Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Curriculum in Ingegneria Meccanica

New Programming Techniques to Enable and Simplify the Integration of Collaborative Robots in Industrial Environment

Ph.D. Dissertation of:
Daniele Massa

Advisor:
Prof. Massimo Callegari

Coadvisor:
Ph.D. Cristina Cristalli

Curriculum Supervisor:
Prof. Ferruccio Mandorli

XV edition - new series

Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Curriculum in Ingegneria Meccanica

New Programming Techniques to Enable and Simplify the Integration of Collaborative Robots in Industrial Environment

Ph.D. Dissertation of:
Daniele Massa

Advisor:
Prof. Massimo Callegari

Coadvisor:
Ph.D. Cristina Cristalli

Curriculum Supervisor:
Prof. Ferruccio Mandorli

XV edition - new series

Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Facoltà di Ingegneria
Via Brecce Bianche – 60131 Ancona (AN), Italy

Acknowledgments

Thanks to Professor Massimo Callegari and to the group of machine mechanics of DIISM, for supporting me and helping me developing my Ph.D. project in those 3 years. Thanks to the Loccioni Group which gave me the opportunity of taking this academic path, and a special thanks to Cristina Cristalli and my colleagues of Research for Innovation, which supported me in developing and realizing the project. A special thanks also to Arne Roennau and the robotics group of FZI for helping me with the final part of my project. Finally a big thank also to my family that supported me the whole time.

Ancona, Gennaio 2016

Daniele Massa

Abstract

Historically, robots have focused on high volume and highly repetitive tasks, exploiting their precision for those tasks that were boring and/or difficult for humans. But there is a big number of repetitive tasks which is still done by operators, which standard industrial robots are not able to do, due to their lack of flexibility and adaptation to new and different contexts. The robot industry is now changing through the introduction of collaborative robots, which are more flexible respect to traditional industrial robots and can work alongside with human operators. This complementarity between industrial robots and collaborative robots opens many new possibilities both in industrial context and in new fields like service robotics. Moreover, with the coming of Industry 4.0 and Cyber-Physical Systems, the need of having flexible and reconfigurable systems is increasing. To increase the flexibility of industrial robots (collaborative or not) it is needed to develop open source software for robotics programming, to enhance the flexibility and interchangeability of robots. The contribution of this Ph.D. thesis is an analysis of the current state of collaborative robotics, including safety aspects. In addition, this work provides tools that simplify the task of robot programming, making it time-saving and user-friendly, so that no particular knowledge in robotics is required to achieve that. In particular the work is focused on finding sound solutions that can fit the needs of industrial contest. Thus, this work proposes innovative methods, validated through experimental results and proposing realistic use-cases, to improve and simplify the robot programming, so to make robots more flexible and well-suited to new needs of Industry 4.0.

Contents

List of Figures

Chapter 1

Introduction

In the recent years industrial robotics is having a new spread after the financial crisis of 2009. In 2015 248,000 units were sold worldwide, with a rise of 12 percent respect to the previous year; according to the International Federation of Robotics (IFR) in 2018, about 2.3 millions units will be deployed in factory floors [1]. Nowadays robots used for industrial applications still represents about the 90 percent of the overall robotics market, while service robotics accounts for the remaining 10 percent. Speaking of sales, in 2014 the worldwide market value for robots increased to a new peak of 10.7 billions US$, which gets to 32 billions US$ considering the cost of software, peripherals and systems engineering. This highlights how the market is fertile both for robot constructors and for system integrators.

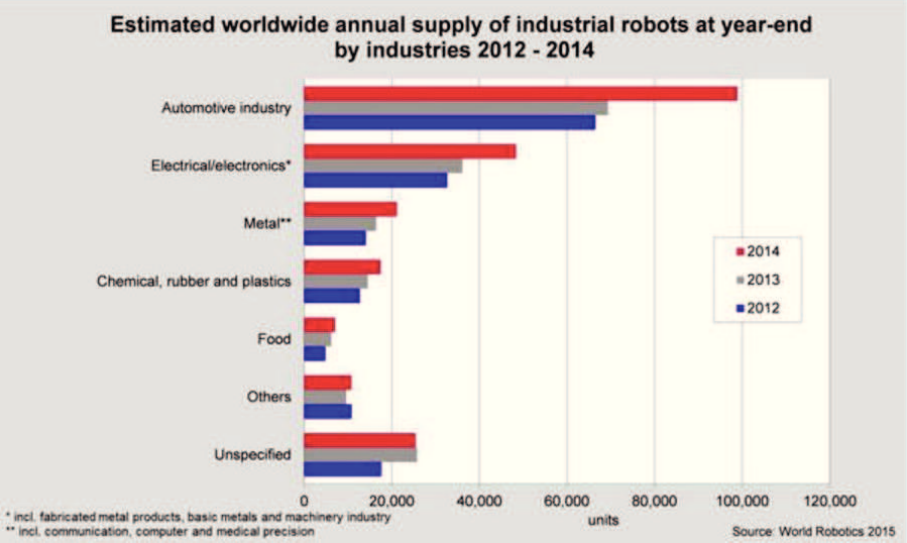


Figure 1.1: Industrial robots market in the last years.

The modern robotics industry has its roots with automotive customers. Robots were first used by General Motors in the 1960s and are now ubiqui-

tous in auto assembly plants globally – penetration of robots in automotive welding for example is now greater than 90 percent. The other big market for industrial robotics is the electrical/electronics industry, which together with the automotive industry constitutes the 65 percent of the overall industrial market for robotics; Figure 1.1 shows the market distribution for industrial robotics in the last years.

Historically, robots have focused on high volume and highly repetitive tasks, exploiting their precision for those tasks that were boring and/or difficult for humans. But there is a big number of repetitive tasks which is still done by operators, which standard industrial robots are not able to do, due to their lack of flexibility and adaptation to new and different contexts. The robot industry is now changing through the introduction of collaborative robots, which are more flexible respect to traditional industrial robots and can work alongside with human operators. This complementarity between industrial robots and collaborative robots, opens a lot of new possibilities both in industrial context and in new fields like service robotics.

Currently collaborative robotics represents just a 5 percent of the overall robotics market, but it is expected to grow tenfold until 2020 (considering both industrial and non-industrial applications), becoming a fundamental actor of the new industrial revolution, Industry 4.0.

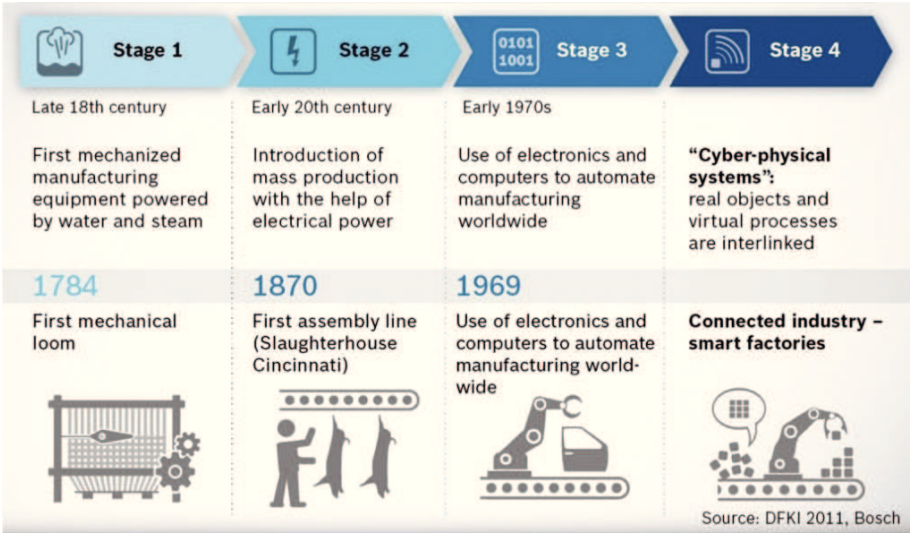


Figure 1.2: Robotics and automation evolution during industrial revolutions.

The Industry 4.0 concept has born thanks to the rise of autonomous robots, contemporary automation, cyber-physical systems, the internet of things, the internet of services, and so on.

Cyber-Physical Systems (CPSs) are the new generation of engineered systems, born from the union of physical devices with networking computing. CPSs are systems featuring a tight combination and integration of computational and physical elements [2]: they are based on computational elements hosted in stand-alone devices, thus extending the concept of embedded systems (see Fig. 1.3). This new kind of systems is at the base of the Industry 4.0 concept [3] and is expected to bring changes not only in the industrial sphere, but also in the society and our everyday life.

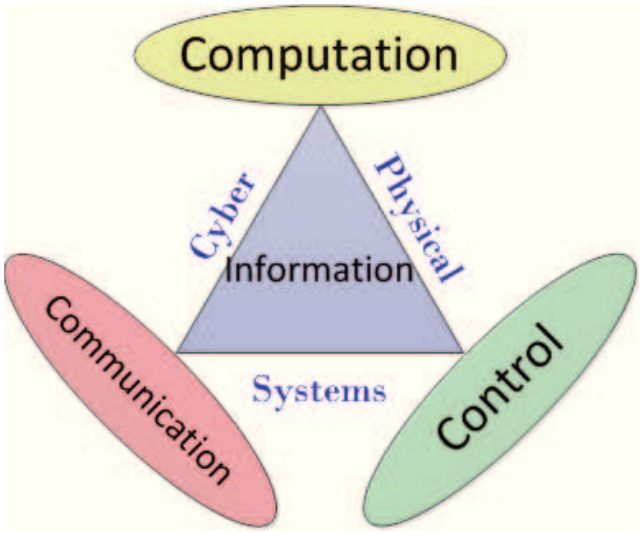


Figure 1.3: Cyber-Physical System representation.

Key features of a CPS are flexibility and adaptability, modularity, autonomy and reliability, safety and efficiency [4]. Cyber-Physical Systems must be able to reconfigure in automatic (or semi-automatic) way and self-organize to change task when needed; in that sense they are modular, meaning that they allow their combination and organization in order to build a more complex system made up by different modules (the single CPS components) communicating between them through the network, exchanging information to accomplish the final overall task. So the spread of CPSs leads to a distributed network of independent but interconnected intelligent autonomous modules, able to communicate, compute and collaborate. Despite their nature and characteristics make CPSs appear as intrinsically complex systems, they way they interface with humans must be as user-friendly as possible, using mobile devices such as smartphones, tablets and other intuitive equipment [5].

The flexibility and easiness of use of collaborative robots, make them strictly connected to CPS: the objective is to create an intelligent system able to per-

form dynamic and adaptable movements and to exploit sensor fusion to perform tasks and collect data, and which is able to collaborate together with other complex systems and with humans.

Until now robot software in the control interface has been mainly proprietary, due to the narrow suite of robot applications – such as material handling and welding - and a relatively concentrated number of suppliers. However, in the last years, the spread of robotics outside of factories and the much wider variety of uses, is leading to the development of open source software for robots control. This means that the software is distributed at source code level and can be redistributed to others unrestrictedly. The main benefits of this trend are both for developer and for users: open source software leads in fact to collaborative development and to more standardized robotic interfaces, which means both easiness of use, both re-usability of competences and knowledge among different kind of robots. On the other hand open source software would mean a lost of income for providers of proprietary software.

Speaking of academic results, a significant help comes also from the well known software framework ROS (Robot Operating System) [6]. The Robot Operating System is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms and allows to test and reuse works of other researchers in a simple and time-saving way. This also means that algorithms used for one kind of robot can easily be extended and reused for a similar one. As said, this is a need coming directly from the evolution of the new Industry 4.0: ROS is in fact also moving in that direction with ROS-Industrial (ROS-I) [7]. The consortium of ROS-I is made up by both of research institutes and robot constructors and the aim is to extend advanced capabilities of ROS to industrial robots and bring them to manufacturing.

The aim of this Ph.D. thesis is to analyse the current state of collaborative robotics and to provide tools that simplify the task of robot programming, making it time-saving and user-friendly, so that no particular knowledge in robotics is required to achieve that.

Therefore, the second chapter will explain why collaborative robotics is emerging and which are the key functionalities of a collaborative robot, analysing also the safety aspects related to ISO standards. The third chapter will propose a method to implement a manual guidance algorithm using a force/torque sensor and will provide experimental results and a use-case. The fourth chapter will focus on those tasks which need the integration of external sensors on the robot, making it difficult to use the basic simplified programming methods that come along with collaborative robots. Thus is here proposed a new method to teach those task in a quick and easy way through few steps. Finally in the fifth

chapter the importance of open source software is highlighted and discussed, and further experimental results are provided.

Chapter 2

Collaborative Robotics

The more extensive usage of robots nowadays is still in big companies and enterprises. The needs of small and medium enterprises (SMEs) can be very different and thus it is difficult for them to put money into traditional industrial robots, as the benefits and the income coming from having an industrial robot would not be enough to pay back the initial cost.

Typically SMEs can have a wide range of products which are constantly changing and for each new product the robot would need to be re-programmed accordingly, which is a time-expensive operation. Moreover traditional robotic cells, for safety reasons, must be isolated and closed with physical barriers, which means big amount of space required and low flexibility.

On the other hand, new collaborative robots, i.e. robots intended to physically interact with humans in a shared workspace, have several advantages which can satisfy the needs of SMEs production:

- they are easy to program, which means also spared time for that operation.
- they do not need physical barriers, which means less space is needed.
- they are intrinsically safe, thus they can work alongside and with human operators.
- they can be moved quickly where an increase of production is needed.

For those reasons, in the very last years, the usage of collaborative robots is increasing in the SMEs contest, enabling the automation of some repetitive tasks that were still done by humans.

A robot shall be counted as 'Collaborative' if at least one of the following three conditions is met:

- Manufactured as a 'Collaborative Robot' - Robot designed for direct interaction with a human within a defined collaborative workspace.
- Deployed in a 'Collaborative Operation' - State in which a purposely designed robot system and an operator work within a collaborative workspace.

- Installed in a 'Collaborative Workspace' - Space within the operating space where the robot system (including the workpiece) and a human can perform tasks concurrently during production operation.

The traditional and still most diffused method for robot programming is through the robot specific teach pendant. Task trajectories are taught to the robot specifying a set of points which the robot must pass through. However, teaching trajectories to the robot in this way turns out to be very slow and must be done over and over again each time there is a little change in the task. More recently, the availability of more performing hardware and advanced CAE (Computer Aided Engineering) tools promoted the use of off-line programming, which allows to check the feasibility of an industrial operation and even to program a task through the use of a Personal Computer, without the need to stop the production system. In this way the availability of kinematic models of the robot and of the corresponding simulation packages, opens a new range of capabilities, but on the other side an expert Engineer is needed to program the robot.

The necessity of easy programming in industrial context is also urged by the needs of the final customers; industry calls for robotic cells that are more and more flexible, modular and adaptable to different production requirements. The demand for an increasingly high productivity level in industrial manufacturing scenarios requires both shorter task execution times and faster robotic systems programming cycles. Moreover, the operator often is not a robotics expert, and teach-pendant programming has become a time-consuming and demanding task to be performed.

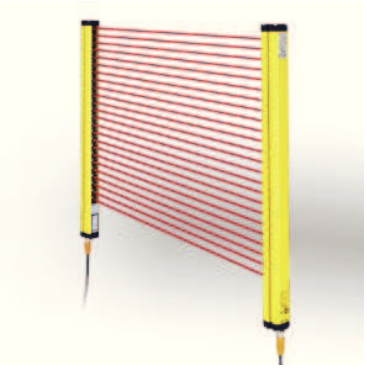
New collaborative robots on the other hand, can all be programmed in an easy and intuitive way. The operator can in fact take the robot by hand, thanks to different methods of manual guidance, and move it to the desired positions needed for the task, record the positions and subsequently use them for task programming. Moreover some collaborative robots have also some templates that can be used to program simple tasks (i.e. pick and place) even faster.

2.1 Safety in human-robot interaction

This new way of approaching robot programming brings with it a significant problem: the safety in physical human-robot interaction [8], which is a fundamental aspect of manual guidance. The main standards for granting a safe operation of industrial robots in factories are given by ISO 10218-1/2 [9, 10]; other safety norms, regarding the safe distance to be maintained between human and robots are contained in ISO 13855 [11]. In addition, since the interaction between humans and robots is now a very relevant argument inside the

robotics community, a new Technical specification (ISO/TS 15066 [12]) about this topic has been recently developed.

(a) Sick Laser Scanner



(b) Safety Light Curtains

(c) Pilz SafetyEYE

(d) SafetyEYE functioning

Figure 2.1: Safety devices: .

Accordingly to ISO standards, in order to use a non-collaborative industrial robot without physical barriers, some safety sensors can be used to monitor the area around the robot: if someone gets too close to the working area of the robot, then the robot is stopped with a safety stop function. Some safety sensors are showed in Figure 2.1 The computation of the minimum distance between human and robot below which the robot must stop [11], takes into account: the maximum speed of the robot, the theoretical maximum speed of the human, stopping time of the robot and the parameters. This distance could result often in a quite big distance if only horizontal sensors are used (it can easily reach 2 meters), which could mean frequent stops of the robot or big free space around the robot required. A way to reduce this minimum distance is to use both horizontal and vertical safety sensors, in order to create a virtual barrier around the robot, while also monitoring if someone approaches. Finally some robots controllers have the possibility to reduce in a safe way the maximum speed of the robot. In that way, two (or more, depending on the

controller and on the safety sensors) distances can be computed: the bigger one at which the robot does not stop, but reduces in a safe way its speed; the smaller one at which the robot stops. In that way the distance at which the robot stops is reduced even more.

The new technical specification ISO/TS 15066 specifies 4 different kinds of collaborative operations, which represent the only ways in which is possible to interact with a robot in a safe way:

- Safety-rated monitored stop
- Hand guiding
- Speed and separation monitoring
- Power and force limiting

The safety-rated monitored stop is the most basic collaborative operation: when the human operator enters the collaborative workspace (which should be monitored with some safety sensor as described above), the robot stops in a safe way. Once the operator exits the collaborative workspace, the robot can resume its work automatically, without the need of additional intervention by the user. Despite being very simple this function permits to share a workspace between human and robot.

Through the hand guiding (or manual guidance) collaborative operation, an operator can use a hand-operated device to transmit motion commands to the robot. The robot must move using a safety-rated monitored speed function to limit the maximum speed of the robot. The hand-operated device must be equipped with a enabling device and an emergency stop, usually mounted on the teach pendant or directly on the end-effector of the robot. This method also permits to integrate the hand guiding operation with some additional features, such as force amplification, virtual safety zones or tracking technologies, in order to improve the manual guidance experience for the operator and/or to prevent the operator from guiding the robot in some areas where he is not supposed to. Ideally, manual guidance methods can be implemented on any industrial robot which is complying with the specification of having a safety-rated monitored speed function and which is having a proper hand-operated device.

With the speed and separation monitoring collaborative operation, the robot and the operator may move concurrently in the collaborative workspace. The risk reduction is achieved by maintaining at least the protective separation distance between operator and robot at all times as described as follows:

$$S_p(t_0) = S_h + S_r + S_s + C + Z_d + Z_r,$$

where $S_p(t_0)$ is the protective separation distance at time t_0, S_h S_r and S_s are the contribution to the separation distance attributable respectively to: the operator's change in location, the robot reaction time and the robot's stopping distance. C is the intrusion distance, as defined in ISO 13855, which is the distance that a part of the body can intrude into the sensing field before it is detected. Finally Z_d and Z_r are the position uncertainties of the operator and of the robot respectively. The intrusion distance C can be reduced if both horizontal and vertical safety sensors are used, as the body of the operator is detected as soon as it enters the sensing field. During robot motion, the robot never gets closer to the operator than the protective separation distance and when the separation distance decreases to a value below the protective separation distance, the robot system stops with a safety-rated stop function. When the operator moves away from the robot system, the robot can resume motion automatically, while maintaining the protective separation distance. When the robot system reduces its speed, the protective separation distance decreases correspondingly. In that way the robot and the operator can share part of the collaborative workspace, without the need to completely stop the robot motion.

Finally the last collaborative operation, the power and force limiting, is probably the most important one, as describes the specification for robot manufacturers to build collaborative robots. The ISO/TS 15066 specifies 3 contact situations:

- intended contact situations that are part of the application sequence;
- incidental contact situations, which can be a consequence of not following working procedures, but without a technical failure;
- failure modes that lead to contact situations;

it also divides the types of contacts into two categories:

- **Quasi-static contact**: This includes clamping or crushing situations in which a person's body part is trapped between a moving part of the robot and another fixed or moving part of the work cell. In such a situation, the robot would apply a pressure or force to the trapped body part for an extended time interval until the condition can be alleviated.
- **Transient contact (or dynamic impact)**: This describes a situation in which a person's body part is impacted by a moving part of the robot and can recoil or retract from the robot without clamping or trapping the contacted body area, thus making for a short duration of the actual contact.

So, one the main reasons why collaborative robots can work alongside humans without physical barriers and without safety sensors is their capability of sensing collisions and properly react to them. There are different ways with which robot can sense collisions: the most precise one (but also the most expensive) is to have joint torque sensors mounted at each joint of the robot. Knowing the planned torque which has been commanded to the robot and making a comparison with the actual one measured from the sensors, some residual can be computed in order to detect collisions [13]. Clearly such an approach requires the robot to react very quickly when a collision occurs in order to be safe. Finally the ISO/TS 15066 establish also the maximum amount of force/pressure that the robot should not exceed, so that the robot cannot injury the human operators in a severe way. That's why most of collaborative robots have also a round and ergonomic shape, without edges and sometimes also with some soft covers, in order to reduce the impact force of the robot during unexpected collisions.

However having a collaborative robot, does not necessarily means that it can always be used alongside human operators. In fact if the robot is carrying an end-effector (or handling some components) that can be potentially dangerous for humans (sharp edges, heavy and rigid components, etc.), the overall solution could not be any more considered safe and collaborative. In order to understand the potential risks of an application, just like for other machines, a risk assessment must be done, in order to prevent and/or limit possible hazards for humans.

2.2 Collaborative robots on the market

Today, a wide range of collaborative robots (Figure 2.2) which are compliant to ISO safety standards can be found on the market, differing for technical specification (like speed, repeatability, payload), but also for cost.

The ABB YuMi is a collaborative dual-arm robot targeted for electronic assembly market, or in general to handle small and light parts. The two arms have soft components to prevent dangerous collisions; moreover the robot comes with a special wrist mechanism to prevent crush injuries, thanks to which if the robot senses elevated forces applied at the end-effector it puts the last two joints of that arm in a completely passive behaviour and stops the task it was performing. The YuMi is very compact and quite light robot (38 kg, including the controller which is in the "torso"), and the two 7-DOF arms give great flexibility to perform the most challenging manipulation tasks. On the other side the very low payload (0.5 kg per arm; 0.25 kg per arm, if it mounts one of its default grippers, which can include a vacuum cup and a smart camera) limits the possible uses.

Recently, ABB bought another company (Gomtec), which realized the collaborative robot Roberta. There were three versions of Roberta, differing for payload (4 kg, 8 kg and 12 kg) and for reach(600 mm, 800 mm and 1200 mm). A peculiarity of this robot was the very intuitive programming, which could also memorize and reproduce whole trajectories; all the different modes were selectable rotating a ring on the end-effector, so it was possible to teach a task without using any teach-pendant, but just interacting with the robot. Soon ABB should release a new version of the Roberta robot.

The APAS collaborative solution from Bosch consists of a standard industrial robot with a tactile soft skin to detect unexpected collisions, giving instant feedback to the controller which stops the robot. To use this solution the robot controller must have the safety-rated monitored stop and safety-rated monitored velocity functions. The robotic arm is mounted on a mobile platform that can be moved manually, so to be able to quickly move the robot where needed. Moreover the area around the robot is also monitored and the robot slows down its task when someone gets too close. Finally the APAS robot comes with a 3-finger gripper and with integrated cameras (2D or 3D).

The new green robot from Fanuc (CR-35iA), is the collaborative robot that is closest to traditional industrial robots, speaking of repeatability, reach and payload. The CR-35iA is able to carry weights up to 35 kg with a repeatability of $\pm 0.08mm$ and has 1813 mm of reach. With this specifications the robot is able to do tasks that other collaborative robots could not be able to do, still being safe thanks to the soft cover and the torque sensors mounted at each joint. On the other hand the robot from Fanuc is less flexible, as it is quite heavy and its pedestal is designed to be fixed to the ground, so it can not be moved easily from one place to another like the other collaborative robots.

The 7-DOF robotic arm Kuka IIWA (Intelligent Industrial Work Assistant) is the evolution of the LWR 4+ in terms of physical features: the robot is in fact equipped with torque sensors on each joint, has high-performance collision detection algorithms and has an incredibly good ratio between weight and payload. The controller (the Kuka Sunrise) is instead quite different from the previous version, as it uses Java as programming language, offering a library containing a lot of high-level functions.

The Baxter robot from Rethink Robotics is a very versatile and easy to use robot. In fact in order to use the Baxter robot no programming knowledge is needed and can be trained in few minutes. Moreover Baxter is a complete system, comprehensive of different sensors, so for simple tasks no integration with other hardware is needed. Finally it has a very low price (the starting price is around 20 k$), considering the number of DOFs that the Baxter has and the various sensors and end-effector that can be mounted on it. On the other hand the Baxter robot has a very low precision and repeatability (due to

mechanical elasticity of joints), which clearly limits the range of applications in which it can be applied.

Sawyer is the more recent collaborative robot from Rethink Robotics. Unlike Baxter, it is made to work in fixed position and it is not mounted on a mobile base. Moreover it is more repeatable respect to the Baxter robot, but still quite far from other collaborative robots. Finally Sawyer mounts a Cognex camera for a more precise vision system and has also force sensing embedded at each joint.

Finally, the robots from Universal Robot consist of six-axis arms of different dimensions: the UR10 can carry up to 10 kg and has a reach of 1600 mm, the UR5 has a 5 kg payload and a reach of 900 mm; finally the newest UR3 has a payload of 3 kg and a reach of 500 mm. All UR robots are quite precise and lightweight, but also quite cheap. As all collaborative robots, they can be moved by hand for a fast teaching. The Universal Robots are probably the collaborative robots that are currently most diffused, especially in industrial context, thanks to their good price (the most expensive one, the UR10 is below 30k€) and their good capabilities.

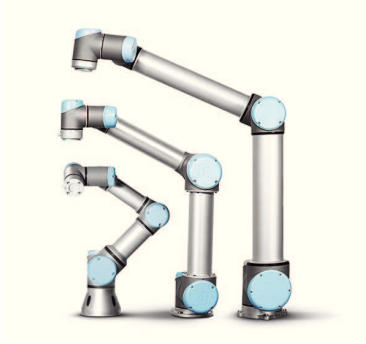
(a) ABB YuMi

(b) Gomtec Roberta

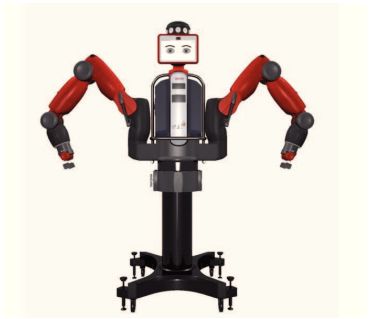
(c) Bosch APAS

(d) Fanuc CR-35iA

(e) Kuka IIWA



(f) UR3, UR5 and UR10



(g) Baxter Robot

(h) Sawyer robot

Figure 2.2: Collaborative robots on the market.

Chapter 3

Enabling Programming by Demonstration

The new promising way for robot programming is the Programming by Demonstration (PbD) [14, 15], which allows the operator to teach to the robot a generalized version of a task in an easy and natural way, thus requiring no experience in robot programming. In the teaching process, apart from the joints position, several measurements can be taken into account and integrated together, such as force profiles or voice commands [16]. These methods can then be customized, depending on the task, in order to handle exceptions using different strategies [17]. An ideal solution would be a scenario where the operator's movements are tracked while he is doing the task and then, using machine learning methods, the capability of reproducing the task is taught to the robot. This is a challenging topic of research of recent years; in addition, from an industrial point of view these methods are still not robust enough. Moreover, if during the task execution lifting heavy weights is required, the operator could not have enough physical strength to perform the task. As a consequence of this, a method that is more robust and that could manage tasks with heavy weights lifting is the manual guidance (or walk-through). This type of control lets the user move physically the robot by hand in a free way, instead of moving it using the teach pendant; these movements can then be encoded and used for learning a generalized version of the task or for reproduction.

Another way to teach a task to a robot is using haptic devices, which can be used to command velocities to the robot, while giving a force feedback to the user; in this way also force tasks can be taught to the robot. Such devices are very expensive and for this reason at the moment they cannot be adopted in industrial contexts; on the other hand, they are mostly used in medical or rehabilitation robotic applications.

3.1 Manual Guidance State of Art

In order to achieve robot manual guidance, several control approaches that use different kinds of sensors, or more in general hardware, can be implemented. Most types of control schemes make use of a force/torque sensor mounted at the wrist of the robot; the use of such sensors allows the design of control schemes that grant a very precise motion of the end-effector, such as admittance/impedance control [18–20] or force control [21]. The main drawback of the majority of these methods is that, apart from the cost of the force/torque sensor, they rely on the availability of a robust dynamic model of the robotic arm. Other researchers propose advanced variants of both types of control, such as variable impedance control [22], adaptive admittance control [23] or admittance control based on virtual fixtures [24]: these are software-generated motion guidance, based on a vision system that extrapolates the geometry of the robot and of the workspace needed to compute the motion constraints. The hybrid force/motion control, that can be achieved both at kinematic [25] and dynamic level [21], describes the task constraints combining a force control and a velocity control, through selection matrices. Since few years, some new lightweight robots come with an implementation of compliance control, as for example the Kuka LWR IV [26] or its evolution, the Kuka iiwa. Again, this is a clear sign that the whole robotic community is moving towards the robot co-worker concept.

In order to make the operator feel like he/she is moving a tool of reduced mass (instead of an heavy and stiff robot), the dynamics of the robot motion can be described as a "virtual tool": the end-effector is modelled as a virtual point of chosen mass which can be placed arbitrarily in a free space environment. Then the robot can be controlled using admittance control [27] or a simple kinematic control [28]. Another interesting approach is to mix up the feedback from the force/torque sensor with visual information [29]. Here the maximum end-effector velocity allowed is proportional to the distance between the robot and surrounding obstacles; in that way the robot prevents collisions. Other works have developed a force control without using force sensors, by using observation methods [30] or seeing the external force as a disturbance [31].

Another way to achieve manual guidance is using fault detection and identification (FDI) methods to sense external forces [13]: in this case, external forces can be detected defining a residual that observes and measures differences between the commanded torque and the actual torque. This method has the drawback that the operator must move the joints one by one, thus resulting in an unnatural feeling of motion; moreover the dynamic model of the robot must be available; in the end, this approach is mainly suited for unexpected collisions detection and reaction.

Other works try to use the currents of motors to estimate the torque at each joint [32]: the torque estimation can then be used to achieve manual guidance. In particular, in [33] low-pass and high-pass filters are applied to motors currents to detect human intentional interaction. Despite the innovation of these methods, they are not suited for industrial use, both in terms of robustness and precision of movements. Moreover, the friction of the motor reductor affects the readings of the currents, and its effect can change during time and in different temperature conditions [34].

Finally, a more recent way to perform kinesthetic movements is through backdrivability, like the Barret WAM [35], which allows robots with backdrivable motors to be guided by the user keeping them in zero-gravity control mode; these robots are used especially to test learning algorithms [15]. The drawback of backdrivability is that it is suited only for lightweight or small robots, because of the robot inertia; thus it can not be applied to industrial robots. Moreover, in this case the user must move the joints one by one, which makes it difficult (especially for robots with many degrees of freedom) to perform natural and smooth movements.

3.2 Manual Guidance implementation

In this section, an implementation of manual guidance based on the use of a force/torque sensor is presented. The sensor is mounted between the wrist and the end-effector of the robot, so a gravity compensation of the weight of the tool is needed. in the following subsections the gravity compensation and control algorithms will be explained.

3.2.1 Gravity compensation

In order to distinguish the input force of the user from the gravity force on the end-effector, a static model of the robotic system (i.e. the robotic arm, the force/torque sensor and the end-effector) must be developed. Since the manual guidance application is here considered as semi-static (accelerations applied by the operator are considered to be very small), a full dynamic model is not necessary. Starting from the forces and torques measured by the sensor, respectively ${}^s\boldsymbol{f}_m$ and ${}^s\boldsymbol{\tau}_m$ we have:

$$
\begin{aligned}
{}^s\boldsymbol{f}_m &= {}^s\boldsymbol{f}_h + {}^s\boldsymbol{g} && (3.1)\\
{}^s\boldsymbol{\tau}_m &= {}^s\boldsymbol{\tau}_h + {}^s\boldsymbol{\tau}_g, && (3.2)
\end{aligned}
$$

where ${}^s\boldsymbol{f}_h$ and ${}^s\boldsymbol{\tau}_h$ are respectively the force and torque applied by the human operator to the end-effector in sensor frame, while ${}^s\boldsymbol{g}$ and ${}^s\boldsymbol{\tau}_g$ are the gravity

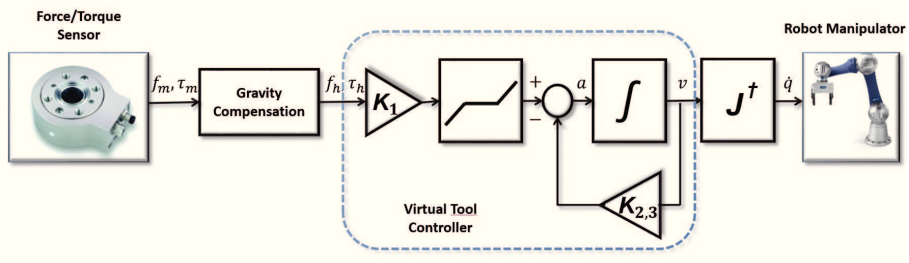


Figure 3.1: Overall scheme of the control.

force and torque acting at the center of mass of the end-effector expressed in sensor frame. Expressing these equations into world frame leads to:

$$
\begin{aligned}
{}^{w}\boldsymbol{R}_{s}\,{}^{s}\boldsymbol{f}_{m} &= {}^{w}\boldsymbol{f}_{h} + {}^{w}\boldsymbol{g} && (3.3)\\
{}^{w}\boldsymbol{R}_{s}\,{}^{s}\boldsymbol{\tau}_{m} &= {}^{w}\boldsymbol{\tau}_{h} + {}^{w}\boldsymbol{\tau}_{g}, && (3.4)
\end{aligned}
$$

where now ${}^{w}\boldsymbol{f}_{h}$, ${}^{w}\boldsymbol{\tau}_{h}$, ${}^{w}\boldsymbol{g}$ and ${}^{w}\boldsymbol{\tau}_{g}$ are expressed in world frame; ${}^{w}\boldsymbol{R}_{s}$ is the rotation matrix from sensor frame to world frame, computed as ${}^{w}\boldsymbol{R}_{s} = {}^{w}\boldsymbol{R}_{ee}\,{}^{ee}\boldsymbol{R}_{s}$, where ${}^{w}\boldsymbol{R}_{ee}$ is obtained from direct kinematics and ${}^{ee}\boldsymbol{R}_{s}$ represents the rotation matrix from the sensor frame to end-effector frame (which is constant). Equations (3.3-3.4) can be resolved to find out the force/torque applied by human:

$$
\begin{aligned}
{}^{w}\boldsymbol{f}_{h} &= {}^{w}\boldsymbol{R}_{s}\,{}^{s}\boldsymbol{f}_{m} - {}^{w}\boldsymbol{g} && (3.5)\\
{}^{w}\boldsymbol{\tau}_{h} &= {}^{w}\boldsymbol{R}_{s}\,{}^{s}\boldsymbol{\tau}_{m} - {}^{w}\boldsymbol{r} \times {}^{w}\boldsymbol{g}, && (3.6)
\end{aligned}
$$

with

$$
{}^{w}\boldsymbol{\tau}_{g} = -{}^{w}\boldsymbol{r} \times {}^{w}\boldsymbol{g},
$$

where ${}^{w}\boldsymbol{g}$ is the weight of the end-effector in world frame, ${}^{w}\boldsymbol{r}$ is the position vector of the center of mass of the end-effector with reference to the origin of the sensor frame, expressed in world frame. In such a way the gravity-compensated force and torque applied by human can be considered in the control loop. For simplicity, in the next subsection ${}^{w}\boldsymbol{f}_{h}$ and ${}^{w}\boldsymbol{\tau}_{h}$ will be referred to as $\boldsymbol{f}$ and $\boldsymbol{\tau}$.

3.2.2 Virtual Tool Controller

The gravity compensated forces and torques, obtained as described in the previous subsection, are fed to a virtual tool controller: it has been reckoned that in comparison with a proportional controller, it would yield a more natural feeling for the operator when moving the end-effector. The virtual tool controller

basically commands an acceleration to the robot end-effector that is proportional to the force/torque applied by the human and adds a damping factor to decelerate the robot and make it stop fast when the external force is not applied any more. In this way, the dynamics of the end-effector is modelled as a virtual point whose lumped mass is located at the center of mass of the end-effector. Since the gravity of the real mass of the robot end-effector has been compensated as described in the above subsection, it has not been considered in the controller.

Since the robot is controlled in the velocity domain, the desired translation and rotation end-effector velocities at time t are defined as:

$$\boldsymbol{v}_t^t = \begin{cases} (\boldsymbol{K}_1(\boldsymbol{f} \pm \boldsymbol{t}_f) - \boldsymbol{K}_2 \boldsymbol{v}_{t-1}^t)\Delta t + \boldsymbol{v}_{t-1}^t, & \text{if} |\boldsymbol{f}| > \boldsymbol{t}_f \\ -\boldsymbol{K}_3 \boldsymbol{v}_{t-1}^t \Delta t + \boldsymbol{v}_{t-1}^t, & \text{if} |\boldsymbol{f}| < \boldsymbol{t}_f \end{cases} \tag{3.7}$$

$$\boldsymbol{v}_t^r = \begin{cases} (\boldsymbol{K}_1(\boldsymbol{\tau} \pm \boldsymbol{t}_\tau) - \boldsymbol{K}_2 \boldsymbol{v}_{t-1}^r)\Delta t + \boldsymbol{v}_{t-1}^r, & \text{if} |\boldsymbol{\tau}| > \boldsymbol{t}_\tau \\ -\boldsymbol{K}_3 \boldsymbol{v}_{t-1}^r \Delta t + \boldsymbol{v}_{t-1}^r, & \text{if} |\boldsymbol{\tau}| < \boldsymbol{t}_\tau \end{cases} \tag{3.8}$$

Equations (3.7) and (3.8) represent the implementation of the virtual tool controller: $\boldsymbol{t}_f$ and $\boldsymbol{t}_\tau$ are threshold vectors for noise filtering, $\boldsymbol{K}_1$, $\boldsymbol{K}_2$ and $\boldsymbol{K}_3$ are positive diagonal gain matrices and Δt is the control loop cycle time. Signs in equations (3.7) and (3.8) depend on the sign of $\boldsymbol{f}$ and $\boldsymbol{\tau}$ respectively. Analysing the virtual tool controller, it can be noticed that there is a first term which is proportional to the force and torque applied by the human, and a damping factor which is needed to provide resistance. In particular, when the absolute value of force/torque is smaller than the threshold, the damping factor is needed to decelerate the motion of the robot. This is why two different gain matrices ($\boldsymbol{K}_2$ and $\boldsymbol{K}_3$) are used for the two cases: because when no forces are sensed, the robot needs to decelerate faster. For this reason the elements of $\boldsymbol{K}_3$ are chosen much bigger than the ones of $\boldsymbol{K}_2$.

Finally the end-effector velocities are transformed in joint velocities:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^\dagger \boldsymbol{v}, \tag{3.9}$$

where $\dot{\boldsymbol{q}}$ is the vector of joint velocities, $\boldsymbol{J}^\dagger$ is the pseudo-inverse of the Jacobian (concerning both position and rotation) and $\boldsymbol{v} = [\boldsymbol{v}^t \boldsymbol{v}^r]'$. It is noted that when the Jacobian is square and non-singular (i.e. the number of joints is equal to the number of degrees of freedom of the task), a simple inversion of the Jacobian could be used instead.

Figure 3.1 illustrates the overall control scheme used in the proposed ap-

proach: forces and torques measured from the sensor are first gravity compensated as described in previous section, then used by the virtual tool controller to obtain a velocity reference for the end-effector. This reference is finally mapped on robot's joints using the pseudo-inverse of the Jacobian.

If the end-effector gets in contact with the environment, the sensor will sense a normal force and will promptly move away from the touch point: such motion will be just a transient effect, characterized by a velocity proportional to the approaching speed and thus to the force sensed by the sensor. If it is needed to interact with the environment (e.g. to follow a profile which has to be deburred) one could manually disable the command along that direction (i.e. probably the end-effector normal axis). However, in the chosen application domain, that is quality control, it is uncommon to need such solution; most of the tasks which need interaction with the environment, can be programmed offline and then the robot must only know "where" to execute the task (e.g. the proposed approach in section 3.4).

Even though robot kinematic singularities are easily avoidable by the user, a solution consists in using a damped pseudo-inverse of the Jacobian, defined as:

$$\boldsymbol{J}_{\lambda}^{\dagger} = \boldsymbol{J}^{T}\left(\boldsymbol{J}\boldsymbol{J}^{T} + \lambda^{2}\boldsymbol{I}\right)^{-1}, \tag{3.10}$$

with damping factor $\lambda << 1$; using the SVD decomposition $\boldsymbol{J} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{T}$, equation (3.10) leads to:

$$\boldsymbol{J}_{\lambda}^{\dagger} = \boldsymbol{V}\boldsymbol{\Sigma}_{\lambda}^{\dagger}\boldsymbol{U}^{T},$$

where $\boldsymbol{\Sigma}_{\lambda}^{\dagger}$ is a diagonal matrix, with elements:

$$\tilde{\sigma_i} = \frac{\sigma_i}{\sigma_i^2 + \lambda^2},$$

where σ_i is the i^{th} singular value of the Jacobian matrix. This means that $\boldsymbol{J}_{\lambda}^{\dagger}$ has singular values $\tilde{\sigma_i}$, with $\tilde{\sigma_i} \to 1/\sigma_i$ for $\lambda \to 0$. In this way the robot can pass through singularities, even though obviously only velocities in the manipulability ellipsoid can be accomplished.

3.3 Experimental results

Experiments have been carried out using the Schunk Powerball robotic arm. As Figure 3.2 shows, the Schunk Powerball consists of three modules, each one containing two motors, for a total of six joints. The robotic arm is controlled with a personal computer in the velocity domain using the ROS packages *schunk_lwa4p* and *ipa_canopen*. No ROS tools for kinematics or planning are used, joint velocities are commanded to the robot each 10 milliseconds and

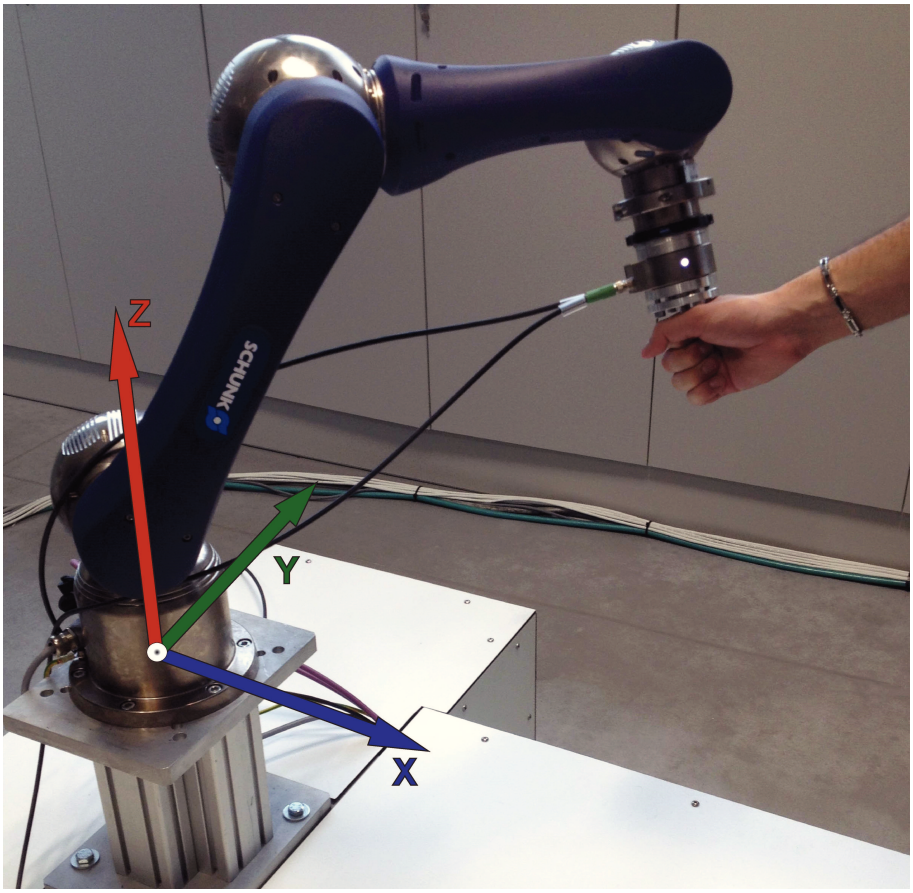


Figure 3.2: Schunk Powerball robotic arm during manual guidance control. The blue, green and red arrows represents respectively x, y and z axis of reference frame.

these commands are then executed by the internal controller of each module; thus the proposed approach is applicable to any industrial robot, provided that the power needed to control the robot complies with the maximum allowed by ISO standards, which is the case of the Powerball robot. Thus, in order to make the experimental set-up fully compatible with ISO standards, an emergency stop and an enabling device to start guiding the robot should be added near the end-effector. The force/torque sensor (KMS40 from Weiss Robotics) is mounted between the sixth joint and the end-effector and communicates with the pc using an ethernet protocol, updating the force and torque signals at a frequency of 500 Hz. So in that case the bottleneck for the algorithm implementation is the robot control frequency (100 Hz), which is the actual frequency at which the algorithm will run. Of course, having an higher control

frequency would make the control algorithm more reactive to external inputs; anyhow for a manual guidance algorithm, were the inputs come from a human operator (usually at low frequencies), a control loop of 100 Hz brings satisfying results.

The experiments made using the proposed approach resulted in smooth and natural motions of the robot.

Parameters of (3.7) and (3.8) were chosen as follows:

$$\boldsymbol{K}_1 = \begin{pmatrix} 0.05 & 0 & 0 \\ 0 & 0.05 & 0 \\ 0 & 0 & 0.05 \end{pmatrix} \boldsymbol{K}_2 = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

$$\boldsymbol{K}_3 = \begin{pmatrix} 8 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 8 \end{pmatrix},$$

for both the force and torque equations. For the thresholds, the values chosen are the following:

$$\boldsymbol{t}_f = \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix} \boldsymbol{t}_\tau = \begin{pmatrix} 0.65 \\ 0.65 \\ 0.65 \end{pmatrix},$$

meaning that all forces below $2N$ and all torques below $0.65Nm$ are considered as noise. All those parameters where chosen experimentally.

The small value of $\boldsymbol{K}_2$ gain makes the robot lighter to the user, while the high value of $\boldsymbol{K}_3$ makes the robot stop almost instantly when no external forces are sensed. This resulted in a both smooth and precise motion of the robot, achieving the expected result. Experimental results are shown in the remainder of the section.

In the first test the robot is moved through an orientation singularity; in particular the 4^{th} and 6^{th} joints are aligned. As showed in the plots of Figure 3.5, the transition through the singularity results in a smooth movement. Notice that, as Figures 3.3 and 3.4 show, the resulting end-effector velocity is a low-pass filtered version of the force and torque signals; this is fundamental, because a simple proportional controller would be otherwise affected by the noise of the sensor.

The second experiment is carried out to test the precision of movements. In order to make the robot more precise (but also stiffer) the elements of matrix $\boldsymbol{K}_1$ are lowered. The plots in Figure 3.6 show how the operator is able to move around the robot in a precise way, being able to describe a small rectangle in the x-y plane, while keeping constant the orientation and the position along the z axis. This means that the maximum precision that is achievable is the operator's precision. Actually the precision could be increased even more, rising

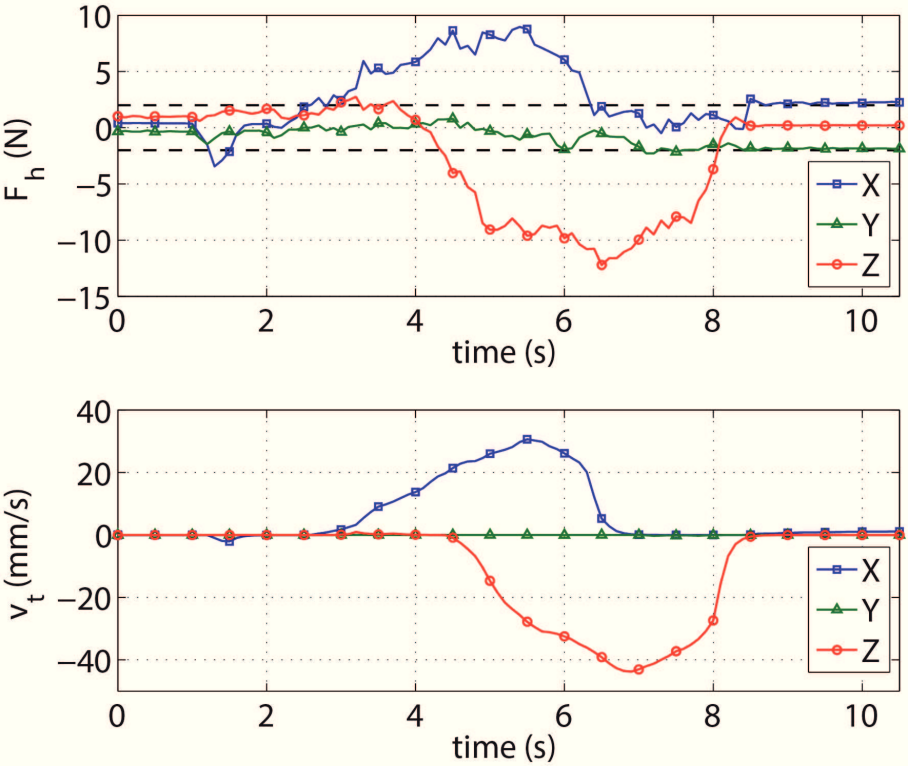


Figure 3.3: Robot guided through an orientation singularity. Above: Force applied by the human along x, y and z axis (respectively in blue, green and red), the black lines represent the threshold; bottom: The resulting end-effector translational velocity.

the threshold values in equations (3.7) and (3.8); in this way the operator should use more force to move a little the robot and the precision would be the robot's one. On the other hand this would result in an unnatural feeling for the operator and in a very stiff motion, so should be used only when strictly necessary.

3.4 Use-Case

The implementation of the robot manual guidance described in the above sections can be used to teach a task to the robot. The chosen application domain is the quality control of front panels of electronic devices (Figure 3.7). A useful task that can be taught to the robot is the test of buttons on panels in order to check how much force is needed to press the buttons. The test of panels is a simple task, but due to the continuous changing of the models it is difficult to

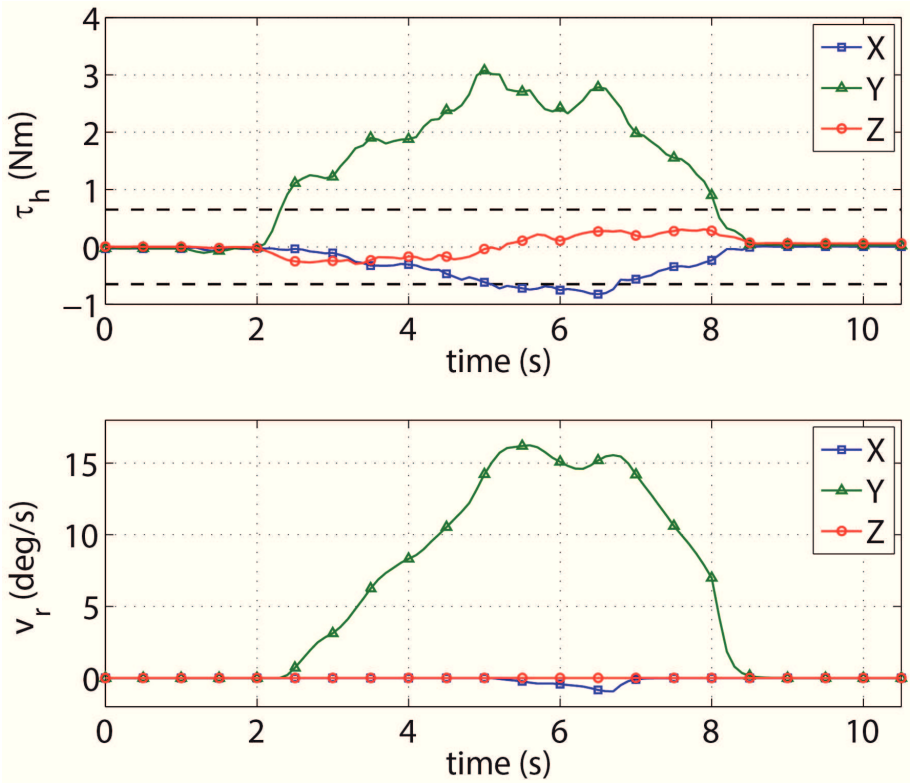


Figure 3.4: Robot guided through an orientation singularity. Above: Torque applied by the human around x, y and z axis (respectively in blue, green and red), the black lines represent the threshold; bottom: The resulting end-effector angular velocity.

automatize; with the proposed approach, each time a new kind of panel must be tested, teaching the new task to the robot can be done in an easy and time-saving way. The user can teach the positions of the buttons by simply bringing the robot's end-effector above them (so without pressing them) and saving the joint configurations through the user interface. The end-effector of the robot has been designed on purpose: it has the double advantage of making the manual guidance algorithm ergonomic and of being able to press buttons for that specific application. In the reproduction of the task, the robot will first move through the saved positions using a simple kinematic control for positioning; when the robot reaches the saved positions, it will then go down along the Z axis direction using an impedance control, implemented as follows:

$$\boldsymbol{v} = \boldsymbol{v}_h + \boldsymbol{v}_I, \tag{3.11}$$

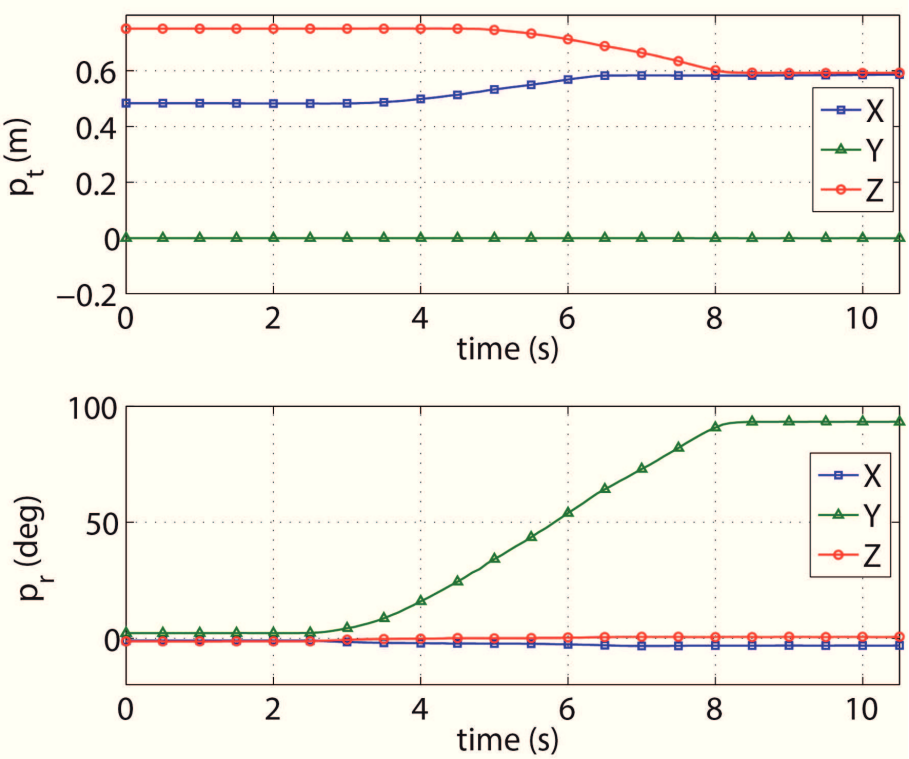


Figure 3.5: Robot guided through an orientation singularity. Above: translational position of the end-effector along x, y and z axis (respectively in blue, green and red); bottom: RPY-orientation of the end-effector.

where $\boldsymbol{v}_h$ is exactly the velocity obtained from equations (3.7) and (3.8), using a $\boldsymbol{K}_1$ with very low coefficients; the $\boldsymbol{v}_I$ is a velocity resulting from the impedance control, computed as:

$$\boldsymbol{v}_I = \boldsymbol{K}_{imp} \left(\boldsymbol{p}_r - \boldsymbol{p}_{EE} \right), \tag{3.12}$$

where $\boldsymbol{K}_{imp}$ is a positive diagonal matrix representing the spring stiffness, while $\boldsymbol{p}_r$ and $\boldsymbol{p}_{EE}$ are respectively the rest position of the virtual spring and the actual position of the end-effector of the robot; a damping factor is not considered in equation (3.12), since the $\boldsymbol{v}_h$ already includes one. Notice that, in order to make the robot move along the z axis, the point $\boldsymbol{p}_r$ is moving. Moreover, since the robot is desired to be compliant along z-axis and rigid along all other directions, equation (3.11) in practice is only implemented for the z-axis component, while the other components are equal to zero. After the force sensed along the z axis exceeds a maximum value admitted (in this case

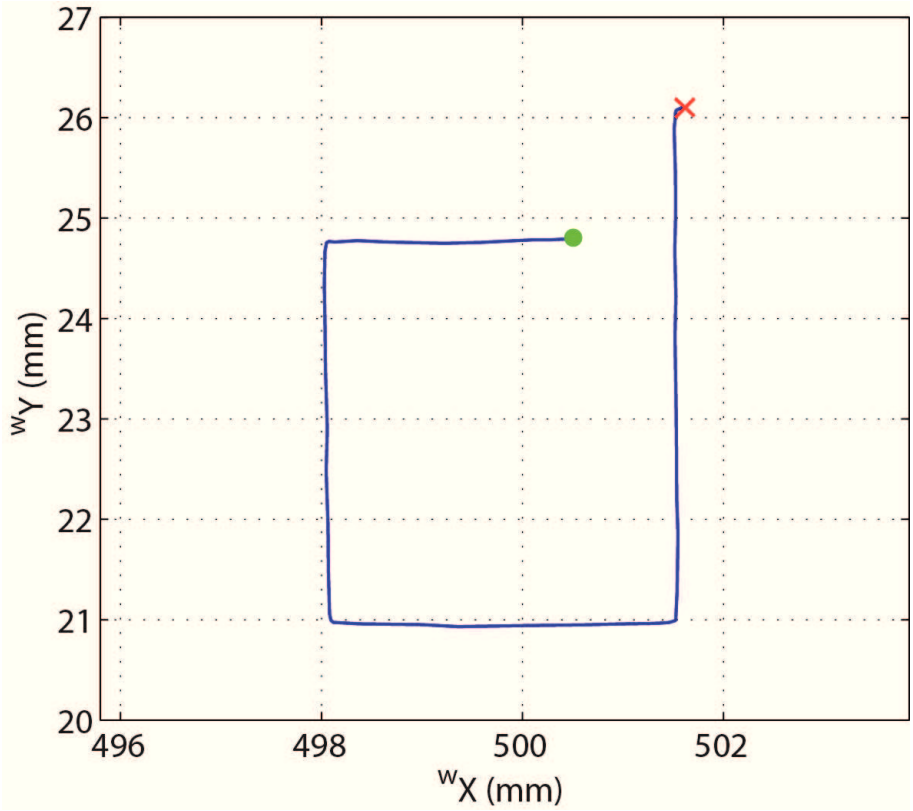


Figure 3.6: Robot guided through a rectangle in the x-y plane. Green dot and red cross represents start and end positions respectively.

$15N$), the impedance control is stopped ($\boldsymbol{v} = \boldsymbol{v}_h$), so that the pushing force does not increase too much. After the robot has pressed a button, it will first go back to the current approach position and then it will move to the next approach position ready to test the next button.

Figures 3.8 and 3.9 respectively show the forces (and the resulting velocities) along the 3 cartesian axis, and the position of the end-effector of the robot during the teaching process of a task. In this task 9 positions (taken above 9 buttons to test) are taught, showed by the red arrows in Figure 3.9; the operator is communicating to the program user interface (pressing a button on a keyboard) that the current position must be saved for reproduction, that is why in these instants the position of the end-effector keeps constant. Again, it can be noticed how the proposed approach results in a smooth and precise movement of the end-effector.

Regarding the reproduction, to make better understand what happens when the robot presses a button, in the plots it is only reported the first of the nine

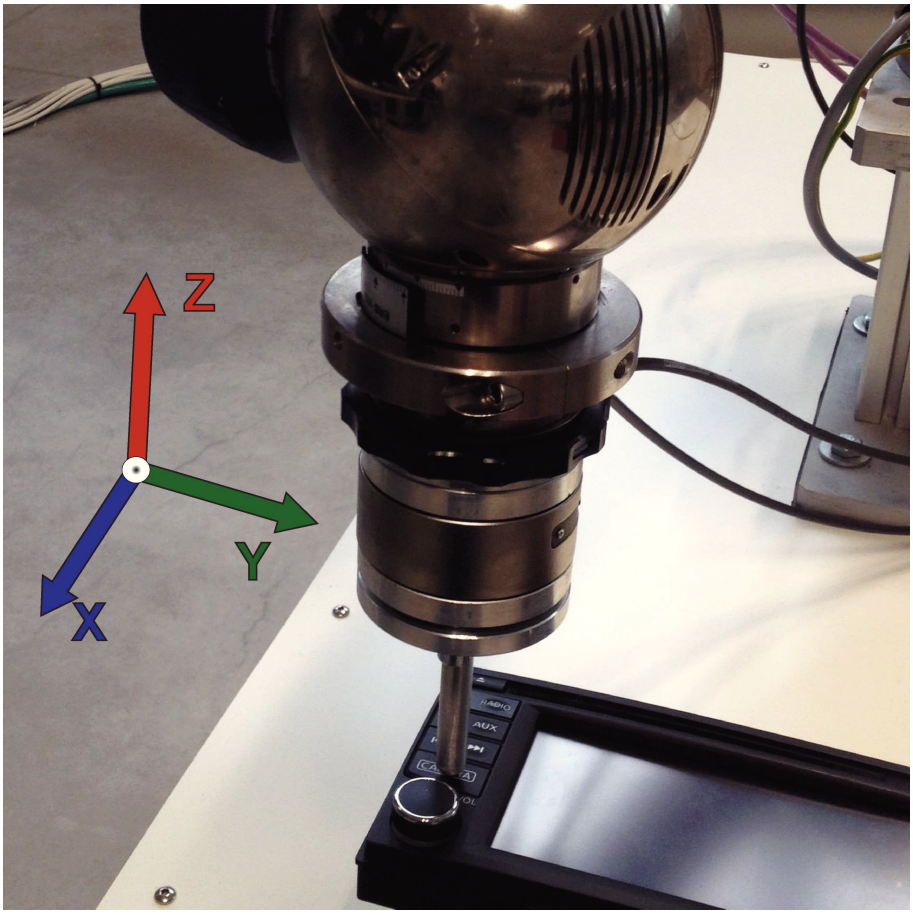


Figure 3.7: Schunk Powerball robotic arm testing a front panel. The blue, green and red arrows represents respectively x, y and z axis of reference frame.

taught positions in Figures 3.8 and 3.9. During the reproduction phase the robot first moves with kinematic control to a taught position, and then (Figure 3.10) it will start moving along the negative direction of Z axis in impedance control; the performed experiments show that at the instant when the button is pressed, a small peak of the external force is sensed (see Figure 3.10). In this way the value of the force needed to press the button can be easily found out. It is noted that in the bottom plot of Figure 3.10 the resulting end-effector velocity is only made up of the impedance control effort until the force sensed exceeds the black threshold (top plot of Figure 3.10).

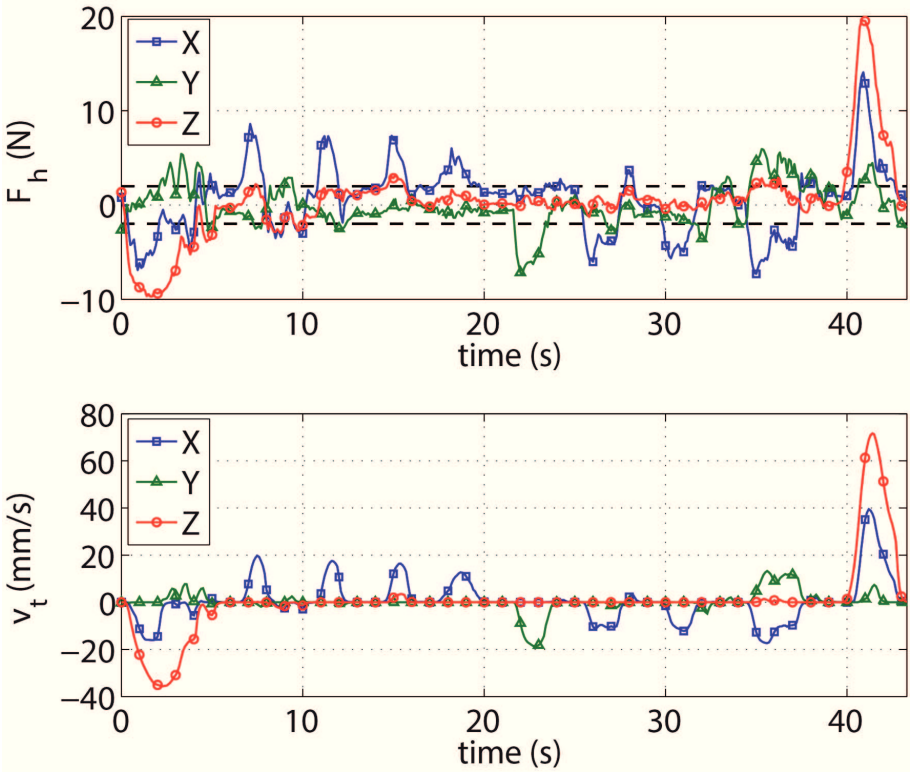


Figure 3.8: Teaching to the robot the positions of nine buttons to press. Above: Force applied by the human along x, y and z axis (respectively in blue, green and red), the black lines represent the threshold; bottom: The resulting end-effector translational velocity.

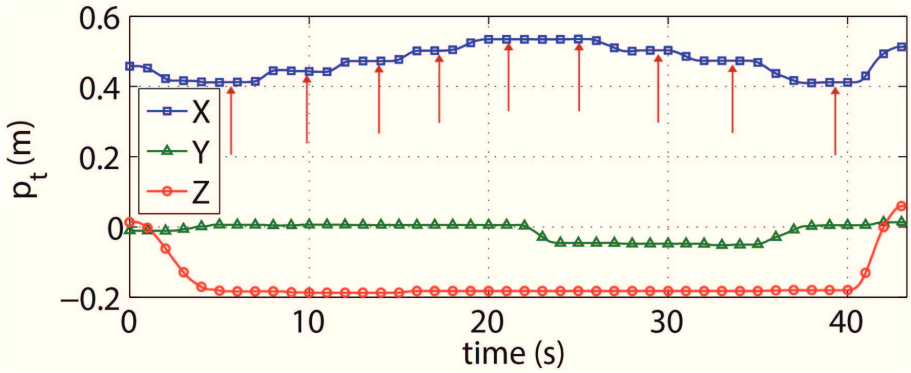


Figure 3.9: Teaching to the robot the positions of nine buttons to press. Translational position of the end-effector along x, y and z axis (respectively in blue, green and red); the red arrows show the positions of the nine points that are saved for the reproduction.

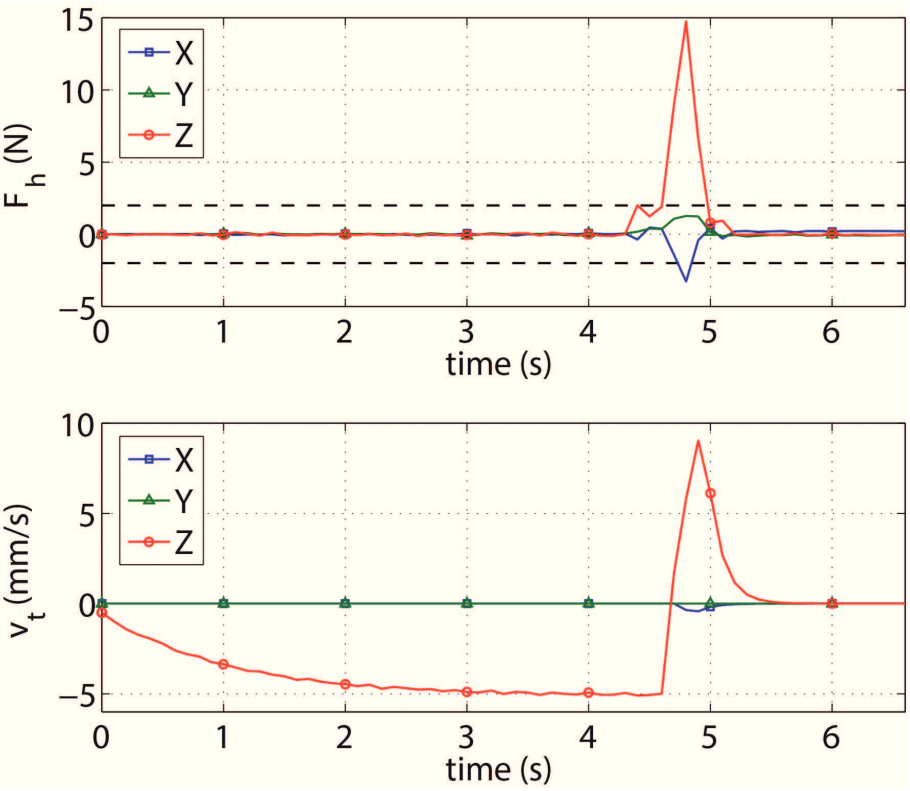


Figure 3.10: Testing a single button. Above: external forces sensed along x, y, and z axis (respectively in blue, green and red); bottom: total end-effector velocity resulting from equation (3.11).

Chapter 4

Simplifying Robot Programming Through Haptic Learning

Without robots, many products of today would not be possible, as their precision, speed and ability to repeat tasks without deviation is an absolutely essential feature for today's production lines. While initially only large companies could afford expensive robots to scale up mass production of goods, they are finally becoming affordable even for small and medium sized businesses. But there is still a major problem with the use of robots in SMEs: Programming robots is very hard, time consuming and expensive [36]

To fulfill the hard requirements of SMEs modern production robots need to be able to work together with the human workers, and adapt quickly to new and complex tasks. Most importantly, the time needed to program these new tasks and the expertise required, needs to be significantly reduced to enable cost effective usage of these complicated tools, as currently the costs for the software outweighs those of the hardware.

As discussed in Chapter 2, new collaborative robots are easier to use and the teaching time is less time-expensive. But what if the task to be taught needs the integration of one or more sensors external to the robot? For example it could be needed a vision system that identifies the work-piece, or the integration of a force sensor for a task where the robot must apply some specified forces. Depending on the task, the easy teaching methods that come out of the box with collaborative robots could not be enough.

One approach to achieve easy robot programming is Programming by Demonstration (PbD) as previously discussed in chapter 3. Robots are programmed by letting the user demonstrate the task to the robot rather than programming it explicitly, therefore reducing the required robotics knowledge significantly, especially for inexperienced users. After observing the user, the robot adapts the observed movements to its own constraints (like number of joints) and reproduces them. This allows not only to skip calibration and knowledge issues but also allows the robot to learn from user experience which can be very important in the productions of specialized goods where small differences in

handling will result in a superior product. But to create a good application and actually harness the power of this approach, the PbD process needs to be well designed, easy to use and most importantly has to reproduce the intended task as accurately as possible. While the idea behind PbD is simple, many

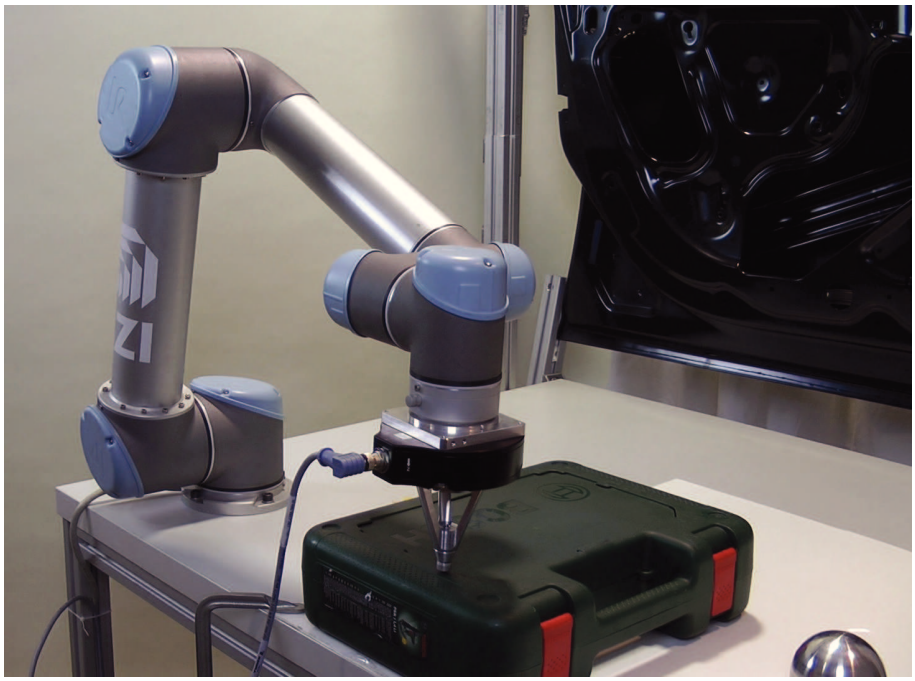


Figure 4.1: Experimental evaluation of new haptic trajectory exploration for the force-based PbD approach with a robot arm, force sensor and test work-pieces.

implementations currently available often lack important features that enable a wide range use or are still time consuming when teaching non trivial tasks. To be successful in real world applications there are some key challenges that a robotic system needs to address:

- *Self-adaptation of trajectories:* As it is difficult to teach a perfect trajectory, the robot should assist the user by adapting to imperfections and work-piece structure.
- *Surface Exploration of trajectory:* It should be sufficient to just show the basic task and let the robot explore unknown objects and how to execute the trajectory by itself.
- *Online user modification:* It must be possible to adapt the work-flow easily and change a robot program in an intuitive way and without the need to re-teach everything.

- *Force-based tasks:* Facilitating a force control on a standard robot arm increases the number of possible tasks, making the robot more flexible and valuable.
- *Intuitive shared programming:* Programming a robot needs to combine the benefits of human skill and robotic precision in the most intuitive way to be successful.

Therefore it is here proposed a new approach for haptic trajectory exploration for force-based programming by demonstration consisting of a three step teach in process:

- *Demonstration* of the desired trajectory is done in an easy and fast way. By teaching a few initial support points above the object, the overall path is defined.
- *Haptic trajectory exploration:* the robot quickly explores the surface of the object in an automated way, without the need to manually teach additional support points.
- *Shared adaptation:* creates an executable trajectory along the path for the robot, which the user can easily refine to fit the needs of the applications and assist with very complex parts.

This novel force-based approach is implemented using an off the shelve low cost, light weight robot arm with an attached force sensor, making it transferable to almost any robot arm that offers a low level control interface (see Fig. 4.1). Enabling force-based execution and haptic exploration not only provides a much easier teaching experience that can be used with almost no training but also allows to perform complex robot tasks that are impossible with classical teach-pendant programming. The proposed system can be easily added to new or existing set-ups and could greatly lower the time and therefore costs it takes to use robots in difficult tasks.

As explained in Chapter 2, the most obvious and common way of teaching something to a robot is using its dedicated teach-pendant; this allows the operator to teach either segments of a path or support points which are to be followed. This method is less complex than programming but still requires some expertise of the operator making it a specialist job that is very time consuming. Complex paths that follow surfaces closely need a very high number of support points and can take several hours to days to produce good results. Generating the paths from CAD models [37] is a very good way to produce accurate paths, even along complex surfaces. Exporting from CAD programs and using it on robots however, is far from a standard procedure. But, even if it is possible, it will usually require fine tuning and calibration by an expert, as the

real scenario of the application always presents some small differences from the CAD model. New industrial controllers have the capability of integrating some sensors, like a force/torque sensor, to close the inner control loop [38]. In this way the operator can move the robot along the whole trajectory using a robot lead-through approach, while the robot is constrained on the work surface with some force/impedance control. This approach is still very constrained on the kind of robot used and the operator must manually move the robot through the whole trajectory.

Probabilistic PbD is able to memorize and encode also force profiles [15], but probabilistic approaches are working well especially for tasks where low precision is required. Other approaches [39] are based on programming the robot through some movement primitives and make use of kinesthetic teaching which extends the capabilities of new collaborative robots. While the current approaches are promising, they still have many issues that inhibit widespread use.

Teaching itself is often counter intuitive, as the worker has to move the robot arm precisely along the desired path while respecting the robots needs for orientation or movement range. Different approaches where the worker is observed with a camera lower these restrictions but introduce the complexity of interpretation which leads to very few use cases where it can be used effectively. Finally, almost all approaches are relying on position based control. While this is sufficient for simple pick&place tasks, they are reaching their limit when the task becomes more demanding as it is the case in SME productions.

4.1 Approach

The only thing needed to build up such an application is a robot arm with low level control capabilities and a force torque sensor: in this case a Universal Robot UR5 [40] and a Robotiq Force/Torque sensor [41] have been used. Fig. 4.2 represents the hardware scheme of the project: the application is running on an external hardware which is communicating at low level with the robot controller using a wrapper of the URScript programming language [42]. Even though real-time capabilities would improve the determinism of the communication between the robot controller and the external hardware, there is no strict need of that, and the external hardware used for the project is a standard PC, which sends the target joint velocity to the controller each 8 milliseconds. The force/torque sensor communicates too with the PC, updating the force signals each 10 milliseconds.

The communications between the PC and the controller and between the PC and the sensor are managed using the ROS (Robot Operating System) middle-ware [6]. From a software point of view, 3 ROS nodes are running on

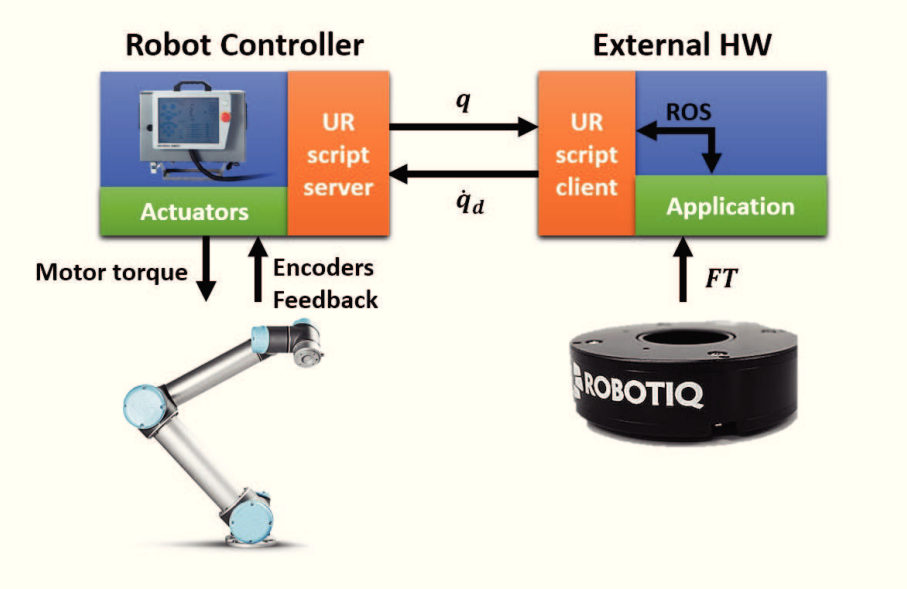


Figure 4.2: Hardware scheme of the work: the robot controller communicates with the URScript programming language with the external hardware, sending to it the encoders feedback.

the PC:

- **FT_sensor:** this *node* is in charge of communicating through USB with the Force/Torque sensor, reading the signals of the sensor at a frequency of 100 Hz and publishing all data on a *topic*, making the signals available for the other *nodes*.
- **UR_client:** this *node* contains the wrapper of the URScript programming language and runs the UR Client which communicates with the UR Server which is running on the controller. This *node* receives the joint positions from the encoders of the robot and publishes them on a *topic*; moreover the *node* reads from another *topic* the commanded joint velocity to send to the robot and communicates it to the controller using a URScript command.
- **HL_application:** the main application, comprehensive of the user interface from keyboard. This *node* implements the robot direct and inverse kinematics (using the standard convention of Denavit-Hartenberg) and all the control laws, trajectory interpolation and motion planning functions. The *node* reads the force/torque sensor and the encoders feedbacks respectively published by the *FT_sensor* and *UR_client nodes* and publishes the command velocity for the robot controller.

Regarding the motion planning and control of the robot, the main application node is running two loops on two different threads: the **control loop** and the **planning loop**. The first one is in charge of continuously publish the command joint velocity at a frequency of 125 Hz, even when the robot stands still. In this way the robot controller is always receiving a command at a precise frequency, without delays that could come from the motion planning, depending on the performances of the PC which is used. In order to use the built-in Universal Robot manual guidance from the robot's teach pendant, the user can disable the control loop from the user interface: otherwise the zero-velocity command would prevail on the UR manual guidance. Once the user has finished to use the robot manual guidance, it can activate again the control loop through the user-interface.

The planning loop is instead in charge of computing the command joint velocity deriving from the control laws or from the trajectory planning (in case of a point-to-point movement) and to update the control loop accordingly. For a point-to-point movement a trapezoidal velocity profile is used, as described in [43] and show in Fig. 4.3.

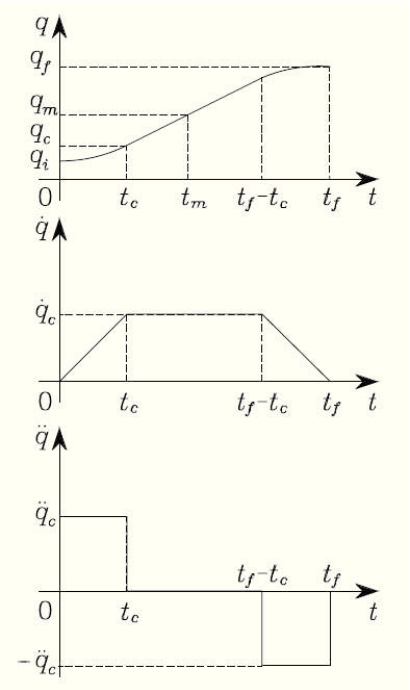


Figure 4.3: Characterization of a timing law with trapezoidal velocity profile in terms of position, velocity and acceleration.

Usually, traditional teaching of trajectories is done by setting the robot in a

learning mode and moving it along the desired path. Collaborative robots can be moved freely by the operator, but still it is a very challenging task to follow a desired path exactly. The task of keeping in contact with the workpiece, especially if it is curved or oddly shaped, i. e. for gluing applications, is nearly impossible with current techniques. To solve this issue it is here used a different approach: let the robot determine the surface structure of the object by itself. By learning the surface of the object, trajectories can be created for complex shapes while keeping constant contact with almost no effort.

Force control of robots has been studied for many years [44] and many algorithms and solutions have been developed [21]. Many of these approaches make use of the dynamic model of the robot, which is, speaking of industrial robots, usually difficult to obtain, even if some methods allow to identify it [45]; even more they are difficult to use, since most of industrial robots are position or velocity controlled. However it is possible to implement force control closing the loop at kinematic level [46], obtaining reasonable results. Force control implemented at kinematic level has though some drawbacks: in order to obtain a stable behaviour of the control, the loop must be closed at an high frequency (preferably something more than 100 Hz). Due to the fact that a robot is a rigid and non-compliant system, it means that during contact a small displacement of the robot end-effector creates a big displacement of force, which can simply modelled as:

$$\Delta F = K_e \Delta x,$$

where F is the force, x is the robot end-effector position and K_e is the impedance of the robot, here simplified as a spring. The impedance of a robot is usually very high as it is depending on the internal control parameters and on the mechanical impedance of the robot arm. On the other hand using a compliant end-effector would mean a loss of precision.

In most tasks it is necessary to move the robot end-effector using force control along some directions and position control along other directions, resulting in a hybrid position/force control [25]; this approach can also be used to realize surface exploration [47]. Haptic exploration to discover object surface and shape has been used mostly in combination with robotic hands, in order to make the robot aware of the manipulated objects [48]. Other approaches make use of some visual feedback to have knowledge of the surface in advance [49], or make use of particular end-effectors developed for surface exploration like in [50] and [51]. In the presented approach a rigid but almost frictionless teaching end-effector has been designed and used, mounted on the force/torque sensor which is on the robot wrist. The end-effector terminates with a commercial ball transfer unit (see Fig. 4.4), which lowers the friction contribution. The shape and the size of the end-effector has been designed so to make ergonomic the human-robot interaction.

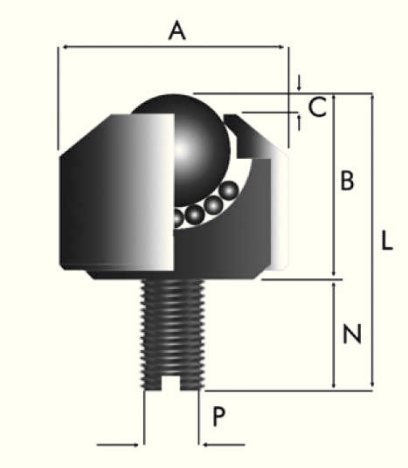


Figure 4.4: Representation and functioning of the used commercial ball transfer unit.

The learning process of the proposed approach can be divided into three main phases: Demonstration, Haptic Trajectory Exploration and Shared Adaptation. Fig. 4.5 shows the overall work-flow of the process. During the demonstration phase the operator guides the robot through some support points of the task and subsequently configures the task through a user interface, specifying the constraints that the robot must keep between a point and another. During the second phase the robot executes the task for the first time; it is assisted by the user and collects information about the trajectory. In the third phase the surface knowledge gathered in the second phase is used and the operator can further adjust the trajectory of the robot interacting physically with it, while it is executing the task. After those three phases the final task is learned and ready to be executed. This approach guarantees that the operator must have knowledge about the task, i.e. he/she must be able to know relevant / key points to teach to the robot. On the other hand, no knowledge in robotics or programming is needed.

4.1.1 Demonstration

In the first phase of the learning process the operator takes the robot and moves it to the support points of the trajectory using a manual guidance approach.

It doesn't make a difference if the operator uses the teach panel, a collaborative robot or a custom implementation using the force/torque sensor signals as input, as described in Chapter 3 [52]. The operator can subsequently configure the task to specify the kind of interaction that the robot must have with the en-

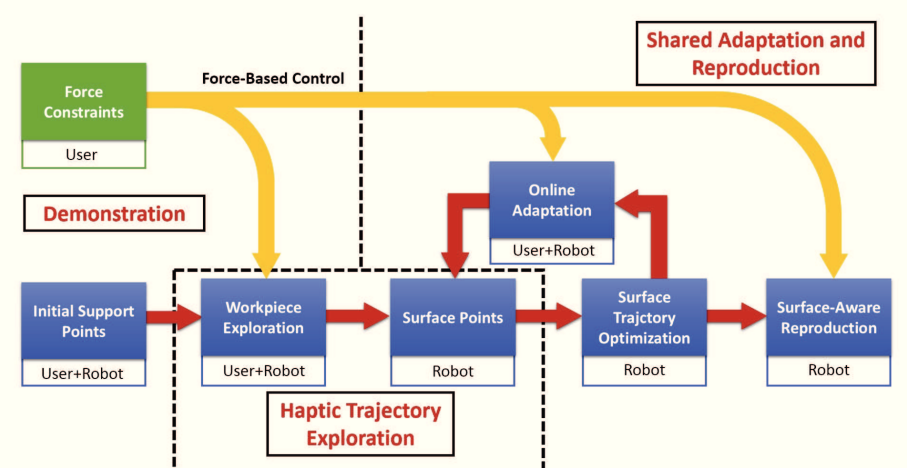


Figure 4.5: Overall work-flow scheme. Under each box is indicated who is involved in the action; the red arrows represent the work-flow direction, while the yellow arrows point to the phases which are influenced by the force control.

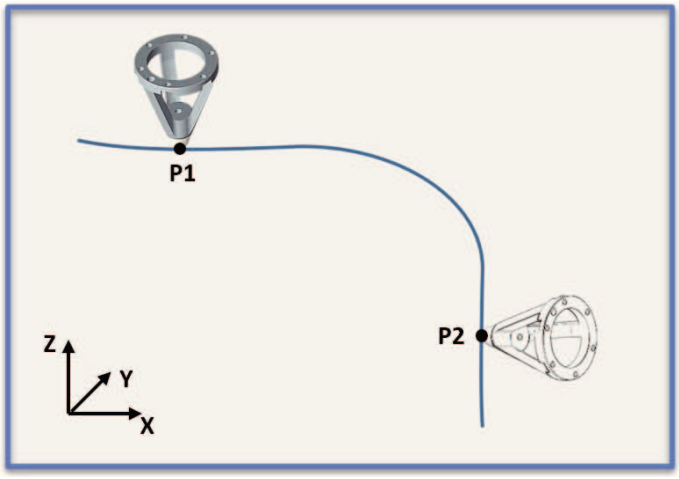


Figure 4.6: Example of task: the operator takes two initial support points, in this case start (P1) and end point (P2).

vironment. The following parameters have to be configured for each "trajectory segment" (i.e. the trajectory between two support points):

- **Force:** the force that the robot should push with the end-effector. If this is set to zero, the movement will just be a point-to-point movement without force control.

- **Time:** the execution time for the movement in the task reproduction. The time constraint will not be used during teaching steps, but only during the final reproduction of the task.

- **X/Y orientation:** specifies if the robot should adjust the orientation or keep constant orientation around X/Y axis during the first learning phase. For some kind of tasks it could be useful to fix the orientation around a certain axis, if it is known a-priori that the orientation must be kept constant.

At this point the robot is ready to learn the task, discovering by itself the trajectory to execute.

4.1.2 Haptic trajectory exploration / learning phase

In this phase the robot explores the task for the first time: the end-effector will move at a constant velocity in a direction computed as the tangent to the object surface, which is laying in the plane composed by the normal vector to the surface and the straight direction to the final position, defining the task frame as in Fig. 4.8.

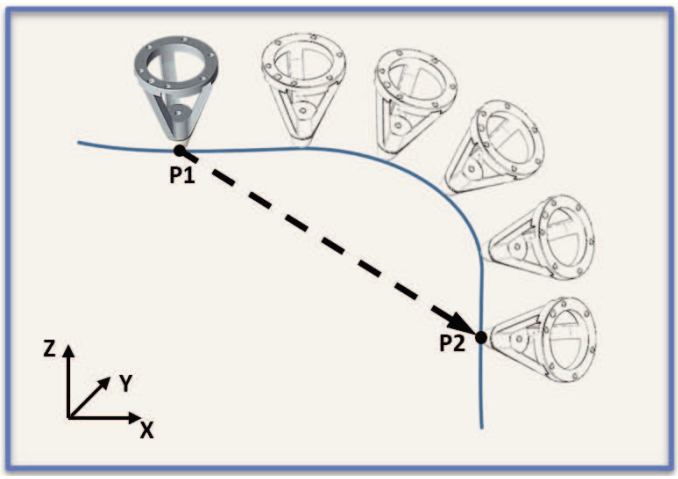


Figure 4.7: Example of task: the robot will try to go in straight direction from the starting point (P1) to the ending point (P2), but it will actually adapt to the surface it finds; the operator can help the robot in this part correcting the tool orientation as desired.

The raw force signal must be gravity-compensated and filtered, in order to make the robot aware of the surface contact and of the user inputs. Thus, the force used in all computations is a low-pass filtered version of the sensor signal,

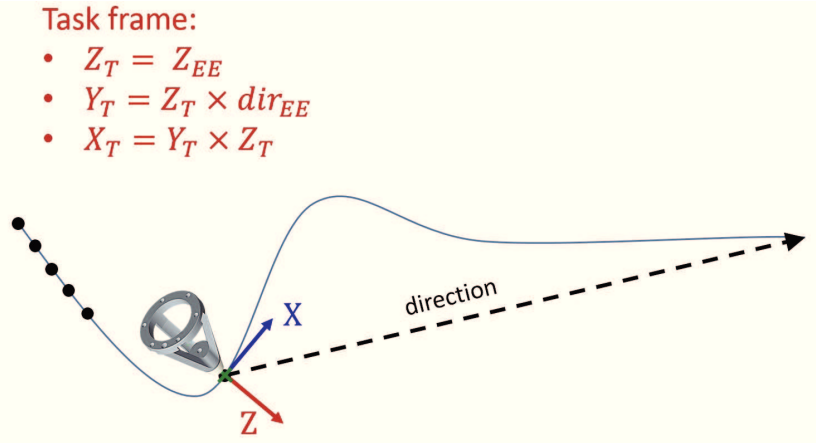


Figure 4.8: Computation of Task Frame.

in order to have a more stable behaviour:

$$F_{low_k} = F_{low_{k-1}} + K_f(F_{raw_k} - F_{low_{k-1}})\delta t, \tag{4.1}$$

where F_{raw_k} is the raw force signal of the sensor at the time instant k, K_f is a positive gain and δt is the control loop time. The gravity compensation is implemented as described in section 3.2.1.

Along the z direction of the task frame the robot will move according to a force control which keeps the robot end-effector in contact with the object surface. The force control is implemented as follows:

$$v_k = v_{k-1} - (K_a(F_{des} - F_{low_k}) + K_s v_{k-1})\delta t, \tag{4.2}$$

where v is the end-effector velocity, F_{des} is the desired force specified in the configuration step, F_{low} is the filtered force signal and K_a and K_s are two positive gains. The end-effector of the robot will move along the z direction accelerating and decelerating accordingly to the force error, while the term proportional to the velocity will act as a damping, giving as a result a more stable control behaviour.

During the first surface exploration, while the robot is moving, the user can adjust the orientation applying forces directly to the robot end-effector. Theoretically the torque signals of the sensor could be used for the same purpose, but the easiest way to apply a torque respect to the sensor it to apply forces on the robot TCP, which could result unnatural. It is thus preferable to use the force signal, so that the user can apply forces on the whole end-effector, instead that just on the TCP, making it easier for the user to move the robot

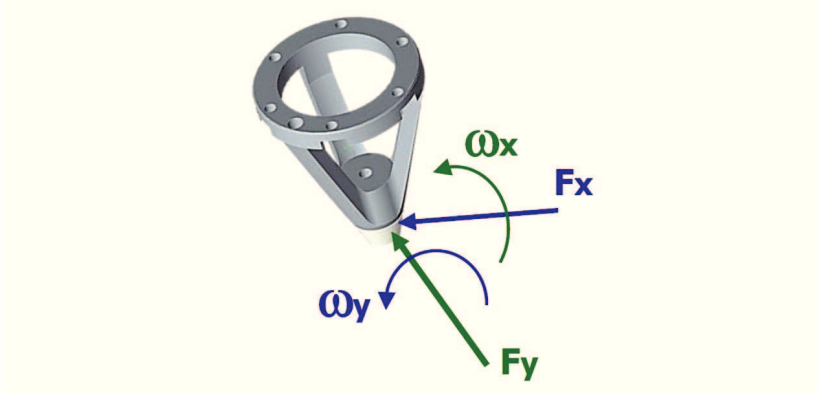


Figure 4.9: Logic of orientation control: applying a force on the x axis of the end-effector frame, will produce a movement around the y axis; vice-versa, a force on the y axis will produce a movement around the x axis.

accordingly to his/her will. In order to accomplish that, a force along x axis will move the end-effector around y axis and vice-versa, as showed in Fig. 4.9. Depending on the configuration parameters *X orientation* and *Y orientation*, the control around the corresponding axis will be activated and the orientation will change according to the following control law:

$$\omega_k = \omega_{k-1} + (K_a F_{low_k} - K_s \omega_{k-1})\delta t, \tag{4.3}$$

where ω is the end-effector orientation velocity and the other symbols have known meaning. Of course this kind of control allows the user to also apply forces along a direction which has components both along x and y axis.

Another advantage of using the force signals to control the robot orientation is the following: supposing to have a perfect force sensor and a control loop fast enough, in absence of external inputs, this control would maintain the end-effector always normal to the work surface. Since the force signal must be filtered as in equation (4.1) and the control loop is quite slow (100 Hz), this control would not be able to keep up with all surface changes. Thus, the user is able to modify the end-effector orientation applying forces to the end-effector itself. Finally, the designed end-effector which ends with a ball transfer unit, makes the friction negligible, which would otherwise interfere with the orientation control during the learning phase.

After the work-piece exploration is complete, the robot data acquired is used to estimate the trajectory surface profile. A b-spline fitting is used, obtaining a smooth trajectory which also filters the high frequency noise of the robot data. The spline fitting is then used as a feed-forward for the following step, giving

more stability to the force control. In order to compute the spline fitting the ALGLIB open source library has been used [53].

4.1.3 Shared adaptation

For complex tasks where irregular contours should be followed, it would be annoying to take many points to adapt to the surface. In this final step the user can adjust the end-effector position on the plane tangent to the surface with an impedance control. Moreover, in this learning phase the robot executes again the task at a constant end-effector speed, but now using the spline generated at the end of the previous step as a feed-forward both for position and orientation. Regarding the force control along the normal direction, equation (4.2) is adapted to use the feed-forward:

$$v_k = v_{ff_k} - (K_a(F_{des} - F_{low_k}) + K_s v_{k-1})\delta t, \tag{4.4}$$

where v_{ff_k} is the velocity feed-forward at time instant k computed from the spline fitting. In other words the robot will now repeat what it learned during the previous phase, but always mantling the desired contact force using the force control with velocity feed-forward above.

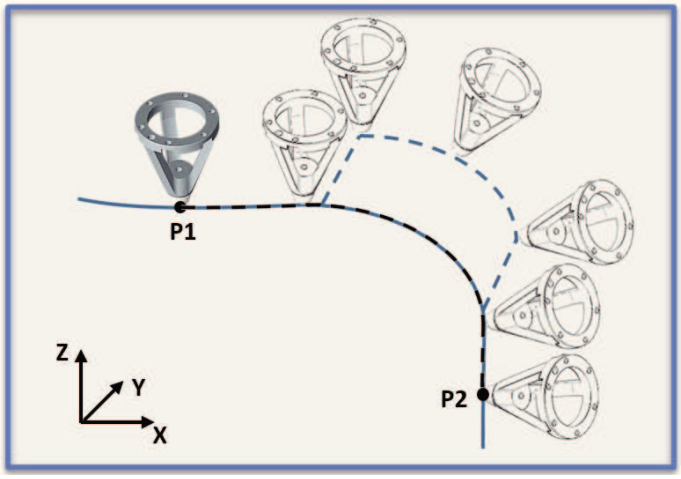


Figure 4.10: Example of task: the robot executes the trajectory learned in the previous phase; this time the operator can apply forces on the end-effector to modify the path.

The user can now also use the force signal input to modify with an impedance control the end-effector position:

$$v_k = v_{ff_k} + (K_a F_{low_k} + K_p(x_{des} - x_k) - K_s v_{k-1})\delta t, \tag{4.5}$$

where x_{des} is the nominal end-effector position for time instant k coming from the spline feed-forward, x is the end-effector actual position and K_p is a proportional gain.

This control will be active only on the y axis of the task frame (Fig. 4.8), while on the forward direction (x axis of the task frame) the user can apply a force in order to temporarily stop the robot progress. Once the robot temporarily stops to move in the forward direction, the user is still able to modify the robot path using the above impedance control: in that way it is possible to modify the robot path also with squared angles.

At the end of this learning phase a new spline fitting of the robot path is done, incorporating the modifications that the user applied during the last phase. At this point one could iteratively continue to refine the path in position and orientation repeating the previous steps, but for most tasks a single iteration is usually sufficient. In the reproduction the robot will not move any more at a constant speed along the forward direction, but the trajectory generated using the spline fitting, which will incorporate all the modification done by the user, will be computed with a trapezoidal velocity profile and accordingly to the time parameter specified by the user. Along the normal axis the robot will always move according to the force control of equation (4.4), so to adapt to the small imperfections of the surface without losing the contact or applying a force different from the desired one. Even though the robot end-effector will move faster during reproduction the feed-forward computed from the spline fitting will maintain the force control always stable.

4.2 Results

All the experiments that have been done, were made taking (and configuring) the first point and the last point of the task not in contact, while all the other points in between where taken in contact. In these cases the robot will first move with a point to point motion from the starting point to a position which is 5 mm detached from the first contact point (the direction normal to the surface is computed considering the end-effector position in the first contact point); after that, the robot will execute the contact part of task with a force control motion and at the end of the task it will move up in the normal direction before moving to the end position, in order to avoid to scratch the working surface while moving. Fig. 4.11 shows this concept: this strategy for approaching/detaching is always used, regardless of if it is a learning phase or a reproduction phase. For the first learning phase, the end-effecor moves in the X direction of the task frame (Fig. 4.8) with a constant speed of 5 mm/s.

The plots that are presented in this section concern only the force control motion part of Fig. 4.11, i.e. the contact part of the task. As said before, the

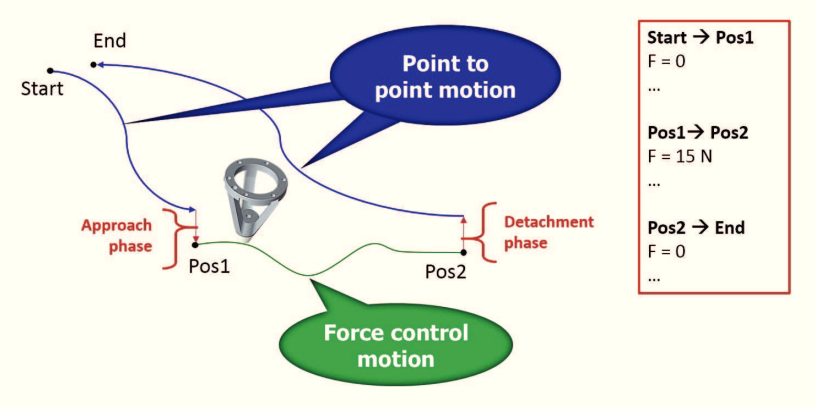


Figure 4.11: Task execution example. The table on the right represents an example of task configuration, while the picture on the left represents the movements the robot will do accordingly to that.

bottleneck for this application is the control loop frequency: in particular the force/torque sensor that is here used has an update rate of 100 Hz. Of course this application would benefit from an higher control frequency, which would make the force and impedance controls more stable. Anyhow, it is not always possible to obtain access to high-frequency control of a robot (such as 1 kHz); moreover the human-robot interaction strategies exploited in this application make that prerequisite not strictly mandatory.

As for the manual guidance algorithm, the tuning of the parameters and of the gains has been done experimentally. The first experiment which is here presented is a task on a plastic box, which is quite a challenging material due to its irregular surface. Fig. 4.12 shows the support points which were taken on the box for the task.

The challenging part of this task consists of both the irregular surface and the two uphill and downhill steps that the robot end-effector must face. During the first learning phase, the orientation of the robot end-effector is adjusted by the user to maintain it normal to the surface. Fig. 4.13 shows the force signal acquired by the force sensor and the filtered version used for the control.

The desired force that the robot should press while in contact is $F = 17.5N$: if the robot could stabilize on a still-standing position the force would get to that precise value and remain constant at steady-state, but since the robot end-effector keeps moving along the forward direction, the force control is not able to totally stabilize the signal. Still it is able to keep the force error limited, despite the disturbances coming from the surface and the user. In the first 15 seconds of the plot it can be noticed that the user pushes along the forward direction in order to adapt the end-effector orientation to the surface, while

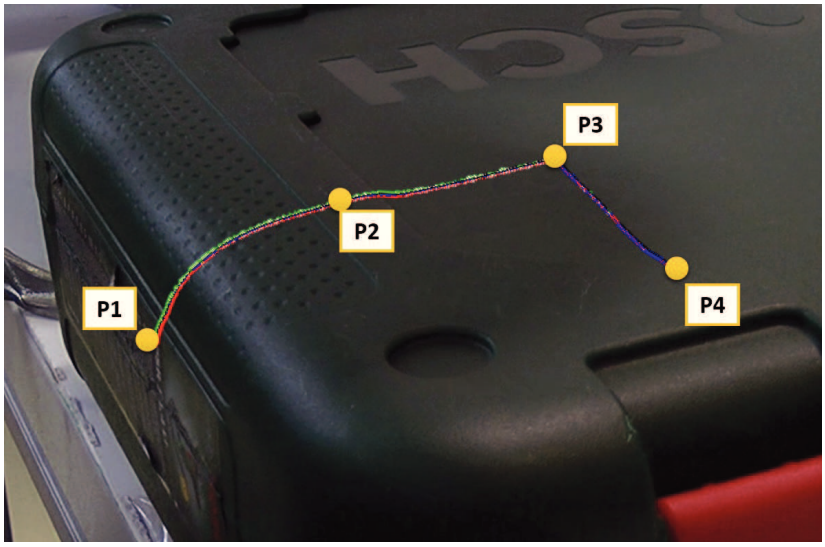


Figure 4.12: Box task with 4 support points in contact. The superimposed trajectory is the resulting spline fitting after the first learning phase.

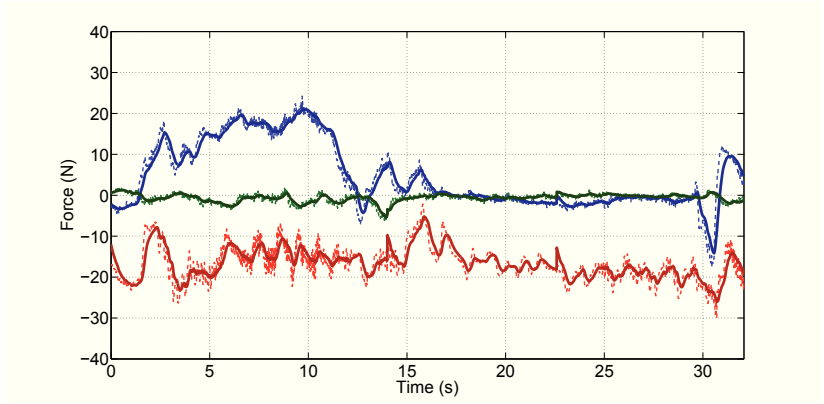


Figure 4.13: Force signal during first learning phase of the box task, along X, Y and Z axis of task frame, respectively in blue, green and red. The thin dashed lines represent the raw signal obtained from the force sensor, while the thick lines represent the filtered force signal used for the control.

after 30 seconds it can be noticed that the robot end-effector encounters the resistance of the uphill step, and is helped again by the user to climb it with the right orientation.

During the second learning phase the user modifies the path executed by the robot applying some forces to the end-effector and trying to do some squared

path. Fig. 4.14 shows the result obtained during the second learning phase and the reproduction, which is smoothed thanks to the effect of the spline fitting, filtering the imprecise movements of the user.

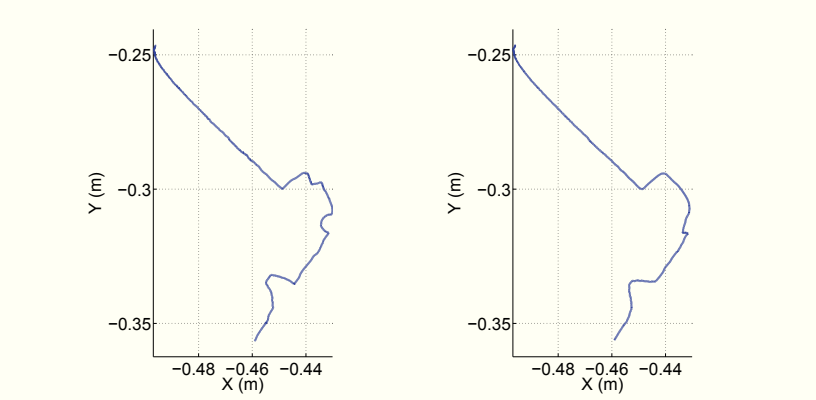


Figure 4.14: 3D plots of the second learning phase (left) and reproduction (right) of the box task, viewed from above.

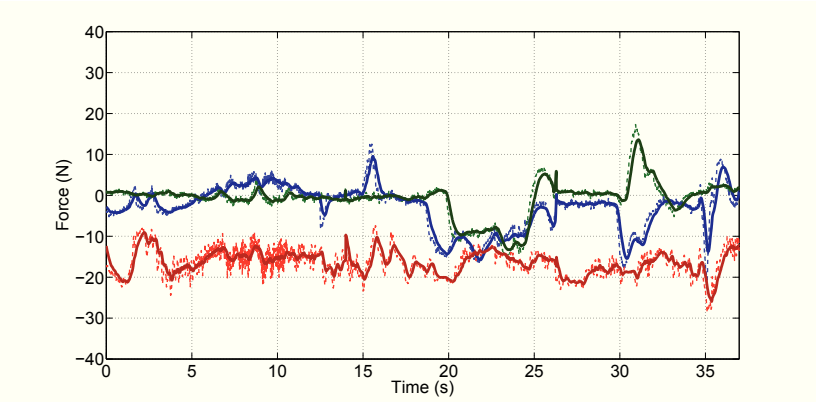


Figure 4.15: Force signal during second learning phase of the box task, along X, Y and Z axis of task frame, respectively in blue, green and red. The thin dashed lines represent the raw signal obtained from the force sensor, while the thick lines represent the filtered force signal used for the control.

Fig. 4.15 shows that the force keeps stable also during the second learning phase. Moreover the final time is greater respect to the first learning phase, because the user extends the path that the robot must do. Finally in Fig. 4.17 it can be noticed that the final time is now shorter, i.e. the robot is using for the reproduction the time specified by the user during the configuration phase.

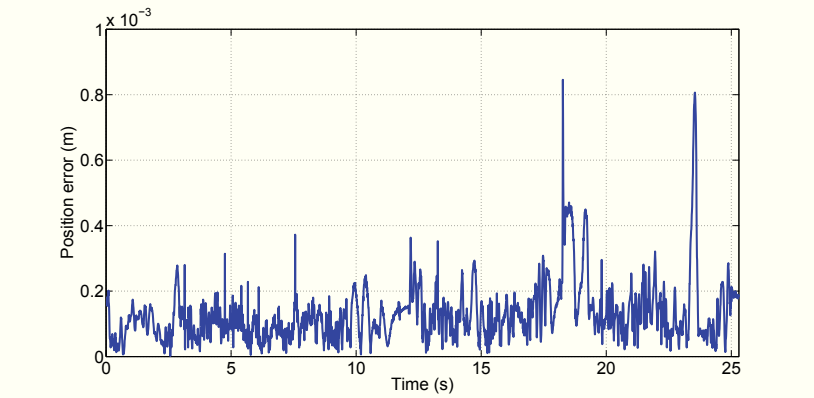


Figure 4.16: Position error during reproduction of task. The error is computed as the norm of the difference between the expected position (computed from the spline fitting) and the actual one during reproduction.

Fig. 4.16 represents the position error between the end-effector position during reproduction phase and the spline fitting generated after the second learning phase: it can be noticed that the error is always smaller than 1 mm, meaning that the robot is following accurately the new path generated, while always keeping the force constraint.

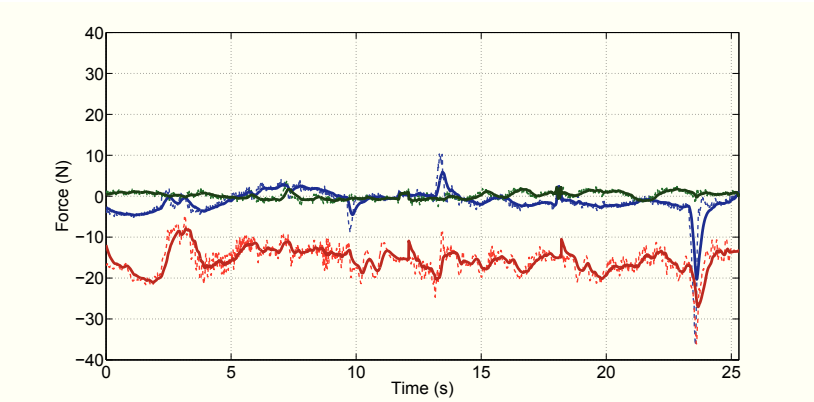


Figure 4.17: Force signal during reproduction phase of the box task, along X, Y and Z axis of task frame, respectively in blue, green and red. The thin dashed lines represent the raw signal obtained from the force sensor, while the thick lines represent the filtered force signal used for the control.

The second learning phase can be very useful for certain tasks where irregular

contours must be followed. Fig. 4.18 is an example where it is needed to push the end-effector of the robot in order to make it follow the real contour of the object, while without this feature the robot would move just in straight line as in the figure.

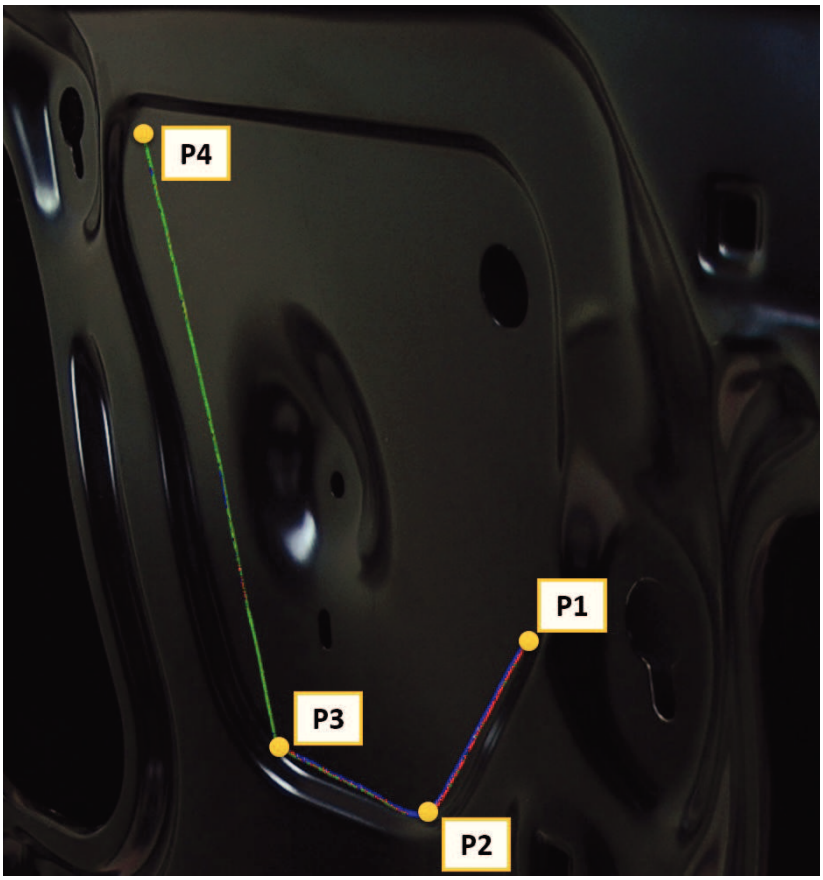


Figure 4.18: Door task with 4 support points. The superimposed trajectory is the resulting spline fitting after the first learning trajectory.

This second task that is here presented is an example of application, where e.g. a gluing task could be needed on a car door. Here the objective is to make the robot follow the contour of the inner part of the door, using 4 support points. Again, the desired force that the robot should press while in contact is $F = 17.5N$. In that case, during the first learning phase, no big changes in orientation is needed, so the user can let the robot adjust its orientation automatically. This is showed in Fig. 4.19, where the force along x and y directions is always quite small, except for the discontinuity that is present at time 15 seconds, which is generated by an imperfection of the surface.

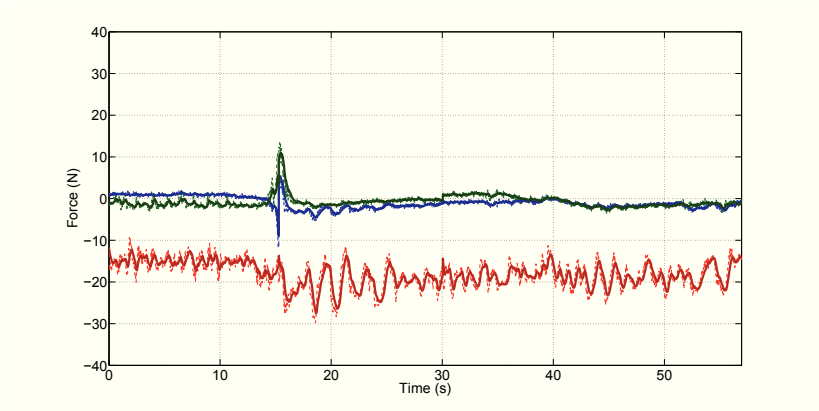


Figure 4.19: Force signal during first learning phase of the box task, along X, Y and Z axis of task frame, respectively in blue, green and red. The thin dashed lines represent the raw signal obtained from the force sensor, while the thick lines represent the filtered force signal used for the control.

During the second learning phase the user interacts with the robot, pushing the end-effector towards the border of the inner part of the door, modifying the trajectory between points P2 and P3 and between points P3 and P4. Fig. 4.20 shows the 3D path of the trajectory during the second learning phase and during the reproduction phase. Again it can be noticed that the path during the reproduction is smoother, thanks to the spline fitting.

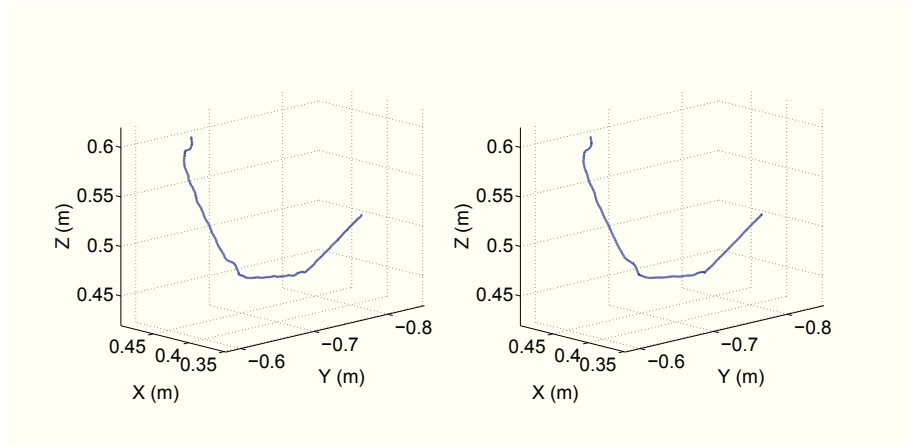


Figure 4.20: 3D plots of the second learning phase (left) and reproduction (right) of the door task, frontal view.

Fig. 4.21 shows the corresponding forces of the second learning phase. Again,

it is important to underline how this phase enables the user to easily teach a complex surface to the robot, which would otherwise result in a big number of support points.

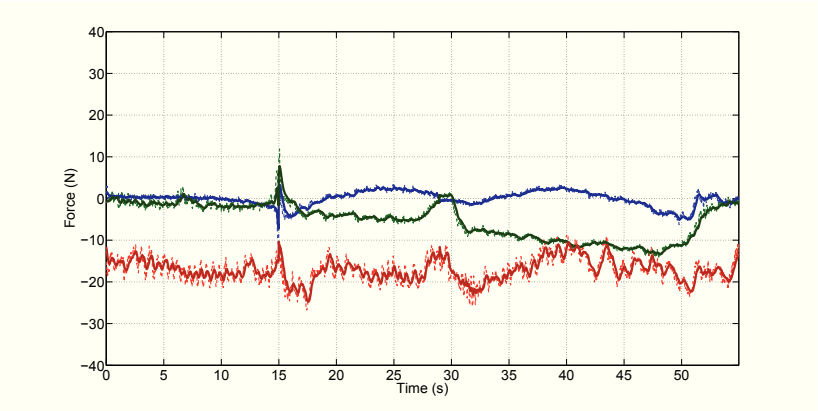


Figure 4.21: Force signal during second learning phase of the door task, along X, Y and Z axis of task frame, respectively in blue, green and red. The thin dashed lines represent the raw signal obtained from the force sensor, while the thick lines represent the filtered force signal used for the control.

Finally Fig. 4.22 shows the forces during the final reproduction of the task. Here the force along the z axis is less stable than during the learning phases, as the robot is now moving faster to respect the time constraint imposed by the user. Anyway the force control is able to keep the force error quite limited, so that the task can be performed successfully. The discontinuity due to the imperfection of the surface, during the reproduction moves at t=10 seconds (instead of t=15 seconds), as the end-effector is moving faster during the reproduction phase.

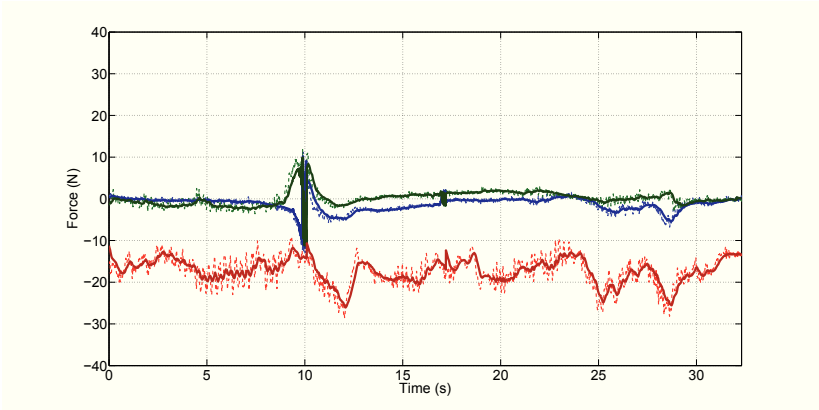


Figure 4.22: Force signal during reproduction phase of the door task, along X, Y and Z axis of task frame, respectively in blue, green and red. The thin dashed lines represent the raw signal obtained from the force sensor, while the thick lines represent the filtered force signal used for the control.

Chapter 5

Open Source Robotics

Proprietary software is nowadays still the standard in industrial robotic applications. However, the spread of robotics to other application fields like medical field, service robotics and others, is now leading to the development of open source software for robotic applications. While the main benefits are clear, to put open source robotics into practice usually needs some effort.

The work presented in Chapters 3 and 4 has been developed keeping that in mind. Both the manual guidance approach and the haptic learning approach are designed so that they can be deployed to any robot with certain characteristics with low effort. In particular what is needed is the possibility to communicate at low level with the robot controller using an external hardware, so to be able to have access to robot parameters and to control at high frequencies (up to 1 kHz) the robot motion from the external hardware.

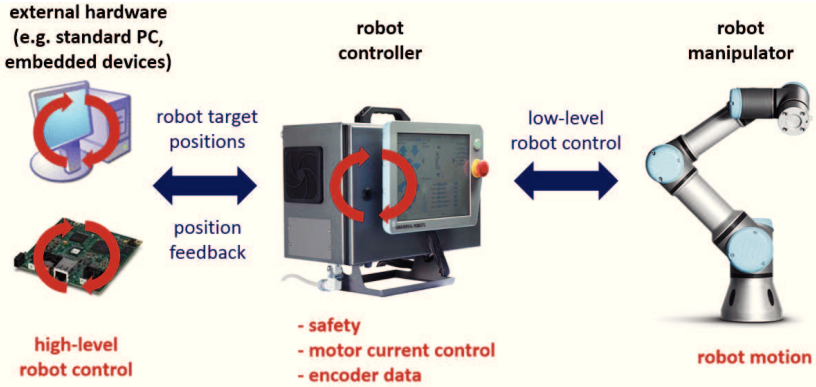


Figure 5.1: Hardware architecture for open software robotics application.

Fig. 5.1 shows this concept: the robot controller takes care of robot low level control, setting the joint currents so that the position task is respected. Moreover all the safety aspects are managed by the controller. The external hardware (which can be a standard pc or an embedded device) is instead in charge of the motion planning and the high level control of the robot. The

robot controller shall communicate in real-time to the external hardware the joint position and other parameters or signals (such as the joint currents) that may be needed for motion planning. The external hardware can in that way communicate (always in real-time) the target joint position to the robot controller.

Figure 5.2: Denso industrial robot in manual guidance control.

While the concept is simple and clear, to put into practice that kind of control it means to communicate with each robot controller in a different way, as each one has its own proprietary communication software. But once that it is done once, that approach can be used for any kind of application, and moreover, what is more important, is that any application can be used on any robot that fulfils those requirements, without the need of changing anything on the application side, as the generation of the trajectory is in that way independent from the robot model.

5.1 The manual guidance algorithm applied to an industrial robot

In order to demonstrate that concept, further experiments of the manual guidance algorithms have been carried out using a Denso 6-axis industrial robot (Denso VS-087) as in Figure 5.2. The b-CAP protocol [54] has been used to interface the controller of the robot with an external hardware. The b-CAP protocol is based on TCP/IP communication protocol and permits to control the robot using a "slave mode", i.e. the controller reads from the ethernet port

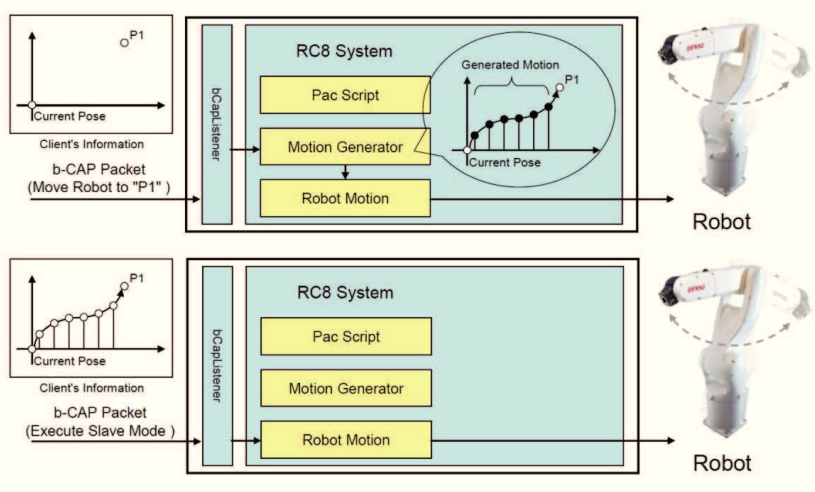


Figure 5.3: b-CAP protocol slave mode functioning. Above: normal functioning, the interpolation of trajectory is executed inside the RC8 Denso controller. Below: slave mode, the interpolation of trajectory is executed outside the RC8 Denso controller. Source: Denso manual.

a joint position each 2 milliseconds and moves the robot to that position; the controller gives the robot joint position as feedback.

Fig. 5.3 shows the functioning of b-CAP slave mode. In normal robot motion commands, RC8 controls the robot by means of generating trajectory in real time in order to achieve the target posture designated by client application. In Slave Mode instead, client specifies the robot posture in order to control robot motion in real time (real-time trajectory control by client application).

A cRIO 9068 from National Instrument running a linux realt-time OS has been used to implement the manual guidance algorithm and to send position commands to the robot controller. In this case, the joint velocity resulting from (3.9) is integrated with Euler method and sent to the controller. The aim of these experiments is to show that the method presented can be easily adapted to any kind of anthropomorphic robot. Fig. 5.4 shows the hardware configuration for those experiments: the cRio is communicating with the force/torque sensor, using the signals to generate a robot trajectory which is then communicated in real-time to the robot controller using the b-CAP protocol.

In the first experiment, the user tries to draw a circle in the xy plane, while keeping the end-effector orientation constant. Figure 5.6 shows the force applied by the operator while moving the robot and the resulting end-effector velocity, while Figure 5.5 shows the end-effector motion.

This experiment shows that the robot end-effector is easy to move also along "diagonal" directions. This depends mainly on two parameters: the threshold,

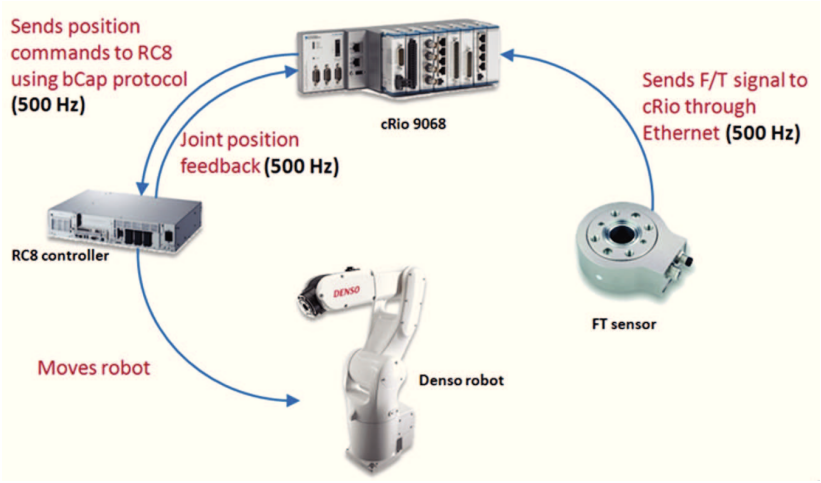


Figure 5.4: Hardware configuration for manual guidance experiments with Denso VS-087 robot.

which has to be as low as possible in order to be able to move the robot using low forces; and the K2 damping parameter (see equations 3.7 and 3.8), which has to be high enough, so to have an accelerated motion of the end-effector and not a instant velocity proportional to the force. The same is true also for the rotational part of the control, as showed by the experiment in Figure 5.7. This second experiment is the complementary of the previous one: here the user tries to do a spherical movement of the end-effector without moving the end-effector tip.

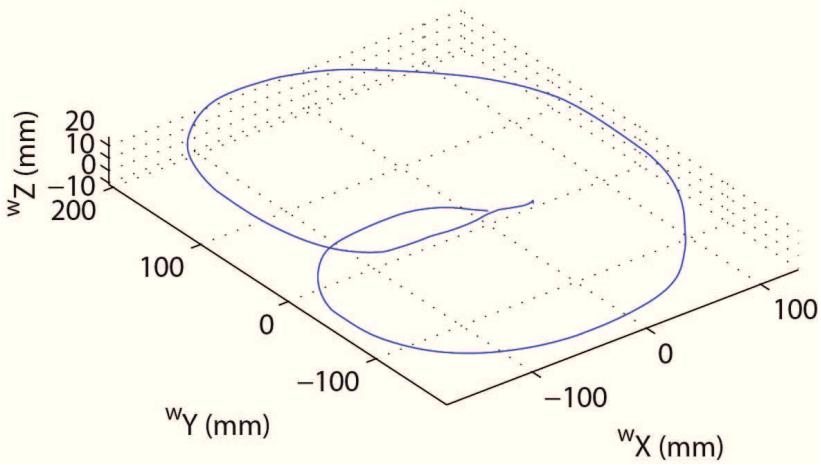


Figure 5.5: Robot guided through a circle. 3D plot of the end-effector motion.

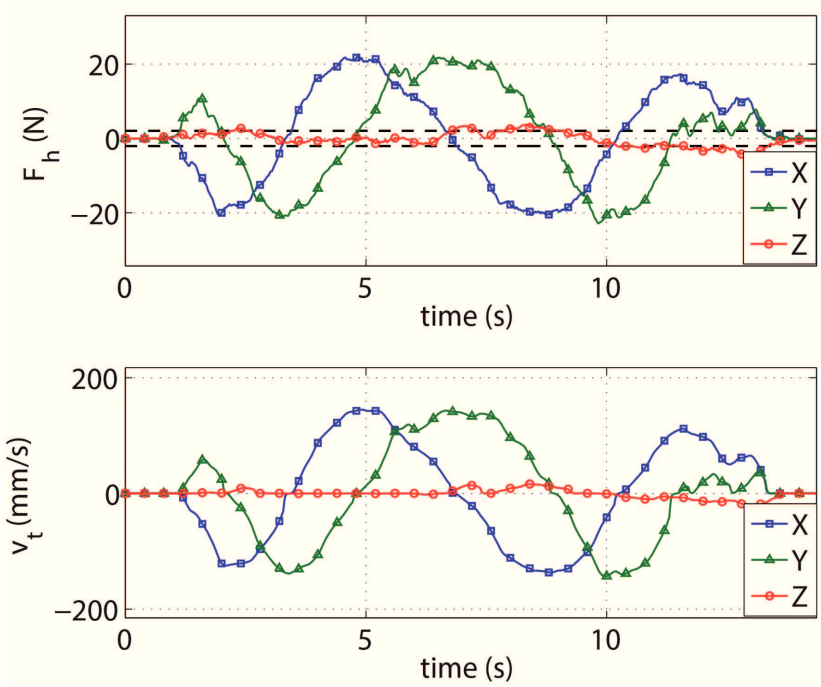


Figure 5.6: Robot guided through a circle. Above: Force applied by the human along x, y and z axis (respectively in blue, green and red), the black lines represent the threshold; bottom: The resulting end-effector translational velocity.

The results obtained with the Denso robot show how the manual guidance algorithm can be quickly deployed to other robots without big efforts. Moreover, having a control loop which runs on a real-time Operating system and at an higher frequency respect to the schunk robot (500 Hz instead of 100 Hz) makes the robot more reactive to user inputs, improving the quality of the manual guidance.

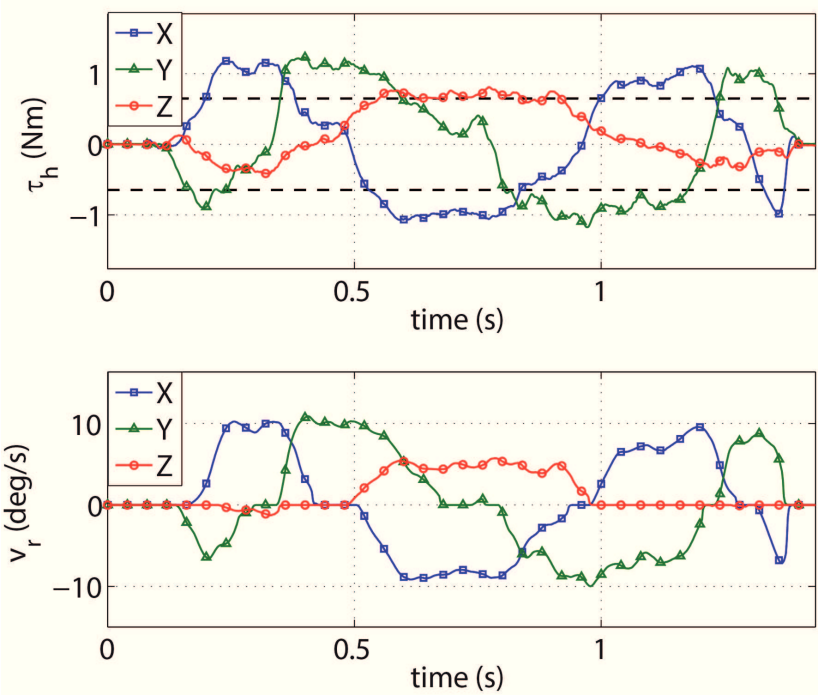


Figure 5.7: Robot guided through a spherical movement. Above: Torque applied by the human around x, y and z axis (respectively in blue, green and red), the black lines represent the threshold; bottom: The resulting end-effector rotational velocity.

Chapter 6

Conclusion

The work presented in this Ph.D. thesis shows how the new collaborative robotics is changing and is going to change the automation world and the concept of robotics itself. In the introduction a picture of the contemporary situation of robotics in industrial contest and the evolution of Industry 4.0 are presented. Furthermore the premises and the motivation of the work are explained. The second chapter introduce the innovations that collaborative robots are bringing. Besides the possibility of working sharing their workspace with humans, they can also be deployed easily and in a fast way in production line. Furthermore the programming of collaborative robots is made simple and intuitive, exploiting manual guidance methods, making them very flexible and easy adaptable for new tasks. The safety aspects deriving from ISO standards are explained and analysed. Finally an exhaustive list of the collaborative robots that are present on the market with pro and cons has been presented.

Chapter 3 proposes an algorithm for manual guidance based on the usage of a force/torque sensor, which can be applied to any robotic arm. A SoA of other methods for robot manual guidance is presented, highlighting the differences and the advantages of each on. Experimental results have been carried out to prove the stability and the functioning of the presented algorithm; moreover a realistic use case has also been presented, showing how the manual guidance can be used to teach in easy and quick way a simple task to a robot. This highlights how the manual guidance can simplify the programming, especially for those tasks which require to be adjusted or updated often, due to for example new models to be tested.

The fourth chapter addresses the problem of teaching complex tasks that require the integration of external sensors with the robot (like a force/torque sensors), keeping the programming of the task simple and time-saving. The innovative proposed approach consists of three phases:

- Demonstration of the trajectory support points using manual guidance (or other methods) and definition of the constraint through the user interface.
- Learning phase, where the robot discovers itself the task, based on the

inputs that the user gave it during the demonstration. While the robot is moving along the task, the user can modify the orientation or the position of the robot end-effector to adjust the task to his will. To that purpose, this phase can be repeated more than once.

- Reproduction of task, where the robot executes in autonomous way what learned in the previous phases.

With such an approach task along complex surfaces can be taught easily and quickly as demonstrated with the experiment presented in this work.

Finally in the last chapter, the importance of open source software for robotics has been highlighted. As the Cyber-Physical Systems are spreading thanks to Industry 4.0, the need of having reconfigurable and interchangeable modules is growing. One of the characteristics of collaborative robots is to be flexible and easily physically interchangeable, as they can be mounted quickly and without the need to have physical barriers specifically designed. But in order to be really interchangeable there is the need to make robots flexible also speaking of software. All the work here presented has been done keeping in mind that concept, i.e. all the proposed approaches where realized independently from the robot used. To demonstrate that other experiments of the manual guidance algorithm have been presented, using a Denso industrial robot arm. This Ph.D. work proposes new methods and programming strategies to enable and improve Programming by Demonstration techniques, taking care of all aspects which are relevant to industrial contest: how to consider safety aspects in the integration of a collaborative robot, flexibility of the solution and easiness of programming and usage. In addition, all the presented concept were supported by experimental results taking in consideration set-ups for realistic use-cases.

The results obtained and here presented were recognized as valid and sound internationally as confirmed by tow publications. [52] presents the manual guidance approach, while [55] shows how open source software techniques can be used to integrate a robot into a more complex system like a CPS, making an industrial robot arm flexible and adaptable on software side.

Bibliography

[1] online. (2106, Aug.) International Federation of Robotics (IFR). [Online]. Available: http://www.ifr.org/industrial-robots

[2] CPS Steering Group, "Cyber-physical systems executive summary," in CPS Summit 2008, march 2008, http://varma.ece.cmu.edu/Summit/.

[3] N. Jazdi, "Cyber physical systems in the context of industry 4.0," in *Automation, Quality and Testing, Robotics, 2014 IEEE International Conference on*, May 2014, pp. 1–4.

[4] E. Lee, "Cyber physical systems: Design challenges," in *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, May 2008, pp. 363–369.

[5] D. Gorecky, M. Schmitt, M. Loskyll, and D. Zuhlke, "Human-machine-interaction in the industry 4.0 era," in *Industrial Informatics (INDIN), 2014 12th IEEE International Conference on*, July 2014, pp. 289–294.

[6] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[7] online. (2106, Aug.) ROS Industrial. [Online]. Available: rosindustrial.org

[8] A. De Santis, B. Siciliano, A. De Luca, and A. Bicchi, "An atlas of physical human–robot interaction," *Mechanism and Machine Theory*, vol. 43, no. 3, pp. 253 – 270, 2008.

[9] "Robots and robotic devices – Safety requirements for industrial robots – Part 1: Robots," International Organization for Standardization, Norm ISO 10218-1, 2011.

[10] "Robots and robotic devices – Safety requirements for industrial robots – Part 2: Robot systems and integration," International Organization for Standardization, Norm ISO 10218-2, 2011.

[11] "Safety of machinery – positioning of safeguards with respect to the approach speeds of parts of the human body," International Organization for Standardization, Norm ISO 13855, 2010.

[12] "ISO: Robots and robotics devices – Safety requirements for industrial robots – Collaborative operation," International Organization for Standardization, Norm ISO 15066, 2016.

[13] A. De Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger, "Collision detection and safe reaction with the dlr-iii lightweight manipulator arm," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 2006, pp. 1623–1630.

[14] S. Calinon, F. D'halluin, E. Sauser, D. Caldwell, and A. Billard, "Learning and reproduction of gestures by imitation: An approach based on Hidden Markov Model and Gaussian Mixture Regression," *IEEE Robotics and Automation Magazine*, vol. 17, no. 2, pp. 44–54, 2010.

[15] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation Learning of Positional and Force Skills Demonstrated via Kinesthetic Teaching and Haptic Input," *Advanced Robotics*, vol. 25, no. 5, pp. 581–603, 2011.

[16] J. Pires, G. Veiga, and R. Araújo, "Programming-by-demonstration in the coworker scenario for smes," *Industrial Robot*, vol. 36, no. 1, pp. 73–83, 2009. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-58349119990&partnerID=40&md5=5c6781a8fd0384288177dedf9e29509b

[17] F. J. Abu-Dakka, B. Nemec, A. Kramberger, A. Glent Buch, N. Krüger, and A. Ude, "Solving peg-in-hole tasks by human demonstration and exception strategies," *Industrial Robot: An International Journal*, vol. 41, no. 6, pp. 575–584, 2014. [Online]. Available: http://dx.doi.org/10.1108/IR-07-2014-0363

[18] L. Bascetta, G. Ferretti, G. Magnani, and P. Rocco, "Walk-through programming for robotic manipulators based on admittance control," *Robotica*, vol. 31, pp. 1143–1153, 10 2013. [Online]. Available: http://journals.cambridge.org/article_S0263574713000404

[19] R. Ikeura, H. Monden, and H. Inooka, "Cooperative motion control of a robot and a human," in *Robot and Human Communication, 1994 3rd IEEE International Workshop on*, 1994, pp. 112–117.

[20] Hogan, N., "Impedance control - An approach to manipulation. I - Theory. II - Implementation. III - Applications," *ASME Transactions Journal of Dynamic Systems and Measurement Control B*, vol. 107, pp. 1–24, Mar. 1985.

[21] B. Siciliano and L. Villani, *Robot Force Control*, 1st ed. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

[22] R. Ikeura and H. Inooka, "Variable impedance control of a robot for cooperation with a human," in *Robotics and Automation, 1995 IEEE International Conference on*, vol. 3, May 1995, pp. 3097–3102.

[23] K. Tee, R. Yan, and H. Li, "Adaptive admittance control of a robot manipulator under task space constraint," in *Robotics and Automation, IEEE International Conference on*, 2010, pp. 5181–5186.

[24] P. Marayong, G. Hager, and A. Okamura, "Control methods for guidance virtual fixtures in compliant human-machine interfaces," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, 2008, pp. 1166–1172.

[25] M. Raibert and J. Craig, "Hybrid position/force control of manipulators." *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, vol. 103, no. 2, pp. 126–133, 1981.

[26] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schaeffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger, "The kuka-dlr lightweight robot arm - a new reference platform for robotics research and manufacturing," in *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, June 2010, pp. 1–8.

[27] G. Ferretti, G. Magnani, and P. Rocco, "Assigning virtual tool dynamics to an industrial robot through an admittance controller," in *Advanced Robotics, 2009. ICAR 2009. International Conference on*, 2009, pp. 1–6.

[28] M. Infante and V. Kyrki, "Usability of force-based controllers in physical human-robot interaction," in *Human-Robot Interaction (HRI), 2011 6th ACM/IEEE International Conference on*, 2011, pp. 355–362.

[29] S. Kuhn, T. Gecks, and D. Henrich, "Velocity control for safe robot guidance based on fused vision and force/torque data," in *Multisensor Fusion and Integration for Intelligent Systems, 2006 IEEE International Conference on*, 2006, pp. 485–492.

[30] K. Ohishi and H. Ohde, "Collision and force control for robot manipulator without force sensor," in *Industrial Electronics, Control and Instrumentation, 1994. IECON '94., 20th International Conference on*, vol. 2, 1994, pp. 766–771.

[31] K. S. Eom, I. Suh, W. Chung, and S.-R. Oh, "Disturbance observer based force control of robot manipulator without force sensor," in *Robotics and Automation, 1998 IEEE International Conference on*, vol. 4, 1998, pp. 3012–3017 vol.4.

[32] Z. Chen and P. Kazanzides, "Force control of a non-backdrivable robot without a force sensor," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 2013, pp. 3570–3575.

[33] M. Geravand, F. Flacco, and A. De Luca, "Human-robot physical interaction and collaboration using an industrial robot with a closed control architecture," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013, pp. 4000–4007.

[34] L. Simoni, M. Beschi, G. Legnani, and A. Visioli, "Friction modeling with temperature effects for industrial robot manipulators," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 3524–3529.

[35] F. Smith and B. Rooks, "The harmonious robot," *Industrial Robot: An International Journal*, vol. 33, no. 2, pp. 125–130, 2006. [Online]. Available: http://dx.doi.org/10.1108/01439910610651446

[36] S. Pieska, J. Kaarela, and O. Saukko, "Towards easier human-robot interaction to help inexperienced operators in smes," in *IEEE 3rd International Conference on Cognitive Infocommunications (CogInfoCom)*, Dec 2012, pp. 333–338.

[37] P. Neto, J. Pires, and A. Moreira, "Cad-based off-line robot programming," in *IEEE Conference on Robotics Automation and Mechatronics (RAM)*, June 2010, pp. 516–521.

[38] G. Zhang, J. Wang, and J. Ge, "Robotic path learning with graphical user interface," in *44th International Symposium on Robotics (ISR)*, Oct 2013, pp. 1–4.

[39] C. Schou, J. Damgaard, S. Bogh, and O. Madsen, "Human-robot interface for instructing industrial tasks using kinesthetic teaching," in *44th International Symposium on Robotics (ISR)*, Oct 2013, pp. 1–6.

[40] Universal Robots. (2015) Universal Robots UR5, CB2. [Online]. Available: http://www.universal-robots.com/products/ur5-robot/

[41] Robotiq. (2016) Force Torque Sensor FT 150. [Online]. Available: http://robotiq.com/products/robotics-force-torque-sensor/

[42] Universal Robots, "The URScript Programming Language," Version 3.2, April 2016, https://www.universal-robots.com/download/.

[43] B. Siciliano, L. Sciavicco, and L. Villani, *Robotics: Modelling, Planning and Control*, ser. Advanced Textbooks in Control and Signal Processing. Springer, 2009.

[44] E. Dégoulange and P. Dauchez, "External force control of an industrial puma 560 robot," *Journal of Robotic Systems*, vol. 11, no. 6, pp. 523–540, 1994.

[45] E. Villagrossi, G. Legnani, N. Pedrocchi, F. Vicentini, L. M. Tosatti, F. Abbà, and A. Bottero, "Robot dynamic model identification through excitation trajectories minimizing the correlation influence among essential parameters," in *Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics - Volume 2: ICINCO,*, 2014, pp. 475–482.

[46] A. Winkler and J. Suchy, "Position feedback in force control of industrial manipulators - an experimental comparison with basic algorithms," in *IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, Nov 2012, pp. 31–36.

[47] D. Bossert, U.-L. Ly, and J. Vagners, "Experimental evaluation of a hybrid position and force surface following algorithm for unknown surfaces," in *IEEE International Conference on Robotics and Automation, 1996. Proceedings.*, vol. 3, Apr 1996, pp. 2252–2257 vol.3.

[48] A. M. Okamura and M. R. Cutkosky, "Feature detection for haptic exploration with robotic fingers," *The International Journal of Robotics Research*, vol. 20, no. 12, pp. 925–938, 2001.

[49] H. Koch, A. Konig, A. Weigl-Seitz, K. Kleinmann, and J. Suchy, "Multisensor contour following with vision, force, and acceleration sensors for an industrial robot," *Instrumentation and Measurement, IEEE Transactions on*, vol. 62, no. 2, pp. 268–280, Feb 2013.

[50] J. Back, J. Bimbo, Y. Noh, L. Seneviratne, K. Althoefer, and H. Liu, "Control a contact sensing finger for surface haptic exploration," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 2736–2741.

[51] J. Jamisola, R.S., P. Kormushev, A. Bicchi, and D. Caldwell, "Haptic exploration of unknown surfaces with discontinuities," 2014, pp. 1255–1260.

[52] D. Massa, M. Callegari, and C. Cristalli, "Manual guidance for industrial robot programming," *Industrial Robot: An International Journal*, vol. 42, no. 5, pp. 457–465, 2015.

[53] ALGLIB Project. (2016) ALGLIB Library, Interpolation and fitting. [Online]. Available: http://www.alglib.net/interpolation/spline3.php

[54] D. W. Inc., "Specifications of b-CAP Communication." Dec. 2011. [Online]. Available: http://www.ccarafa.it/RC8/img/001511/b-CAP_Guide_en.pdf

[55] C. Cristalli, S. Boria, D. Massa, L. Lattanzi, and E. Concettoni, "A Cyber-Physical System Approach for the Design of a Modular Smart Robotic Cell," in *IEEE IECON 2016 (42th Annual Conference of the IEEE Industrial Electronic Society)*, 2016.