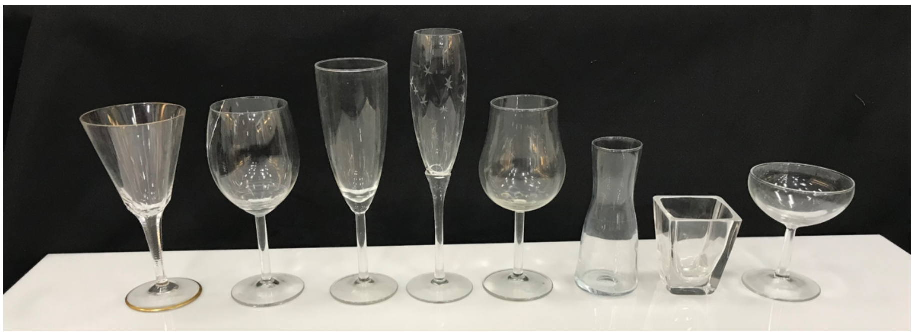


Figure 5.1: Set of glasses utilized for the experiments

- *no initial position adjustment*: the palm of the hand is brought into the initial configuration, without accounting for the presence of the position offset. At this point, all the fingers are brought into contact with the object with the 1-dof force control algorithm described in Chapter 4.

- *initial position adjustment*: the palm of the hand is moved towards the glass, the first touch is detected utilizing the *approaching detection* method. This allows the arm to adjust its position, compensating for the initial offset error. Once the palm position is improved, the fingers are closed as done for the first method.

For both cases, once all the fingers are in contact with the object, three ways to solve the *force distribution* problem are used:

- *Naive finger closing.*
- *Normal detection* method + WBF.
- *Circle grasp* + WBF.

The first method consists in grasping the object in a really simple way, it is easily programmable and it doesn't require knowledge about the object shape. A desired vector of joint angles $\mathbf{q}_d$ is identified where $q_i = q_{i,curr} + \Delta\theta_i$. $q_{i,curr}$ is the initial angle of the i^{th} joint and $\Delta\theta$ is an offset that leads the fingers to close themselves. The presence of the object prevents this closing movement from happening and the fingers end up applying a force to the object, grasping it. The various $\Delta\theta_i$ were quickly euristically optimized. The second method involves the use of the algorithm described in Section 2.3.2, the approximated normals and points of contact are computed and the WBF method is resolved. Often, the real shape of the glass

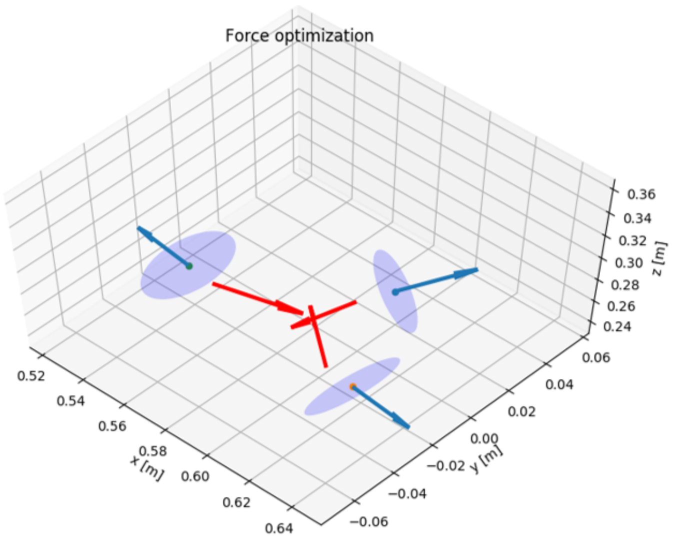


Figure 5.2: Solution of the WBF method using the *circle grasp* normal detection method.

can be approximated to a cylinder whose horizontal section is a circle. Given the 3 points of contact $k_{1,0}$, $k_{2,0}$ and $k_{3,0}$ the center of the circle that goes through those points is identified and the approximated normals are easily computed. Once the normals are known, the evaluation of the point of contacts C_i is straightforward and the WBF is then resolved. The method that relies on this kind of approximation is here called *circle grasp*. A solution for the hand configuration shown in Figure 5.8 is presented in Figure 5.2. The way the initial position offset is imposed is illustrated in Figure 5.3. The offset is computed between the axis of the glass and the axis of the grasp. The latter is parallel to the former but it passes through the geometric center of the three fingertips right before grasping the object.

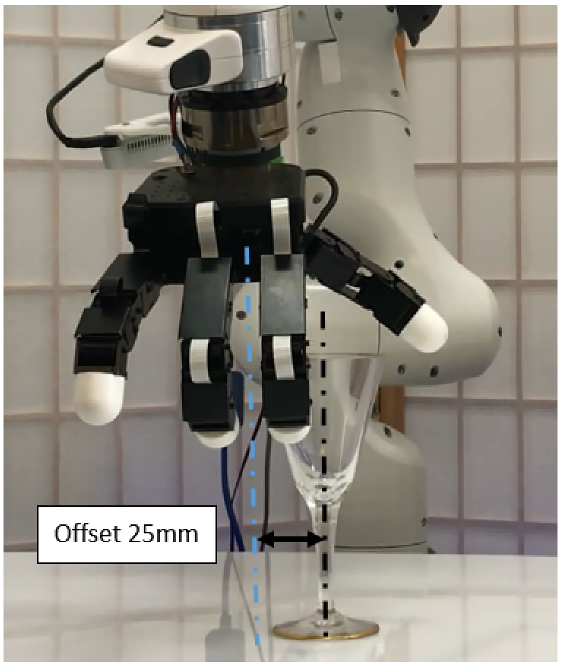


Figure 5.3: Initial offset example

5.1.1 Policy description

To adjust the initial position, the arm/robot system takes advantage of the *approaching detection* method. The steps followed by the policy are shown in Figure 5.4. The index finger is given the subscript 1 while the middle finger is recognized as the 2^{th} finger. The arm is moved with an operational space motion control towards the object and a first touch is detected. When this happens the operational point is switched to the reference frame $\{K_i\}$ of the finger that touched the object. A new rotational motion (with direction dependent on the finger that first touched the object) of the arm around this point is imposed until the second finger comes into contact with the object. After the second touch, the tip of the thumb is brought onto the opposite side of the two fingers that are touching the object. The thumb is then finally moved with a force control algorithm towards the object. To avoid excessive rotation of the palm the algorithm previously described is slightly modified. If the (c) step lasts more than a time threshold t_{th} the hand is brought to the initial position with an offset of few millimeters towards the first finger that touched the object, in the direction perpendicular to the approaching direction. Once the palm found itself in this new initial position the policy is repeated.

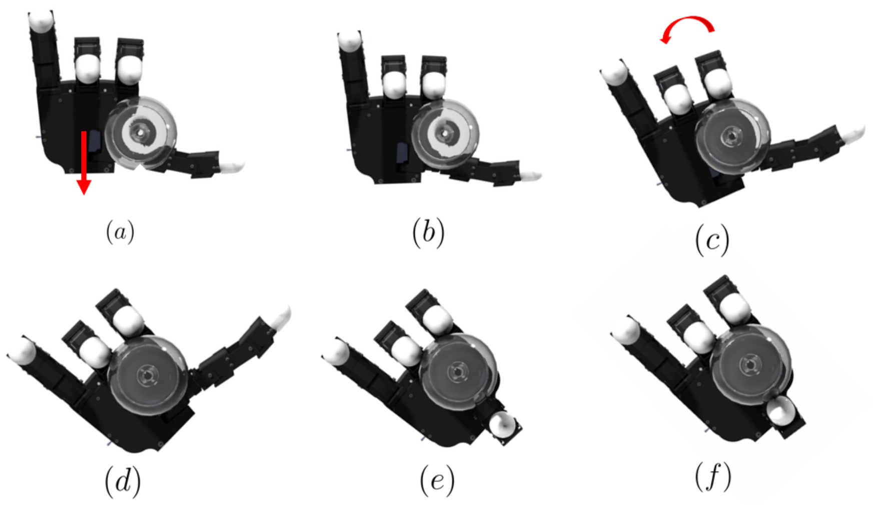


Figure 5.4: *Initial position adjustment* steps

5.1.2 Results

Both methods, adjusting the initial position and not, were tested on each glass. The only information given to the controller about the object was the height of each glass. For each method, the three different ways to resolve the *force distribution* problem were tested using the following offsets: $0mm$, $5mm$, $10mm$, $15mm$, $20mm$ and $25mm$ for a total of 288 tests. In Figure 5.5 the adjusting position policy in

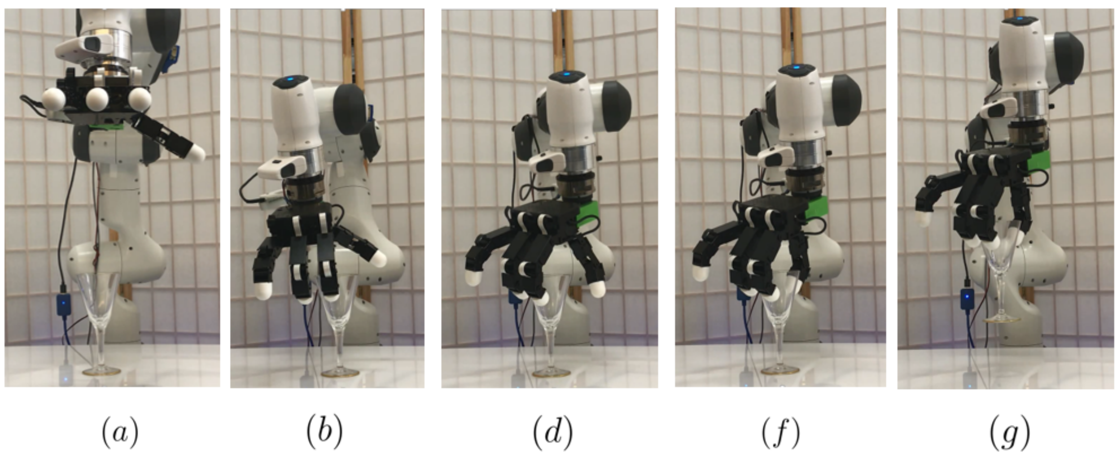


Figure 5.5: *Initial position adjustment* results

action is illustrated. The subscript below each finger corresponds to those used in Figure 5.4 and the lifting step (g) is added. Figure 5.6 shows how the controller behaves when the step c (rotating the hand around $\{K_i\}$) lasts more than t_{th}. As it is illustrated in the pictures the hand is brought back to a modified position (a'), similar to (a) but slightly moved to the side of the first finger that touched the object. This leads to a better initial position of the hand. The policy is then reapplied bringing the hand to lift the object (g). The results of the tests are shown in Figure 5.7. The grasp was considered successful when the glass was robustly grasped. The results underline the effectiveness of the grasping policy. The initial position adjustment allows the hand to bring itself in a position that allows for

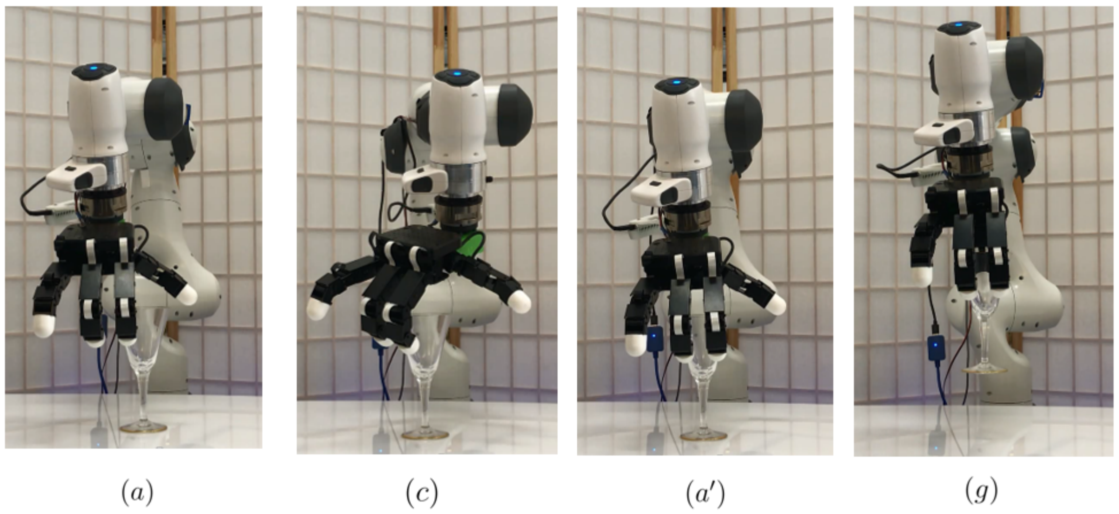


Figure 5.6: Behavior of the *initial position adjustment* policy when (*c*) lasts more than t_{th}

successful rates analogous to a null offset. Given the shape of the glasses, the *circle*

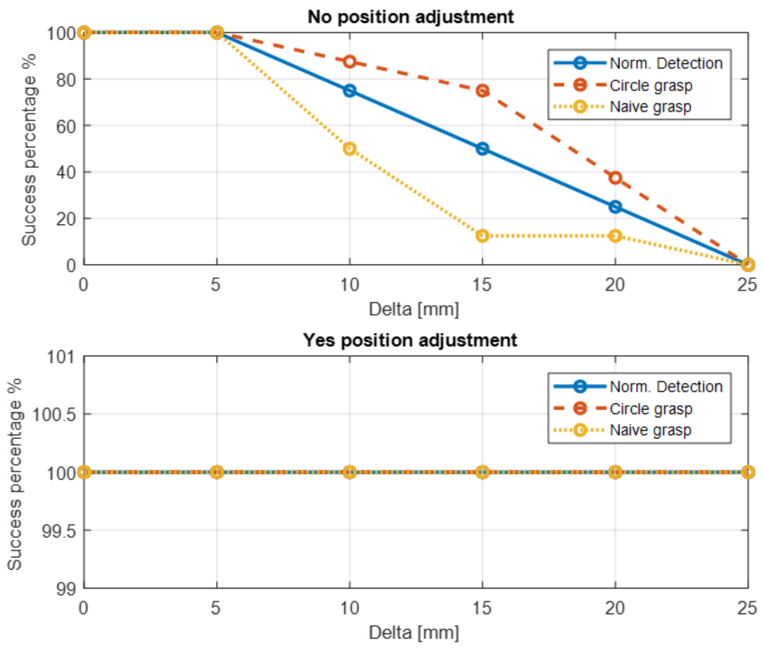


Figure 5.7: Success percentage with and without the position adjustment

grasp performed better than the other methods when no initial position adjustment was applied. The reason why this happens can be understood by looking at Figure 5.8. In the picture, an offset of $15mm$ was applied. It can be seen that once the hand closes the tip of the finger it ends up on the side of the glass. The *circle grasp* method easily compensates for this problem. With the *naive grasp* method the forces applied at the index are inevitably outside of the friction cone. The *normal detection* method fails because the angle α defined in Section 4.3.3 is too high and the error in the normal detection is excessive.

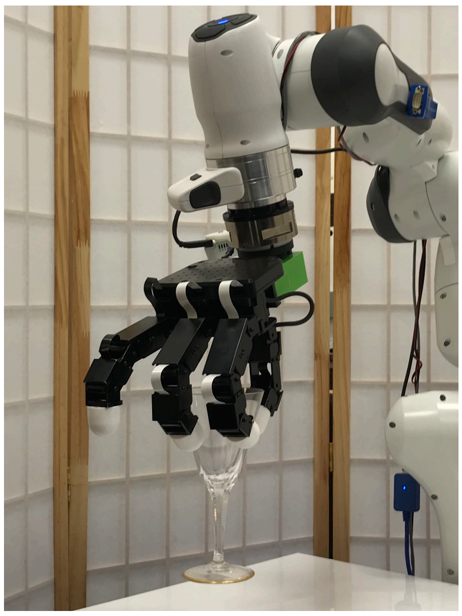


Figure 5.8: Grasp with offset of 15mm when no position adjustment is applied

5.2 Robust grasp

In any grasping task, the grasp planning step plays a key role. The object is detected and a first clue of where and how to grasp the object is obtained. Several algorithms were developed in the last decades grasping research but none of them have yet achieved human-level success rate. The success rate, seen as the simple ability of the hand to lift the object, is not the only skill humans are good at. For some kinds of object, like a glass full of water, the orientation of the object has to be kept constant throughout all the grasping process. What is more, it could be requested that the object remains still as soon as the fingers apply the necessary forces to it. The force distribution resolution plays a really important role in this situation. In this section, *UniGrasp* [58], a well-performing grasp planning data-driven algorithm, is utilized. The grasping task is then improved by applying the methods described in Chapter 3. The objective is showing that only the grasp planning information is not sufficient to meet specific requirements for the grasping task. An interaction with the object and the ”tactile skills”, given by the *normal detection* method, with which the arm/hand system is equipped, can instead help to meet these requirements and improve the grasping task.

5.2.1 Unigrasp and Vision algorithms

A brief description of *UniGrasp* is given. It was implemented in the controller architecture and can be located, looking at the scheme in Figure 3.7, in the *openCV* block. It is written in python and it relies on TensorFlow, a well-known open-source platform for machine learning. A description of the gripper used is given to *UniGrasp* together with a point cloud of the object to be grasped. The neural network on which *UniGrasp* is based, called *Point Set Selection Network* (PSSN), is able to give as output a set of possible grasping points in force closure and reachable by the hand.An illustration of how *UniGrasp* works is given in Figure 5.9. An autoencoder based on PointNet [59] was utilized to learn a low-dimensional latent

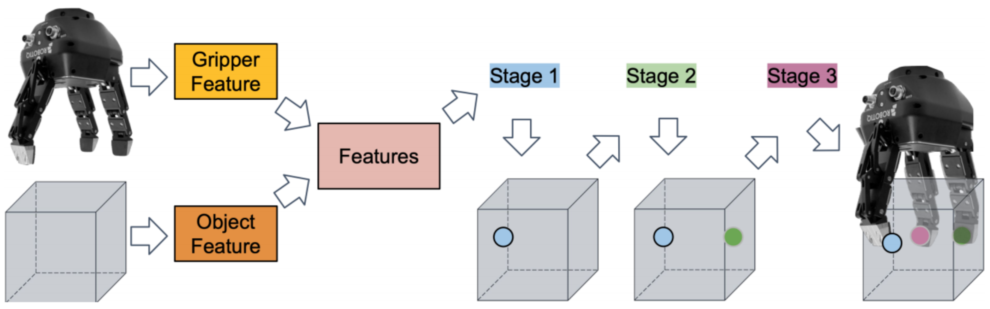


Figure 5.9: *UniGrasp* scheme

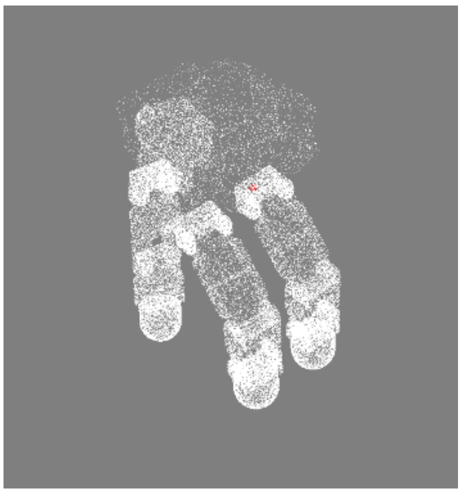
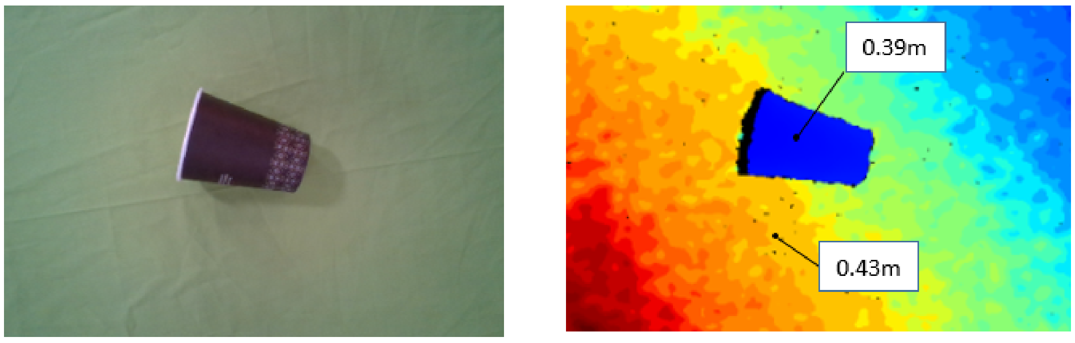


Figure 5.10: Point cloud obtained from AllegroHand URDF

space able to encode the characteristics of the gripper in a specific configuration. The autoencoder was then utilized to extract features from a point cloud of the AllegroHand obtained from its URDF file (fig 5.10).
As far as the object is concerned, a point cloud is obtained by the RGBD camera. PointNet++ [60] is used to extract features and predict the normals of each point in the point cloud. The new extracted feature of the object, together with the features of the gripper are given to the PNNS that proceeds to give a set of n grasping points (in which n is the number of fingers of the gripper). As said before, only 3 fingers



(a) RGB image (b) Depth Image

Figure 5.11: Camera signals

of the AllegroHand were used, the URDF file of the hand was in fact modified to eliminate the finger unused. To detect the point cloud of the object a single image from the RGBD camera is taken. The position p_c and orientation R_c of the camera, with respect to the world frame $\{N\}$, when the picture is taken, is recorded. The points of the cloud fed to *UniGrasp* are still written with respect to the camera frame. Before moving the arm, the coordinates of the three grasping points given by *UniGrasp* are translated to the world frame. In Figure 5.11 an RGB and the depth image taken by the camera are shown. This leads to the point cloud shown in

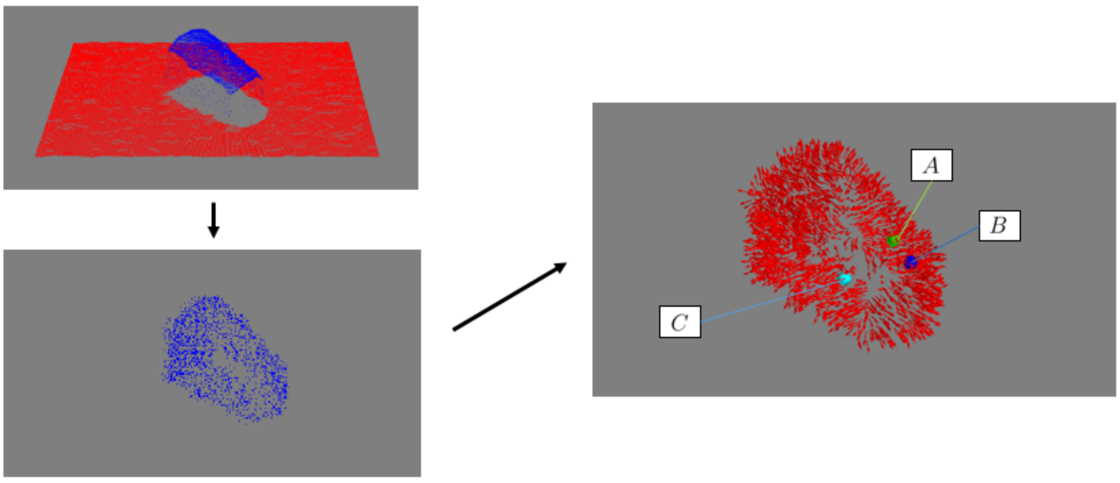


Figure 5.12: Point cloud of the object (right side of the picture) and pre-grasp points selected by *UniGrasp* (right side of the picture)

Figure 5.12a, from which the point cloud of the object is detected (fig. 5.12b). To extract the object from the table only the points above the table are taken. 5.12(c) shows the 3 points selected by *UniGrasp* and the same object grasped in those 3 points is shown in Figure 5.13.

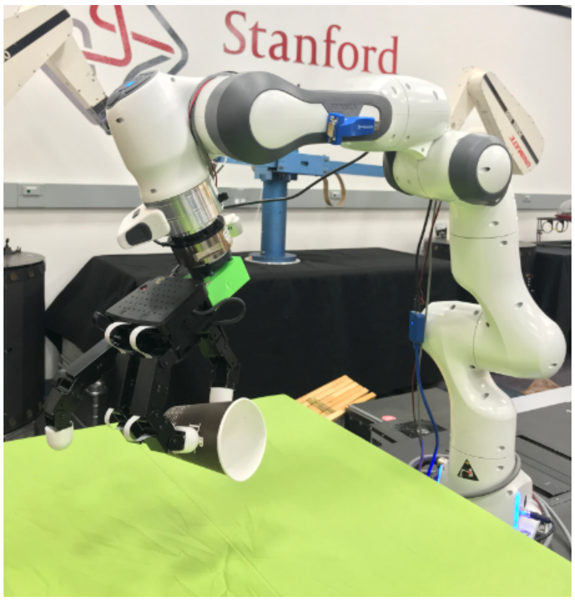


Figure 5.13: Grasp success example

5.2.2 Approaching methodology

Once the three points of contact are elaborated by the controller and their position is translated with respect to the world frame, the controller computes the right torques to bring the three fingers of the hand in contact with the desired points of the object. The procedure followed by the controller is the following: a reference frame $\{H\}$, composed of the three unit vectors $\hat{\mathbf{n}}_H$, $\hat{\mathbf{t}}_H$ and $\hat{\mathbf{o}}_H$, is defined as shown in Figure 5.14 where A, B and C are the points of contact given by *UniGrasp* respectively for the index, the medium finger and the thumb, written w.r.t. the world frame $\{N\}$. The plane *pc* that passes through the three points is detected and its normal is parallel to $\hat{\mathbf{n}}_H$. The vector $\hat{\mathbf{t}}_H$ lies on the plane *pc* and is parallel to the segment CK where K is the middle point of the segment AB. As soon as $\{H\}$ is detected, the arm starts moving in order to bring the finger in contact with the points suggested by *UniGrasp*. A reference frame $\{P_0\}$ is defined fixed to the palm. The desired pose of $\{P_0\}$ is given by the reference frame $\{P_2\}$ that is uniquely defined once $\{H\}$ is computed. The relationship between $\{P_2\}$ and $\{H\}$ is empirically determined to increase the workspace of the hand and ensure that each finger is able to apply the *normal detection* method without reaching joint limits. The arm starts from the

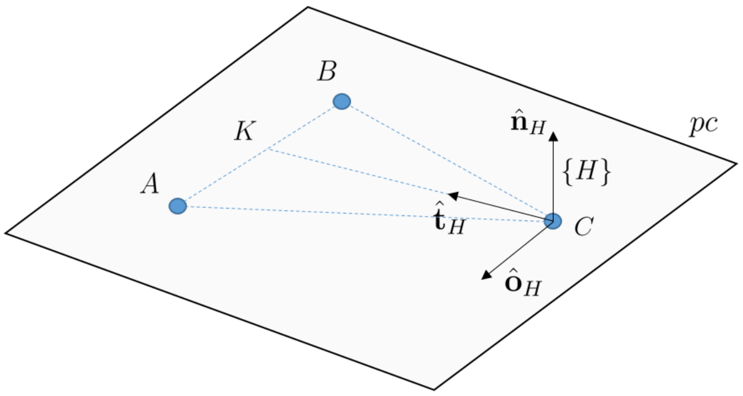


Figure 5.14: $\{H\}$ reference frame definition

position shown in Figure 5.16a. The steps followed by the arm/robotic system to bring the fingers in contact with the object are the following:

- The reference frame $\{P_0\}$ is brought into a first configuration $\{P_1\}$. $\{P_1\}$ has the same orientation of $\{P_2\}$ but its position is located at $k\hat{\mathbf{n}}_H$ from $\{P_2\}$ where k is a constant (around $0.1m$). The hand is still distant from the object as shown in Figure 5.16b and it can approach the object from the $\hat{\mathbf{n}}_H$ direction.

- An inverse kinematic resolution is performed to command the angles of each finger joint and to bring the tip of the fingers into contact with the three points A,B and C. The fingers are closed and contemporary the palm is brought from $\{P_1\}$ to $\{P_2\}$. The third joint of the thumb and the second joint of the other fingers are kept at $\Delta\theta$ from the desired angular position (fig. 5.16c). This helps to avoid touching the object during the hand movement from $\{P_1\}$ to $\{P_2\}$.

- Given $\Delta\theta$, the palm is in the right position but the fingers are not touching the object yet. A 1-dof force control method, as described in Section 4.3.2, is utilized to bring the fingers in contact with the object. It is worth noting that the $\Delta\theta$ are applied only to the active joints. The final points of contact are therefore coincident with those requested by *UniGrasp*, without considering uncertainties (fig. 5.16c)

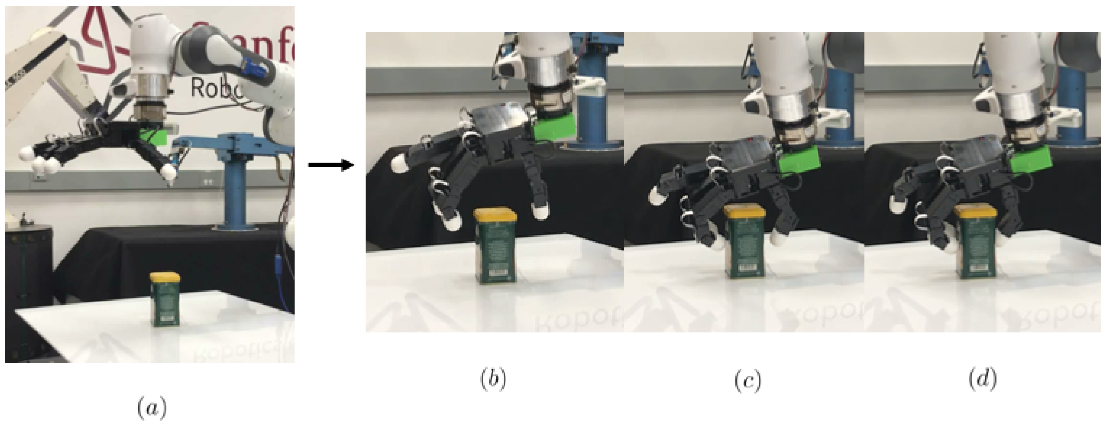


Figure 5.15: Object approaching steps

5.2.3 Experiment description

After the fingers are brought into contact with the object the *normal detection* method is applied, the approximated normals are used to resolve the *force distribution* problem and the object is lifted. As it was done in Section 5.1 other methodologies were used to grasp the object to have a good comparison of the methods used. In total the following methods were used:

- *normal detection* method (NOG).
- circle normal method (CIG).
- naive grasp (NAG).
- naive grasp without first touch (NNG).

The first three methods were already described in the previous section. The latter is equivalent to the naive grasp method but the fingers are closed from the configuration shown in fig. 5.16c. The choice of proposing also this kind of grasp was done to understand the importance of a first delicate contact with all the finger, possible thank to the torque-control capabilities of the hand. To test the methods the objects in 5.16 were utilized. Each object was positioned in four different configurations. The *UniGrasp* algorithm was run only once for each configuration. Each method was then tested on those three pre-grasp points to have a good comparison.

5.2.4 Results

Several tests were done on the objects in Figure 5.16. Some of the successful grasps are shown in Figure 5.17. The result of each test was firstly categorize as:

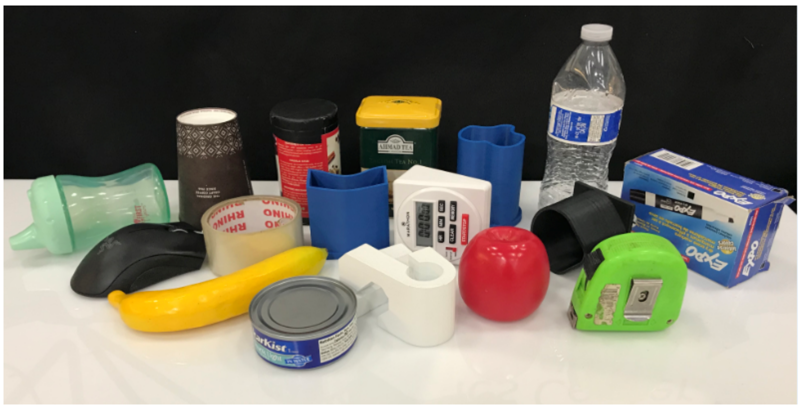


Figure 5.16: Set of objects utilized for the tests

- Successful grasp: the hand was able to lift the object and the grasp was stable even when little external forces were applied to the object.
- Failing grasp: the hand wasn't able to lift the object or the grasp was evidently unstable.

As far as these two categories are concerned, the results are shown in the left graph of Figure 5.18. The percentages of successful grasps over all the tests performed are shown. It is worth noting that the *NOG* and the *NAG* performs the same way with

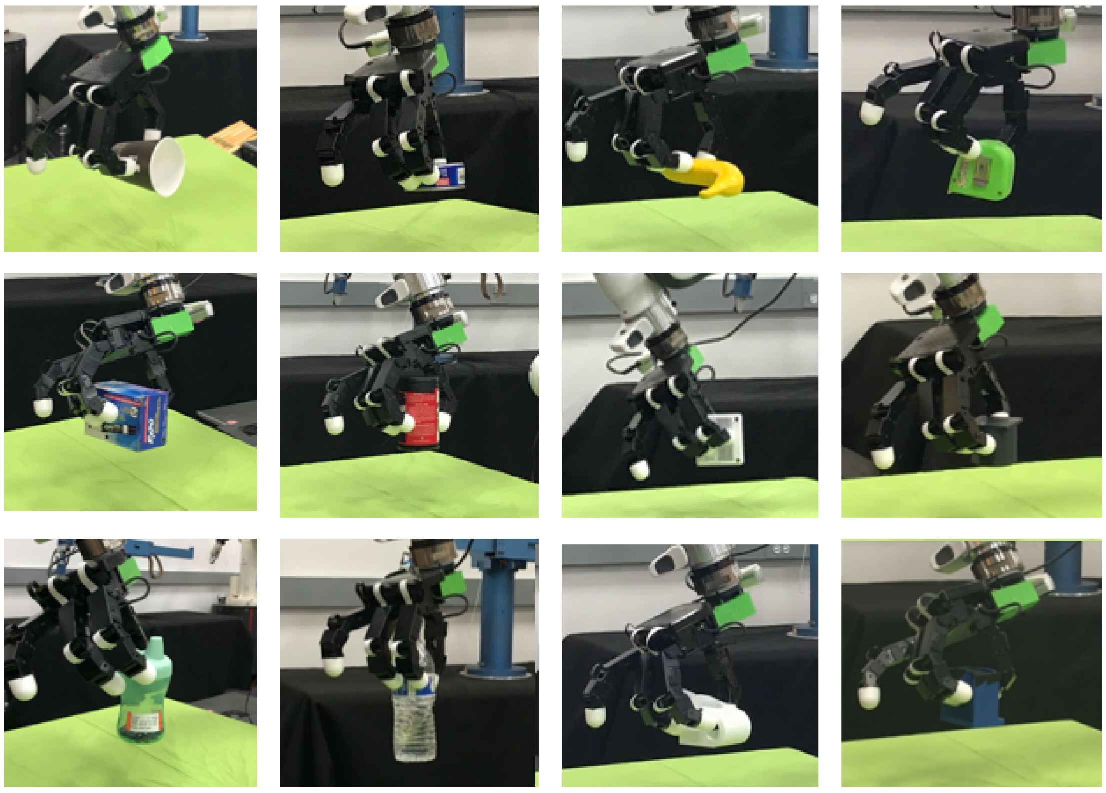


Figure 5.17: Successful grasps examples

a percentage of around 85%. This is close to the results obtained in [58]. For this reason, if the objective of the grasping task is that of lifting the object without other constraints, the NAG is recommended given the simplicity of the controller and the velocity of the grasping task. The successful grasps were further categorized in:

- *Complete success* (S): the object maintains his orientation during the lifting process, the fingers don't change position after the grasping forces are applied.
- *Finger moved* (MF): the orientation of the object is kept constant but the finger slides on the object after the grasping forces are applied.
- *Object moved* (MO): in this case the orientation or position of the object changes as soon as the grasping forces are applied.

A *complete success* can be requested when the object has to be manipulated after being lifted or when the task requires a delicate lifting of the object itself. For example, in analogy with Section 5.1, a glass full of liquid has to be lifted in such a way to avoid pouring the content outside. The results of the categorization of the successful grasps are shown on the right side of Figure 5.18. It is evident how the *normal detection* method allows a *complete success* of a high majority of grasps.

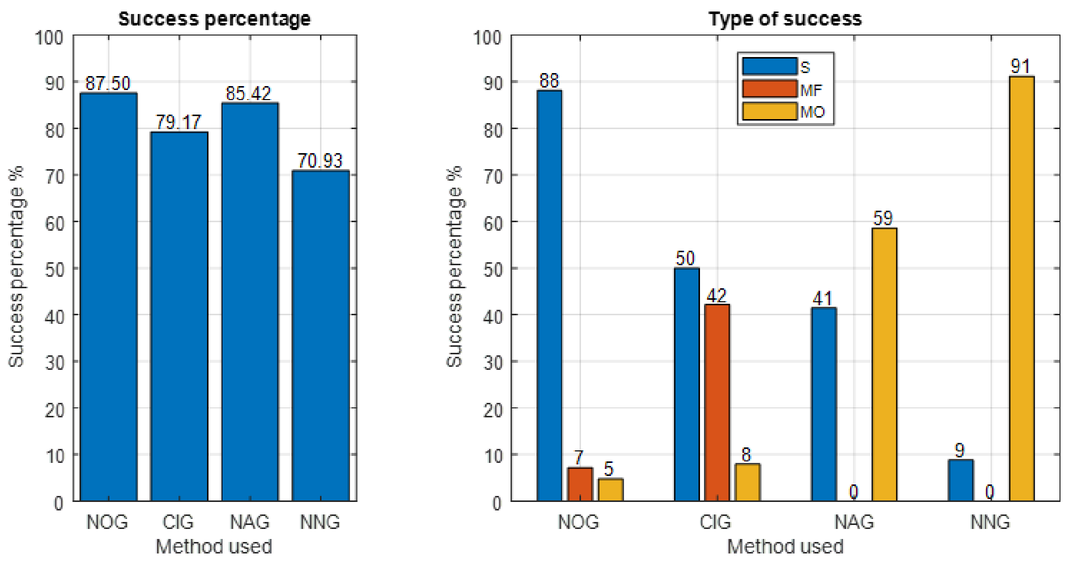


Figure 5.18: Tests results. Successful rate on the left, type of successful grasp on the right

5.3 In-hand Manipulation

Once the object is lifted, there could be the need of manipulating it using just the fingers. In this section, it is shown that this can be achieved using an operational space formulation. The WBF method allows the application of a known wrench $\mathbf{F}$ to the object. To control the pose of the object, the reference frames $\{C_i\}$ have to be updated as the tip of the finger rolls on the object (supposing that no slippage occurs). To update the reference frames, the finger and object surface parameters have to be known. In the following, an operational space control is applied to the reference frame $\{H\}$ and the desired pose for this reference frame is given. The

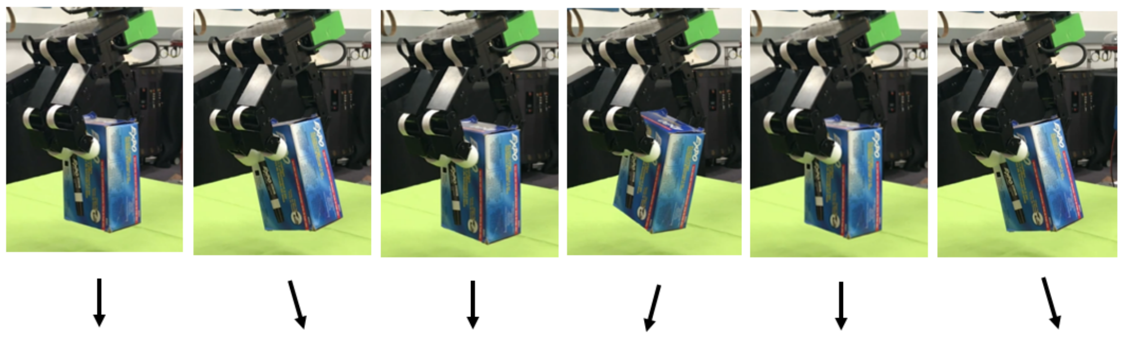


Figure 5.19: In-hand manipulation

fingers are going to apply forces to the object to control the configuration of $\{H\}$. Although the actual control is on $\{H\}$ and not on the object, it is illustrated that this control can be executed in real-time. Therefore, a proper control of the object is possible once a rolling finger model is incorporated.
A desired pose of the $\{H\}$ reference frame is imposed. The normal of the objects w.r.t the world frame are supposed to be constant and the point of contact is updated accordingly. In spite of the fact that during the real movement the contact normals

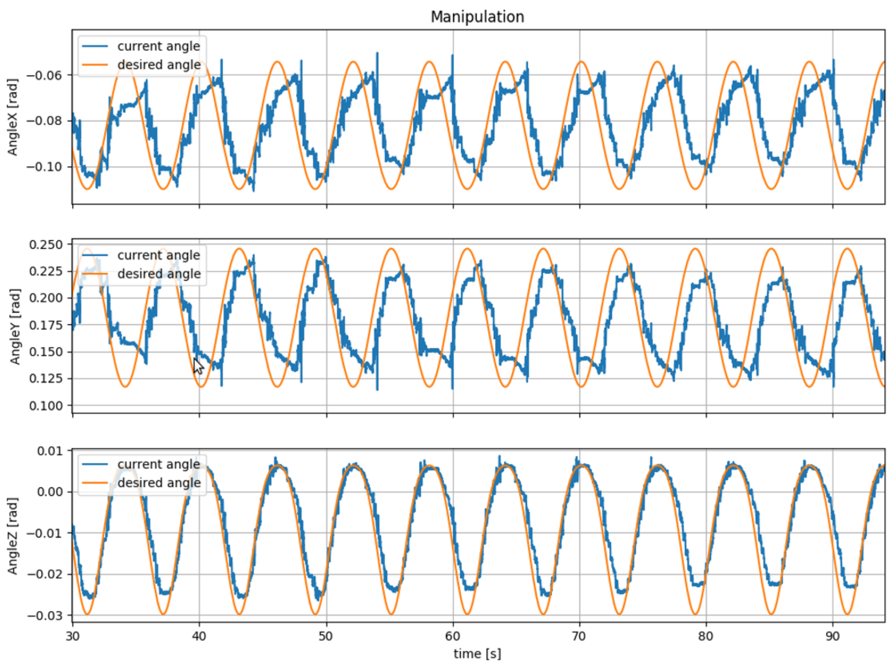


Figure 5.20: In-hand manipulation results

change their orientation, the desired orientation of the $\{H\}$ is imposed so that the error introduced is small. A sequence of the hand rotating the object is shown in Figure 5.19 while the results in terms of angular orientations of $\{H\}$ compared to the desired ones are shown in Figure 5.20. The orientation of the frame $\{H\}$ is really close to the desired one. This demonstrates the real-time ability of the controller. It is reminded that the results are not coincident with a direct control of the object. However, these results are promising from the moment in which a rolling model of the fingers onto the object is introduced.

Chapter 6

Conclusion

The objective of this thesis was to underline the importance of an active interaction between the robotic hand and the objects. As human beings, we strongly depend on the information obtained from contacts between the fingers and the object surfaces not only when our vision abilities are impaired but in every grasping task we perform. The limitations in terms of hardware, such as the absence of tactile sensors and force sensors, has lead to the development of the methodologies explored in Chapter 4. Although these methods do not expect to substitute the huge amount of data and information provided by the use of appropriate sensors, it was shown that a simple interaction between the fingers and the object can help to perform successful and robust grasps. This was further underlined in Chapter 5, where a policy that relies on the *approaching detection* method was presented and the successful results were illustrated. The introduction of the *normal detection* method in a grasping task based on the data-driven grasp planning algorithm *UniGrasp* showed important improvements in terms of the way the hand grasped the object. The *force optimization* problem was solved efficiently leading to a delicate and precise grasp where the orientation of the object was preserved. In future applications, the methods explored in this thesis could be run on multi-finger hands equipped with further sensors leading to more successful and robust grasps. The interaction between the fingers and the hand achieved thanks to the compliance of the AllegroHand, turned out to be extremely useful and potentially even more effective, with regards to grasping applications. It was in fact shown that it is possible to touch the objects several times without disturbing it, taking advantage of the compliant characteristics of the multi-finger hand used. The same algorithms applied on a higher-performance hand in terms of hardware characteristics could lead to even more interesting results and applications.

The results obtained confirmed the fact that the amount of data collected after the hand has touched the object plays a key role in solving the grasping problem. For this reason, in the development of grasping control framework, the grasp planning step should not be seen as sufficient for achieving human-like skills in robotic systems and more algorithms that take advantage of contacts between the hand the object should be studied and developed.

List of Figures

Bibliography

[1] Dov Katz, Jacqueline Kenney, and Oliver Brock. "How Can Robots Succeed in Unstructured Environments?" In: (Dec. 2010).

[2] M. Jorda, E. G. Herrero, and O. Khatib. "Contact-Driven Posture Behavior for Safe and Interactive Robot Operation". In: *2019 International Conference on Robotics and Automation (ICRA)*. May 2019, pp. 9243–9249. DOI: `10.1109/ICRA.2019.8793691`.

[3] S. Haddadin, A. De Luca, and A. Albu-Schäffer. "Robot Collisions: A Survey on Detection, Isolation, and Identification". In: *IEEE Transactions on Robotics* 33.6 (Dec. 2017), pp. 1292–1312. ISSN: 1941-0468. DOI: `10.1109/TRO.2017.2723903`.

[4] L. Manuelli and R. Tedrake. "Localizing external contact using proprioceptive sensors: The Contact Particle Filter". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2016, pp. 5062–5069. DOI: `10.1109/IROS.2016.7759743`.

[5] Antonio Bicchi, Michael A. Peshkin, and J. Edward Colgate. "Safety for Physical Human–Robot Interaction". In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1335–1348.

[6] J. Edward, Witaya Wannasuphoprasit, and Michael Peshkin. "Cobots: Robots For Collaboration With Human Operators". In: (Mar. 1999).

[7] Oussama Khatib et al. "Torque-position transformer for task control of position controlled robots". In: June 2008, pp. 1729–1734. DOI: `10.1109/ROBOT.2008.4543450`.

[8] G. Hirzinger et al. "DLR's torque-controlled light weight robot III-are we reaching the technological limits now?" In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. Vol. 2. May 2002, 1710–1716 vol.2. DOI: `10.1109/ROBOT.2002.1014788`.

[9] R. Bischoff et al. "The KUKA-DLR Lightweight Robot arm - a new reference platform for robotics research and manufacturing". In: *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*. June 2010, pp. 1–8.

[10] *Franka Emika website*. URL: `https://www.franka.de/`.

[11] C. Piazza et al. "A Century of Robotic Hands". In: *Annual Review of Control, Robotics, and Autonomous Systems* 2.1 (2019), pp. 1–32.

[12] L. Biagiotti et al. *How Far Is the Human Hand? A Review on Anthropomorphic Robotic End-effectors*. 2003.

[13] A. Bicchi. “Hands for dexterous manipulation and robust grasping: a difficult road toward simplicity”. In: *IEEE Transactions on Robotics and Automation* 16.6 (2000), pp. 652–662.

[14] Antonio Bicchi and Vijay Kumar. “Robotic Grasping and Contact: A Review”. In: *Proceedings - IEEE International Conference on Robotics and Automation* 1 (Mar. 2000).

[15] J.R. Napier. “The prehensile movements of the human hand”. In: *Surger* 38.4 (1956), pp. 902–913.

[16] M. R. Cutkosky. “On grasp choice, grasp models, and the design of hands for manufacturing tasks”. In: *IEEE Transactions on Robotics and Automation* 5.3 (June 1989), pp. 269–279. ISSN: 2374-958X. DOI: 10.1109/70.34763.

[17] I. M. Bullock et al. “Grasp Frequency and Usage in Daily Household and Machine Shop Tasks”. In: *IEEE Transactions on Haptics* 6.3 (July 2013), pp. 296–308. ISSN: 2334-0134. DOI: 10.1109/TOH.2013.6.

[18] Thomas Feix, Roland Pawlik, and Heinz-Bodo Schmiedmayer. “The generation of a comprehensive grasp taxonomy”. In: (Jan. 2009).

[19] A. Sahbani, S. El-Khoury, and P. Bidaud. “An overview of 3D object grasp synthesis algorithms”. In: *Robotics and Autonomous Systems* 60.3 (Mar. 2012), pp. 326–336. DOI: 10.1016/j.robot.2011.07.016. URL: https://doi.org/10.1016%2Fj.robot.2011.07.016.

[20] Máximo A. Roa and Raúl Suárez. “Grasp quality measures: review and performance”. In: *Autonomous Robots* 38.1 (Jan. 2015), pp. 65–88. ISSN: 1573-7527. DOI: 10.1007/s10514-014-9402-3. URL: https://doi.org/10.1007/s10514-014-9402-3.

[21] Randy C. Brost. *Planning Robot Grasping Motions in the Presence of Uncertainty.* Tech. rep. CMU-RI-TR-85-12. Pittsburgh, PA: Carnegie Mellon University, July 1985.

[22] Yu Zheng and Wen-Han Qian. “Coping with the Grasping Uncertainties in Force-closure Analysis”. In: *The International Journal of Robotics Research* 24.4 (2005), pp. 311–327. DOI: 10.1177/0278364905049469.

[23] R. Balasubramanian et al. “Physical Human Interactive Guidance: Identifying Grasping Principles From Human-Planned Grasps”. In: *IEEE Transactions on Robotics* 28.4 (Aug. 2012), pp. 899–910.

[24] J. Weisz and P. K. Allen. “Pose error robust grasping from contact wrench space metrics”. In: *2012 IEEE International Conference on Robotics and Automation.* May 2012, pp. 557–562.

[25] A.T. Miller and P.K. Allen. “Graspit! A versatile simulator for robotic grasping”. In: *Robotics Automation Magazine, IEEE* 11.4 (Dec. 2004), pp. 110–122. ISSN: 1070-9932. DOI: 10.1109/MRA.2004.1371616.

[26] Jeannette Bohg et al. “Data-Driven Grasp Synthesis - A Survey”. In: *CoRR* abs/1309.2660 (2013). arXiv: 1309.2660. URL: http://arxiv.org/abs/1309.2660.

[27] Roland Johansson and John Flanagan. “Coding and use of tactile signals from the fingertips in object manipulation tasks”. In: *Nature reviews. Neuroscience* 10 (May 2009), pp. 345–59. DOI: 10.1038/nrn2621.

[28] Joseph Romano et al. “Human-Inspired Robotic Grasp Control With Tactile Sensing”. In: *Robotics, IEEE Transactions on* 27 (Jan. 2012), pp. 1067–1079. DOI: 10.1109/TRO.2011.2162271.

[29] Moslem Kazemi et al. “Human-inspired force compliant grasping primitives”. In: *Autonomous Robots* 37.2 (Aug. 2014), pp. 209–225.

[30] R.D. Howe et al. “Grasping, manipulation, and control with tactile sensing”. In: June 1990, 1258–1263 vol.2. DOI: 10.1109/ROBOT.1990.126171.

[31] Johan Tegin and Jan Wikander. “Tactile sensing in intelligent robotic manipulation—A review”. In: *Industrial Robot: An International Journal* 32 (Feb. 2005). DOI: 10.1108/01439910510573318.

[32] Mark R. Cutkosky, Robert D. Howe, and William R. Provancher. “Force and Tactile Sensors”. In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 455–476.

[33] Lorenzo Natale and Eduardo Torres-jara. “A sensitive approach to grasping”. In: *In Proceedings of the sixth international workshop on epigenetic robotics*. Citeseer, 2006, pp. 87–94.

[34] Javier Felip and Antonio Morales. “Robust sensor-based grasp primitive for a three-finger robot hand”. In: *Intelligent Robots and Systems. IROS 2009. IEEE/RSJ International Conference on*. 2009. DOI: 10.1109/IROS.2009.5354760.

[35] K. Hsiao et al. “Contact-reactive grasping of objects with partial shape information”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2010, pp. 1228–1235. DOI: 10.1109/IROS.2010.5649494.

[36] P. Pastor et al. “Online movement adaptation based on previous sensor experiences.” English (US). In: *IEEE International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2011, pp. 365–371.

[37] Francois R. Hogan et al. *Tactile Regrasp: Grasp Adjustments via Simulated Tactile Transformations*. 2018. arXiv: 1803.01940 [cs.RO].

[38] Samuel Buss. “Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods”. In: *IEEE Transactions in Robotics and Automation* 17 (May 2004).

[39] E. Kreund. “The structure of decoupled non-linear systems”. In: *International Journal of Control* 21 (1975), pp. 443–450.

[40] Bruno Siciliano et al. *Robotics: Modelling, Planning and Control*. Jan. 2011. DOI: 10.1007/978-1-84628-642-1.

[41] Bruno Siciliano and Luigi Villani. *Robot Force Control*. 1st. USA: Kluwer Academic Publishers, 2000. ISBN: 0792377338.

[42] Tsuneo Yoshikawa. “Force control of robot manipulators”. In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)* 1 (2000), 220–226 vol.1.

[43] N. Hogan. “Impedance Control: An Approach to Manipulation”. In: *1984 American Control Conference*. June 1984, pp. 304–313. DOI: 10.23919/ACC.1984.4788393.

[44] M. T. Mason. “Compliance and Force Control for Computer Controlled Manipulators”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 11.6 (June 1981), pp. 418–432. ISSN: 2168-2909. DOI: 10.1109/TSMC.1981.4308708.

[45] O. Khatib. “A unified approach for motion and force control of robot manipulators: The operational space formulation”. In: *IEEE Journal on Robotics and Automation* 3.1 (Feb. 1987), pp. 43–53. ISSN: 2374-8710. DOI: 10.1109/JRA.1987.1087068.

[46] Richard M. Murray, S. Shankar Sastry, and Li Zexiang. *A Mathematical Introduction to Robotic Manipulation*. 1st. USA: CRC Press, Inc., 1994.

[47] Domenico Prattichizzo and Jeffrey C. Trinkle. *Grasping*. Ed. by Bruno Siciliano and Oussama Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 671–700.

[48] Jeffrey Kerr and Bernard Roth. “Analysis of Multifingered Hands”. In: *The International Journal of Robotics Research* 4.4 (1986), pp. 3–17.

[49] M. Buss, H. Hashimoto, and J. B. Moore. “Dextrous hand grasping force optimization”. In: *IEEE Transactions on Robotics and Automation* 12.3 (June 1996), pp. 406–418.

[50] Martin Buss, Leonid Faybusovich, and J.B. Moore. “Recursive algorithms for real-time grasping force optimization”. In: May 1997, 682–687 vol.1. ISBN: 0-7803-3612-7. DOI: 10.1109/ROBOT.1997.620115.

[51] M. Buss, L. Faybusovich, and J.B. Moore. “Dikin-type algorithms for dextrous grasping force optimization”. In: *International Journal of Robotics Research* 17.8 (Aug. 1998).

[52] Youshen Xia, Jun Wang, and Lo-Ming Fok. “Grasping-force optimization for multifingered robotic hands using a recurrent neural network”. In: *IEEE Transactions on Robotics and Automation* 20.3 (June 2004), pp. 549–554. ISSN: 2374-958X. DOI: 10.1109/TRA.2004.824946.

[53] Per Henrik Borgstrom et al. “Weighted barrier functions for computation of force distributions with friction cone constraints”. In: *2010 IEEE International Conference on Robotics and Automation*. May 2010, pp. 785–792.

[54] D. Williams and O. Khatib. “The virtual linkage: a model for internal forces in multi-grasp manipulation”. In: *[1993] Proceedings IEEE International Conference on Robotics and Automation*. May 1993, 1025–1030 vol.1. DOI: 10.1109/ROBOT.1993.292110.

[55] *Franka Control Interface Documentation*. URL: https://frankaemika.github.io/docs/.

[56] *AllegroHand wiki*. URL: `http://wiki.wonikrobotics.com/AllegroHandWiki/index.php/Allegro_Hand`.

[57] *Redis*. URL: `https://redis.io/`.

[58] Lin Shao et al. *UniGrasp: Learning a Unified Model to Grasp with N-Fingered Robotic Hands*. 2019. arXiv: `1910.10900 [cs.RO]`.

[59] Charles R. Qi et al. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. 2016. arXiv: `1612.00593 [cs.CV]`.

[60] Charles R. Qi et al. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. 2017. arXiv: `1706.02413 [cs.CV]`.