Intuitive, iterative and assisted virtual guides programming for human-robot comanipulation

Susana Sanchez Restrepo

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *01/02/2018* **par :**

Susana Sánchez Restrepo

INTUITIVE, ITERATIVE AND ASSISTED VIRTUAL GUIDES PROGRAMMING FOR HUMAN-ROBOT COMANIPULATION

JURY

Olivier BRUNEAU	Professeur des Universités à l'IUT de Cachan, Université Paris-Sud	Président du jury
Guillaume MOREL	Professeur à l'Université Pierre et Marie Curie	Rapporteur
Philippe FRAISSE	Professeur à l'Université de Montpellier	Rapporteur
Simon LACROIX	Directeur de recherche CNRS, LAAS/CNRS	Examinateur
Julie DUMORA	Ingénieure de Recherche en Robotique, CEA Tech	Examinatrice
Xavier LAMY	Ingénieur de Recherche en Robotique, CEA List	Encadrant
Daniel SIDOBRE	Maître de Conférences à l'Université Paul Sabatier	Directeur de thèse

---

École doctorale et spécialité :
EDSYS : Robotique 4200046
Double mention :
EDSYS : Automatique 4200046
Unité de Recherche :
Laboratoire d'analyse et d'architecture des systèmes
Directeur(s) de Thèse :
Daniel SIDOBRE et *Xavier LAMY*
Rapporteurs :
Guillaume MOREL et *Philippe FRAISSE*

a Olga y Julio,
a Sara,
a Samuel,
à Joris.

Acknowledgments

“ ... para ser cada vez menos humanos ... ”

Camilo Arango Vélez, Congreso de filosofía
Colegio San Ignacio. Medellín, 2008

Je souhaite tout d'abord remercier l'ensemble des membres du jury : les professeurs Guillaume Morel et Philippe Fraisse qui ont accepté sans hésitation le rôle de rapporteur de mes travaux de thèse, Olivier Bruneau en tant que président et Simon Lacroix et Julie Dumora en tant qu'examinateurs. J'ai beaucoup apprécié nos échanges lors de la soutenance et vos compliments resteront gravés dans ma mémoire !

Je remercie tout particulièrement mon directeur de thèse Daniel Sidobre et mon encadrant Xavier Lamy. Daniel, merci pour ta réactivité et tes conseils, notamment pour la rédaction de mes articles et du manuscrit. Merci aussi pour ton accueil si chaleureux à Toulouse et pour ta grande disponibilité. Xavier, tu as été depuis le début de la thèse une source d'inspiration pour moi et sache que j'admire fortement ton travail et la passion avec laquelle tu mènes tes projets. Merci de m'avoir guidée jusqu'à la fin... Tu demandais toujours un peu plus et cela m'a beaucoup motivée. Merci aussi pour ton soutien dans les moments difficiles, que ce soit pour me remonter le moral, travailler avec les quaternions, pour déboguer la TAO, essayer d'avoir un ordinateur portable ou alors pour faire un peu de maintenance sur les robots. Enfin, merci pour la confiance que tu m'as accordée et pour ta disponibilité même quand tu n'avais pas vraiment le temps.

Cette thèse n'aurait pas été possible sans le LRI au CEA. Je remercie donc Yann Perrot pour m'avoir accueillie au laboratoire. Je suis arrivée au LRI pour mon stage de Master Recherche sous l'encadrement de Mathieu Grossard et Nolwenn Kammerer. C'est grâce à eux que j'ai décidé de continuer dans le monde de la recherche. Mathieu, Nolwenn, j'ai beaucoup aimé travailler avec vous. Merci pour le soutien et les conseils que vous m'avez apportés, du début du stage à la fin de la thèse. Je souhaite aussi remercier Fares et Dominique de m'avoir si bien accueillie au BE... Toujours un sourire et des paroles d'encouragement ! Fares, même si notre

collaboration a été courte, c'était un plaisir de travailler avec toi sur les actionneurs à câbles. Je remercie aussi mes amis de l'atelier, et tout particulièrement Benoît qui a toujours été là pour fabriquer de belles pièces pour mes démonstrations et mes expérimentations (même si je le prévenais quelques heures avant). Et tous ces bons moments partagés à la petite pause café de l'atelier à la fin de ma thèse. Un grand merci à Elodie, la personne la plus efficace dans son travail que je n'ai jamais rencontrée. Toujours un grand sourire pour gérer les petits problèmes administratifs : quelle grande qualité humaine ! Merci Dominique Schoen pour tes compliments et ton intérêt pour ma thèse. Merci Yvan et Max pour nos échanges intéressants sur la cobotique et le Sybot. Merci Olivier David et Franck Geffard pour votre intérêt pour ma thèse et vos questions pertinentes ! Merci Franck pour m'avoir motivée à participer aux journées FéDev et aux journées de thèses à l'ENSTA. Merci Baptiste pour nos échanges et débats sur la robotique, le management et l'architecture logicielle. Merci pour avoir contribué à ma santé en me motivant pour aller courir et nager à midi. C'était un plaisir de partager tout cela avec toi. Merci Olivier Lebec pour ton aide avec le matériel informatique et pour toutes ces discussions où l'on essayait de refaire le monde et où l'on rêvait d'une société différente. Merci aussi de m'avoir rapprochée de Niccolo et Giorgia ! Amigos italianos, gracias por tanto apoyo durante la tesis. Nicco, gracias por tus consejos y motivación. Je voudrais remercier aussi ceux qui sont partis avant moi du CEA et qui ont eu une grande influence sur la façon dont ma thèse s'est déroulée. Ils étaient là quand j'avais besoin d'un coup de motivation ou de conseils pour la rédaction et la soutenance. Je les appelle "mes anciens" : Alex Caldas, Franck Gonzalez, Pauline Maurice, Davinson, Jérémy Dallard et Walid. Je remercie également Selma pour m'avoir aidée à rédiger mon plan de thèse et pour les pauses détente dehors qui m'ont beaucoup aidée à relativiser. Tu es une femme que j'admire ! Merci Titouan pour l'intérêt que tu as porté à ma thèse et pour la bonne ambiance en bas ! François Lansade et Alexandre Douin, merci pour le recul que vous avez apporté sur mes travaux et sur le doctorat en général. François, merci pour ton expertise industrielle qui m'a inspirée pour mes expérimentations. Alex, merci pour tes retours d'expérience, nos discussions sur le management et ta motivation et organisation des courses de relais CEA. J'admire ton énergie !

Maintenant, j'ai besoin de dire que pendant ces années au CEA j'ai eu la chance de partager les déjeuners, les pauses et les robots avec des gens exceptionnels : Marie Charlotte (ma complice, ma copine de bureau, une ingénieure robotique avec beaucoup de classe, attention aux *crocrodrilles* et aux *écroucreuilles*, tu m'as énormement manqué à la fin), Nolwenn (madame rigueur et organisation, j'admire tes travaux de thèse... merci pour tes mots doux et ta motivation pendant ma dernière année), Vaiyee (toujours là pour nous aider à décompresser et faire des nouveaux amis), Emeline (l'artiste du groupe, merci d'avoir toujours été présente), Laura (la que comprende mi cultura colombiana, una super ingeniera mecatrónica et toujours motivée pour faire du sport)... et les garçons : Benoît ("tout va bien

se passer"... merci infiniment pour ton aide (le débogage et l'optimisation de mes codes), merci pour ta bonne énergie et pour avoir toujours été là pour moi, attention aux *canards*), Djibri-lo (le maker le plus passionné du labo, j'ai adoré parler avec toi de geekeries), Anthony (toujours content et la motivation incarnée), José (l'expert des robots mobiles et du vélo, des conversations toujours intéressantes), Oscar y Juan Miguel (la mejor oficina donde se habla español, gracias por tanto apoyo, por escucharme y por interesarse en mi trabajo... ánimo que les falta poco ! y cuidado con las plantas <3), Ugo (le petit dernier rigoureux, prends bien soin de mon bureau). Merci aussi à tous les stagiaires qui sont passés par le LRI/LSI et qui ont contribué à la bonne ambiance de travail ! Une pensée à Courosh, Sophie, Félix, Jérémy, Simon, Olivier, Pierre, Clément, Samouri, Rabah...

During my thesis I had the pleasure to work with Gennaro Raiola. Together we were very efficient and working seemed more like playing with robots. Gennaro, you are very talented and I hope we can keep up our collaborative work ! Thanks for sharing all your expertise with me and for the good moments at the lab. Et un grand merci aussi à Pauline Chevalier avec laquelle j'ai eu la chance de collaborer pour réaliser mes premiers tests utilisateurs. Merci Pauline pour ta grande contribution et ta bonne énergie.

Je voudrais remercier très particulièrement les stagiaires que j'ai co-encadrés pendant mon doctorat. Thomas et Jéssé, nous nous sommes complétés parfaitement et votre travail sur les interfaces en réalité augmentée a été très utile pour la thèse. Merci aussi pour la bonne ambiance et l'esprit de travail en équipe ! Je garderai toujours tous ces bons souvenirs avec moi.

Ce doctorat a été pour moi, plus qu'un travail de recherche et d'expérimentation en robotique, une grande opportunité d'enseigner et de transmettre mes connaissances. Fabien Dionnet, merci de m'avoir accueillie à l'ISEP et de m'avoir fait autant de confiance pour encadrer tes élèves puis pour diriger les TD. J'ai beaucoup appris de toi et de cette expérience merveilleuse. Un grand merci à mes élèves qui m'ont indirectement motivée et avec lesquels j'ai grandi. Gracias a mis profesores colombianos Hugo y Fabio que me motivaron a enseñar desde que estaba en el colegio y por siempre creer en mí. Mi eterna gratitud !

Je remercie toutes les personnes qui ont participé à mes expérimentations avec les différents robots, qui prenaient parfois longtemps ! Sans vous, cette thèse n'aurait pas eu de sens.

Merci à mes nouveaux collègues de chez iFollow qui sont venus renforcer les applaudissements : Vincent, Nicolas, Romain, Théo, Quentin, Jonathan ! Merci Jonathan pour ton aide à la préparation de ma soutenance.

Je voudrais aussi remercier très spécialement mes amis proches qui ont eu une influence indirecte sur la thèse mais très importante. Valentin, merci d'avoir eu plus de confiance en moi que moi-même, merci de m'avoir appris le français (on a encore du boulot...), merci pour ton soutien inconditionnel ! Guillaume et Guillaume... merci pour l'intérêt que vous avez porté à mes travaux et de m'avoir motivée à faire du sport, me déplacer en vélo à Paris et garder une vie sociale active pendant le doctorat ! Thomas (Toto), merci d'avoir partagé avec moi le congrès à Toulouse et notre passion pour la robotique depuis le Master avec Benoît (*canards*). Merci de m'avoir forcée à voyager en France pour me changer les idées, and thanks for being just a call away. Lina, gracias por compartir conmigo desde los proyectos integradores de mecatrónica hasta la tesis de doctorado. Gracias por tus palabras d'encouragement, por tus regaños toujours pertinents, por tu honestidad et tes relectures. Gracias por tanto, ya casi te toca ! Andrés, qué hubiera sido de mi sin vos todas esas tardes en el lab en las que faltaba motivación. Gracias por tus consejos y por escucharme hablar tanto de robots y con este acento paisa. Maud et Bertrand, merci d'avoir toujours été là et de m'avoir ouvert l'esprit. A mis amigas que desde Colombia nunca dejaron de apoyarme y alentarme e incluso algunas pasaron por la casa parisina llenándome de buena energía y amor: Laura, Maria, Andrea, Mariana, Manuela R., Ana, Manuela A., Isabel. En fin, al innombrable al que le debo parte de mi pasión por la mecatrónica y a mis compañeros de la EIA !

Y entre esos amigos... Camilo: mientras escribí esta tesis sentí como si todo hubiese comenzado gracias a aquel debate de filosofía en el 2008. Te debo mi amor por la France, por la ciencia y l'envie de débattre. Gracias por interesarte por esta tesis même si elle est loin de faire partie de ton domaine. Gracias por tanta motivación e inspiración. Gracias por esos dos meses de invierno en París que compartimos y que me ayudaron a seguir trabajando duro. Gracias por admirarme, j'ai toujours du mal à le croire, un genio como vos. Et merci d'être venu à ma soutenance, ce fut une grande surprise et un plaisir indescriptible de t'avoir eu dans l'amphi. Et une pensée à Juan Diego !

"El hombre crece de dentro para afuera", decía Fernando González en 1936. Y en su misma obra, *Los negroides*, donde critica la vanidad y la ceguera de América Latina, dijo: *"Lo primero es conocerse, y lo segundo, cultivarse. Nuestra individualidad es nuestro huerto, y la personalidad es nuestro fruto"*. Y en esos estados de consciencia propia y de aprendizaje, **la familia** es el ejemplo, la motivación, la fuente... la familia es el color de esas formas que cada uno dibuja, es el inicio y el fin, sin ser un molde. Yo aprendí de mi familia el amor, la entrega y el trabajo constante; Julio: el esfuerzo y el liderazgo, Olga: los buenos hábitos y la tenacidad, Sara: ejemplo de talento y creatividad inagotable, César: un maravilloso rebelde con causa a quien le debo mis primeros conocimientos en informática y el gusto por la *buena* música, y Samuel: pura energía e inspiración ! Un agradecimiento especial a Sara por haber contribuído activamente a la tesis con gran parte de las ilustraciones del

manuscrito y la presentación. Mil gracias también por tanto apoyo durante mis años de estudio y por ser incondicional incluso en la distancia. Gracias a mis padres por el alto nivel de educación que me dieron, por esas bases que han sido claves en mi desarrollo como persona y profesional. Gracias por creer en mí, por haberme dado las alas para volar lejos, lejos, lejos de casa. Gracias a los tres por venir a verme a la defensa de la tesis y escuchar tantas horas de presentación en francés, gracias por tanto amor.

A los Restrepo... especialmente a Miryam (mi amiga en la distancia pero a la vez tan cercana), Marta (por compartir conmigo la pasión por la docencia y la academia), Jorge (por el apoyo incondicional durante mis estudios y sobretodo por compartir conmigo la pasión por la domótica y la buena mesa), Andrés (esa visita no tiene precio, gracias por creer en mí) y Camilo (fuente de inspiración inagotable, de trabajo constante, un amor fraternal difícil de explicar). Manu, gracias por las tardes que pasamos juntas en Medellín cuando viajé durante el doctorado, tu interés en mi trabajo y tu sonrisa constante. Esta familia, es sin duda un ingrediente clave en el éxito de mi carrera.

Gracias a la familia Sánchez por el apoyo a distancia y los mensajes alentadores !

Un grand merci à la famille Guerry pour votre soutien et l'intérêt que vous avez porté à ma thèse, et surtout à mon bien être. Merci pour les weekends et les vacances en famille ! Merci à Liliane et Yves Guerry.

La thèse, plus qu'une expérience professionelle est un projet personnel. Et une partie très importante d'un tel projet, c'est l'état d'esprit. Pendant ces années de doctorat, j'ai eu la chance de partager ma vie avec Joris, qui a eu la grande responsabilité de me faire garder l'état d'esprit nécessaire à la réussite de mes travaux. Mais pas que... Joris je te remercie aussi pour toutes ces nuits où l'on a travaillé ensemble, que ce soit pour faire des figures avec GIMP, des simulations avec Matlab, des relectures, des vidéos sur mes expérimentations, des maths ou encore pour préparer mes présentations avec toute ta rigueur ! Merci pour tes bonnes idées qui ont eu une influence si positive sur ces travaux. Merci pour ta contribution à mon niveau de français et pour ton intérêt (si important pour moi) pour ma langue maternelle et ma vie en Colombie. Merci notamment pour la préparation de ma soutenance de thèse, pour laquelle le français me rajoutait une difficulté supplémentaire. Merci pour la relecture de ce manuscrit, l'encouragement pendant la rédaction et les bonnes pratiques en LaTeX. Merci pour nos débats sur la robotique et l'avenir de notre société. Merci pour ton enthousiasme et ta confiance en moi. Merci d'avoir construit avec moi nos deux projets de doctorat en plus de notre *chez nous*. Merci de m'avoir laissée prendre la main quand c'était nécessaire... mais surtout de ne pas avoir laissé coulé *le bateau* quand je perdais le contrôle. Merci de m'avoir transmis ton positivisme et ta passion. Gracias por tanto amor y entrega !

Y cierro este capítulo con González:

> *La emoción del conocimiento es lo que embellece... La obra una vez terminada, es objeto. Lo único dinámico, siempre prometedor y finalidad última es el espíritu ...*

... ese espíritu que crece en el amor, la responsabilidad y la disciplina. Mi espíritu apasionado, rebelde y acompañado ...

Gracias, totales !

Contents

List of Figures

List of Tables

Glossary

AR Augmented Reality. 81, 155–157

CEA The French Alternative Energies and Atomic Energy Commission. xiii, 10, 15, 26, 90, 104

cobot Collaborative Robot. xiii, 9, 11–13, 15, 20–24, 26, 31, 35, 36

ctg Constrained tool geometry. 32

DMFFD Direct Manipulated Free-Form Deformation. 96

DOF Degrees of freedom. xiv, 21, 37, 43, 47, 49, 66, 90, 154

DVG Dynamic Virtual Guides. 65, 66

geodesic In differential geometry, a geodesic is a generalization of the notion of a straight line to curved spaces. Geodesics are (locally) the shortest path between points in the space.. 71, 73

GMM Gaussian Mixture Models. xiii, 17, 18, 30, 31, 36, 54, 154

GMR Gaussian Mixture Regression. 30, 31, 54, 154

GUI Graphical User Interface. 36

HMM Hidden Markov Models. 17, 31

PbD Programming by Demonstration. 11, 15–17, 20, 23, 30, 31, 35–37, 51, 54, 64, 89, 151

RMSE Root Mean Square Error. i, xv, 124–127, 133–135, 145

SCODEF Simple Constrained Object Deformation. 96–99

SLERP Spherical Linear Interpolation. xiv, 68, 71–74

SQUAD Spherical Cubic Interpolation. xiv, 73–78, 83, 152

SRD Simple Radial Deformations. 96

Chapter 1

Introduction

> *How inferior the human machine is, compared to man-made machines. They can be decoked, unscrewed, oiled and parts replaced. Decidedly, nature is not a very wonderful thing.*

Joris-Karl Huysmans, In a letter to Arij Prins

Contents

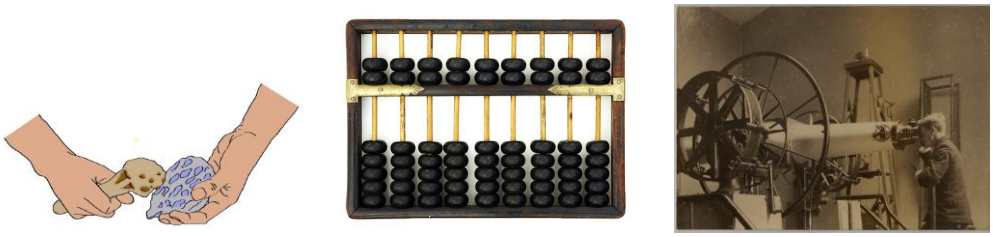
This thesis presents a new kinesthetic teaching framework to program Virtual Guides Assistance in the context of human-robot comanipulation, applied to industrial scenarios. Our approach is intuitive and flexible so it can be used by non-robotics experts and easily reprogrammed when the tasks or the environment change. Moreover, the human operator is assisted during the programming phase in order to reduce physical effort and cognitive overload. In this Chapter, we present

the context of this doctoral research in Section 1.1 by first introducing human-robot interaction (1.1.1), then defining the concept of collaborative robots (cobots) used for comanipulation tasks (1.1.2). Here we explain the features of cobots, including Virtual Guides Assistance and Programming by Demonstration, which are the base concepts used in our work. In Subsection 1.1.3, we present the concept of our programming framework. Finally, the state of the art of Virtual Guides and the related problematics are presented in Section 1.2, to conclude with the contributions of this thesis, raised in Section 1.3.

1.1 Context

1.1.1 Towards Interactive Robotics

One of the most distinctive characteristics of human beings is their will to increase their abilities by using a variety of tools. These abilities can either be physical, cognitive or perceptive. An example of tools used to enhance human power are shown in Figure 1.1.



(a) Homo habilis using a bone as a hammer.

(b) Abacus used for performing mathematical operations.

(c) Sheepshanks Telescope (1900) used in astronomy.

Figure 1.1: Examples of tools used to improve the (a) physical, (b) cognitive and (c) perceptive abilities of humans.

Robots are a kind of tool invented by humans in order to assist them at work. The first robotic applications were designed for specific tasks where industrial constraints such as precision, repeatability and production rate, shaped the conception of automatic machines. The robots from the 80's where conceived to execute repetitive tasks at high speed, in closed environments with very limited human intervention. An example of application is a "pick and place" task where a robot allows fast and accurate positioning of electronic components on a circuit board. Other examples are tasks such as palletization or industrial manufacturing. In general, the core idea of robotics has been to assist the human being to achieve difficult, arduous or dangerous tasks, such as space exploration missions (Figure 1.2).

The robotics field has evolved together with recently scientific developments in mechanics, electronics, informatics, applied mathematics and materials. These developments have improved areas like sensor technology, human-machine interfaces and high-level programming (see Figure 1.3), making robots more cost-effective,

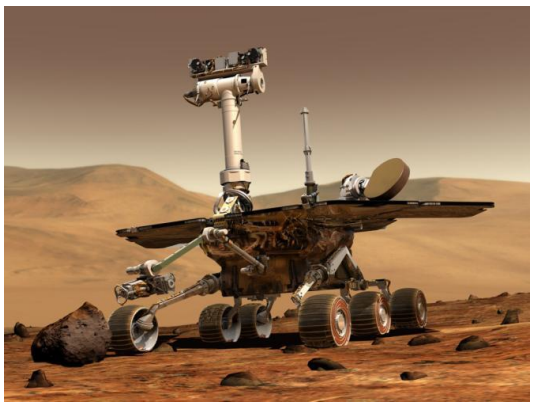


Figure 1.2: Spacecraft Opportunity used in the Mars Exploration Rover (MER) mission. Image credits: NASA/JPL/Cornell University.

robust, flexible and easy to use. We have passed from heavy and intimidating automatic machines to more safe and friendly mechatronic systems (see Figures 1.3(a) and 1.3(b)). Today, other than industrial and military applications, we can talk about service robotics, exploration robots, medical robots and even educational robotics (see Figure 1.4). These new fields and the growing people acceptance of new technology have revolutionized industrial robotics, particularly in small industries where traditional hard automation was complex or expensive to apply. Thus, new industrial comanipulated systems have been developed, in which the robot and the operator physically interact for carrying out the task. The human and the robot skills are combined in order to add flexibility to the system and enhance operators' ergonomy at work.

Human-Robot Physical Interaction in Industry In the last decade, manufacturing has benefited from the development of robotics. Productivity as well as quality have been enhanced and operational manufacturing costs have been reduced in the case of repetitive, complex or arduous tasks. The vision of the industry of the future, sometimes called *Industry 4.0*, is to reach even higher levels of productivity, efficiency, and self-managing production processes where human operators and robots not only share the workspace but communicate and cooperate with each other **through physical interaction**. Figure 1.5 shows the 9 components of Industry 4.0, where the key concepts are: *Internet of Things* (IoT), *Cyber-Physical Systems* (CPS), and *Smart Factories*.

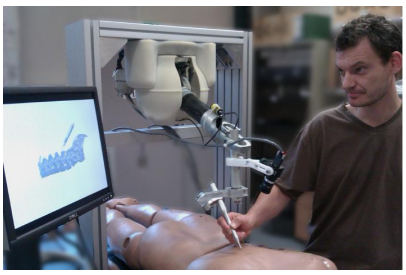
The context of this thesis is related to *Smart Factories*. A *Smart Factory* is defined as a factory that is context-aware and assists people and machines in the execution of their tasks [Lucke 2008]. A major component on *Smart Factories* development is *Flexible Automation*. Robots of today must be able to quickly and easily adapt to a change of product or environment. For example, in the case of small and medium-sized companies, improved robotic responsiveness and **flexibility** are of highly importance to decide whether to automate a process or not.

(a) Bebop drone and pilot interface from Parrot.

(b) Franka robot. The user can interact with the robot using a tablet or the tangible interface on the robot case.

(c) Embedded Kinect camera on a Robotnik Summit XL robot that allows to obtain low cost RGBD images. Image inspired from [Guerry 2017], where new neural networks fusion approaches using RGBD images are applied for people detection tasks in robotic exploration.

Figure 1.3: Examples of recent innovative human-robot interfaces and sensors.



(a) The surgicobot.

(b) Roomba.

Figure 1.4: Examples of recent innovative human-robot interfaces and sensors. a) The Surgicobot robot. Designed for milling assistance, especially in orthopedic surgery. It was developed by HAPTION (Laval, France) as part of the Surgicobot project (ANR Tecsan). Image extracted from [Françoise 2013a], where different control strategies for Surgicobot are proposed. b) iRobot Roomba Vacuum service robot. This mobile robot is equipped with a camera that allows it to navigate using VSLAM (Vision Simultaneous Localization and Mapping).

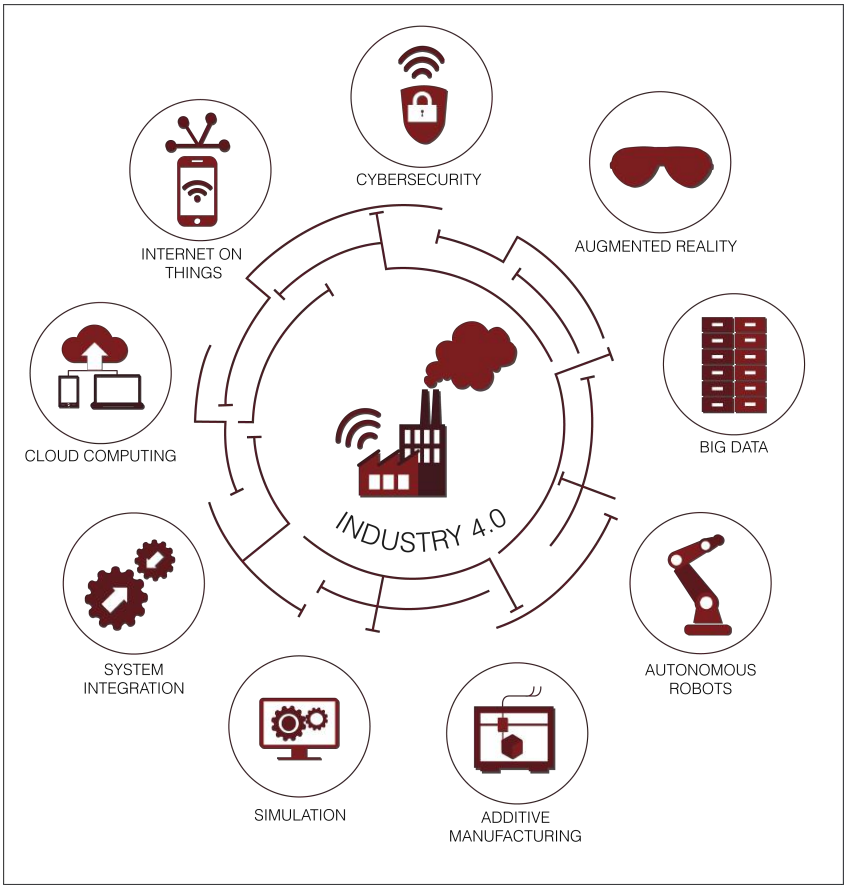


Figure 1.5: Components of the industry 4.0.

In this *Smart Factory* and *Flexible Automation* context, robots play a major role. Robotic systems must be able to "smartly" accomplish tasks while being **safe**, **flexible**, **versatile** and **collaborative**. Without needing to enclose the working environment of these robots, their integration into human workspaces becomes more cost-effective and productive, giving place to multiple possible industrial applications. Thus, smart robots will not only take place in simply structured workflows within closed areas, they will work hand in hand with humans to enable lean and intelligent production methods (see Figure 1.6). Indeed, the human operator interacts in the autonomously organized manufacturing system when human skills, such as dexterity and decision making, are required. Furthermore, he/she becomes a strategic decision-maker and a flexible problem solver. On the other hand, due to the increasing complexity of production, humans benefit from the support of *assistance systems*. These systems need to aggregate and show information comprehensibly to ensure that humans can make informed decisions and solve urgent problems on short notice [Gorecky 2014]. **They can also physically assist the human worker in the accomplishment of painful, exhausting or unsafe tasks**. For this assistance to be effective, successful and safe, it is necessary that robots **interact intuitively** with their human co-workers and that humans are properly trained for this kind of human-machine collaboration. Finally, these human-robot collaboration is a promising solution to enhance flexibility and promote the robotization of small industries.

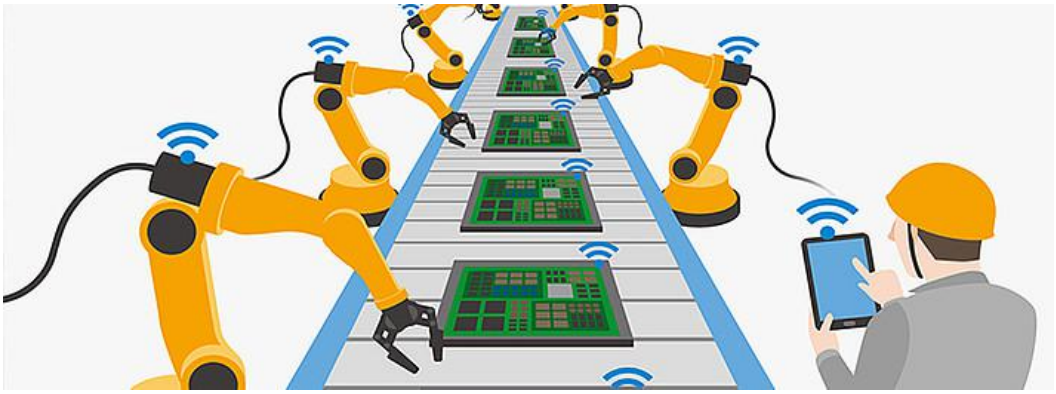


Figure 1.6: The vision of *Industry 4.0* is to have assembly lines where humans and robots directly interact in order to guarantee high flexibility in the production of highly customized products.

1.1.2 Collaborative Robotics and Comanipulation

For a very long time, automation was driven by the use of traditional industrial robots placed in cages, programmed to repeat more or less complex tasks at their highest speed and with maximum accuracy. This robot-oriented solution is heavily dependent on hard automation which requires pre-specified fixtures and time con-

suming programming, hindering robots from becoming flexible and versatile tools. These robots evolve towards a new generation of small, inexpensive, inherently safe and flexible systems that work hand in hand with humans. In these new collaborative workspaces the human operator can be included in the loop as an active agent. As a teacher and as a co-worker he/she can influence the decision-making process of the robot.

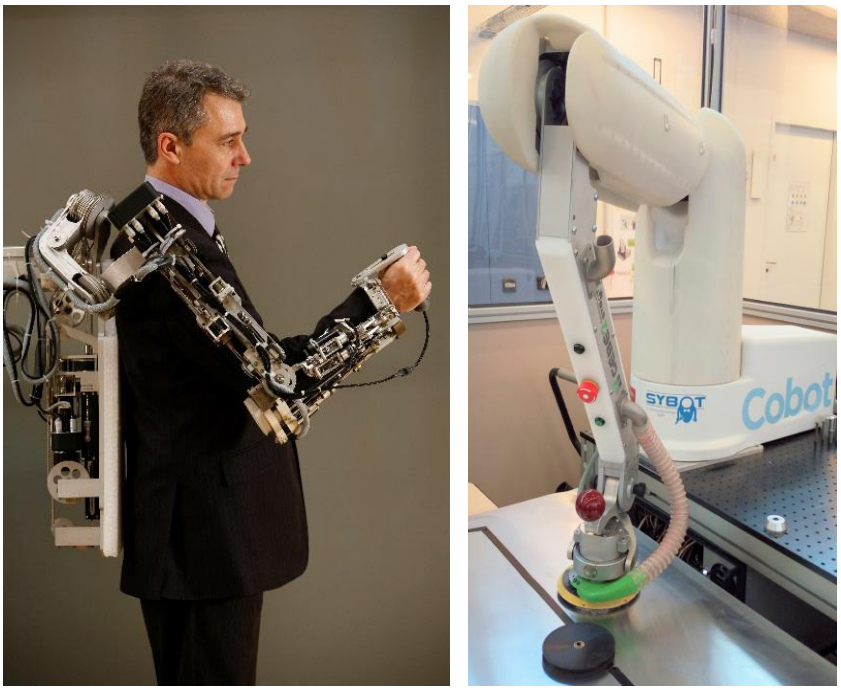
The context of this thesis resides on the so called *Collaborative Robotics*. In the literature, collaborative or cooperative robotic systems are understood as: Human-Robot collaboration [Dumora 2014], [Rozo 2014] and Robot-Robot collaboration [Shima 2009]. In this doctoral work we focus on Human-Robot collaboration and specifically on **Human-Robot comanipulation tasks in industrial environments**.

In this context, human-robot comanipulation means to add the human operator in the control loop. It aims to combine the job expertise, dexterity and cognitive skills of humans with the mechanical performances of robots (see Figure 1.7).

Figure 1.7: Collaborative robot from Universal Robots (UR5) used in Volkswagen's production line - "...While safety is imperative, that's simply the cost of entry into the *cobot* market now. We believe that being collaborative is just as much being accessible, lowering the automation barrier by placing robots within reach of manufacturers that never thought they would be able to deploy robots." Universal Robots.

Traditional industrial robots have been used for manipulation purposes [Lee 2006]. In the context of human-robot collaboration, it is desirable that the mechanical qualities of the robot should be as complementary as possible to those of the human worker. Industrial manipulator arms have been optimized for years to ensure high accuracy and speed in handling objects that can reach several tens of kilograms. This optimization work has resulted in a mechanical architecture that offers the best compromise workspace, speed, accuracy, strength and rigidity. All these qualities very complementary to those of the human workers are very appreciable in the context of comanipulation. The main counterpart is that to ensure significant rigidity and strength, more material and transmissions with high reduction ratios are needed.

This leads to relatively heavy robots, with significant inertia and joint friction. To overcome this challenges, [Newman 1994] proposed an impedance control reducing friction and inertia into the robot behavior. This solutions implies the use of a force sensor at the robot end-effector.



(a) Orthese ABLE. (b) ISybot.

Figure 1.8: Cobotic systems conceived by *CEA*.

In this thesis we will focus on comanipulation tasks with collaborative robots – called *cobots* –, which are specifically designed to **work hand in hand with humans** and though present higher transparency than traditional industrial robots conceived to work with limited human intervention.

An essential characteristic of the human-robot combination is the transparency of the robot, i.e, its capacity to move without resistance in the presence of external forces. A good transparency can be obtained with the help of the mechanical properties of the actuation chain of the robot. For example, as mentioned before, the high speed and accurate movements with important forces allowed by traditional industrial robots, are obtained thanks to the high reduction ratio transmissions which also produces non negligible articular friction. In addition to the high inertia resulting from their heavy structure, these robots present low transparency. Among the robots developed these last years for improving the transparency (see Figure 1.8), some of them use a solution presented by [Garrec 2010]. In this work, a novel actioning and transmission system was conceived based on a Screw and Cable System (SCS), allowing to obtain low friction and inertia and though high transparency. One

drawback is that this kind of transmission reduces the stiffness of the mechanical structure. However, a hundred percent transparency could be "an evil friend": when a robot is carrying heavy loads, if there is no viscosity there is going to be a lot of inertia and though it would be harder to manipulate the robot. Thus, a compromise between stiffness and transparency may be done depending on the final application.

1.1.2.1 Cobots

The term *cobot* was first introduced to refer to wheeled robots using computer-controlled steering for motion guiding [Colgate 1996]. Despite its specific initial meaning, the term *cobot* is now often used to refer to robots capable of safe physical interaction with human operators within a shared workspace. These cobotic systems must:

- be intrinsically safe (e.g, lightweight robots with collision detection features),
- reduce human physical effort and cognitive overload,
- enhance human ergonomy at work (reducing work-related musculoskeletal disorderser),
- take advantage of the user's gesture expertise,
- be flexible enough to handle mutable process and uncertainty and
- be intuitive enough to be set and programmed by non-robotics experts.

[Morel 2012] proposes an interesting classification of *cobot*s based on the characteristics of the physical interaction, which can be *parallel*, *serial* or *orthotic* (see Figure 1.9). The physical interaction is called parallel when the human manipulates the robot by its end-effector, the human-robot system forms a parallel kinematic system. Serial refers to hand-held devices, the human-robot system then forms a serial kinematic chain. With a parallel comanipulator, the efforts of the operator are added to those exerted by the active system to produce the movements of the manipulated object, while with a serial comanipulator, the velocities are added. Finally, the physical interaction is orthotic when the human-robot interaction is parallel but is distributed in multiple contact points. Orthotic comanipulators are also called exoskeletons. In this thesis we focus on parallel comanipulation with *cobot*s.

1.1.2.2 Cobots properties

*cobot*s provide a variety of functionalities such as *weight compensation*, *inertia masking*, *strength amplification*, *tremor filtering*, *Programming by Demonstration (PbD)* and *motion guidance assistance*. These properties aim at reducing the human worker's effort resulting from the interaction with the tool or the environment and improving the workstation ergonomics.

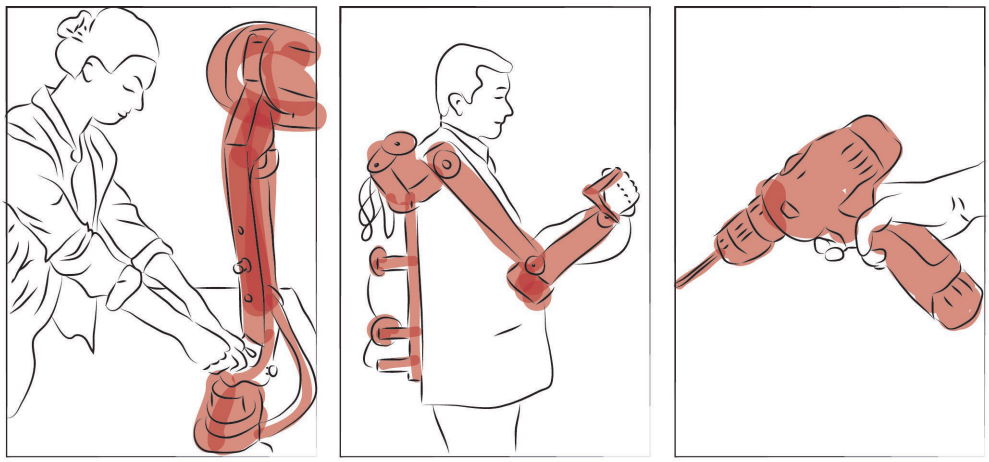


Figure 1.9: Classification of comanipulation. From left to right: parallel, orthotic, serial.

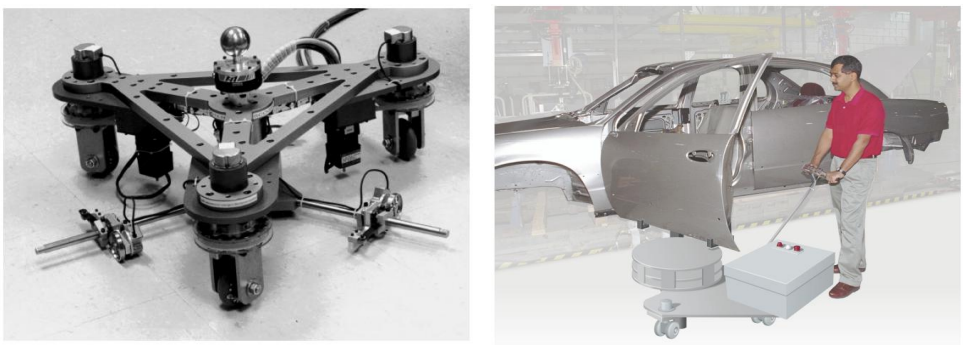
Weight compensation: This function is used in manual handling jobs, it consists in canceling the vertical component of the load gravity wrench. The load is hung up to a variable-length cable, and in current systems its vertical manipulation is power-assisted thanks to a force sensor set on the user handle (see Figures 1.10(a) and 1.10(b)). This kind of assistance was first proposed in 1969 with the Tool Balancer [Powell 1969]. This system was pneumatic and the compensating force was manually adjusted by fixing the pressure. An improvement was proposed by [Kornely 1989] using an electric motor to operate the system and introducing a force sensor, placed on the winder, to automatically adjust the weight compensation. More recently, [Kazerooni 2001] proposed to positionate the force sensor directly at the operator's handle, which makes the system even more sensitive and avoids the calibration procedure for each new object being transported. On the other hand, the operator can no longer handle the load directly. It is important to notice that all cable systems require the cable attachment to be aligned with the center of gravity of the object to be handled. In most cases, the cable can only be fixed above the center of gravity of the object. This does not allow an effortless displacement of the load along its roll and pitch axes, since the moments of the weight are not compensated. Only rigid mechanisms such as industrial manipulator arms allow the object to be freely manipulated in all its orientations independently of the position of the center of gravity relative to the point of attachment.

Inertia masking: Inertia masking is used for manual shifting of heavy objects. It consists in reducing the starting, stopping, and turning forces when manipulating a load, and ensuring that motions in all directions respond equally to human input. This function was proposed by [Peshkin 2001] on their scooter *cobot* (see Figure 1.11(a)). They proposed to control the steering of wheeled robots based on a virtual fixtures surfaces strategy. The load is fixed to a trolley and the orientation of the

(a) Free Standing EasyArm^{TM}, Gorbel. (b) iLift^{TM}, Stanley Assembly Technologies.

Figure 1.10: Weight compensation systems

wheels is automatically adapted according to the force applied by the user. The inertia effects experienced by the user during direction changes are therefore reduced. However, intertia is not reduced at acceleration or deceleration phases. Another example of inertia masking system is the iTrolley^{TM}(Stanley Assembly Technologies), which is an improvement of the iLift^{TM}weight compensation system. The inertial effects are compensated via fricton reduction by using a servo-controlled trolley and a measure of the cable deviation from the vertical axis.

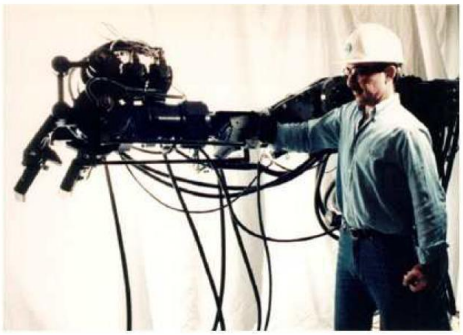


(a) Scooter three-wheeled *cobot*. (b) General Motors three-wheeled cobot

Figure 1.11: *cobot*s presented in [Peshkin 2001], where motion guidance assistance and inertia masking functionalities are applied.

Strength amplification: Strength amplification for comanipulation can be applied in two ways:

- to increase the effort the operator exerts on the tool or
- to increase the force felt by the operator relative to the one applied by the tool.



(a) Extender project.

(b) Hook Assistant.

(c) HULC exoskeleton from Ekso Bionics.

(d) Hercule exoskeleton from RB3D and CEA-List.

Figure 1.12: Strength amplification systems

In the first case, strength amplification consists in controlling the robot so that the force it exerts on the manipulated tool (or environment) is an amplified image of the force applied by the worker onto the robot. This function was first implemented

during the Hardiman project (General Electrics) [Groshaw 1969]. The system was based on an unilateral position coupling between a master manipulator physically attached to the user and a slave manipulator following the user motions while exerting amplified efforts on the environment. This idea was modified by [Kazerooni 1993] in the Extender project (see Figure 1.12(a)). In this system the master robot was removed so the user could directly interact with the slave robot. Today, exoskeletons are probably the most known strength amplification systems [Bogue 2009] as HULC[1] (Ekso Bionics) in Figure 1.12(c) or Hercule (RB3D, CEA-List), shown in Figure 1.12(d). These two systems are designed to help the user carry heavy loads without limiting the displacement. However non-orthotic strength amplification systems also exist. For instance, the Hook Assistant from Kinea Design (Figure 1.12(b)) designed for beef boning [Santos-Munne 2010], or the *cobot* 7A.15 (RB3D, *CEA*, CETIM) which has been used for various machining jobs. Strength amplification has also been implemented on generic industrial manipulators [Lee 2006, Lamy 2011].

In the second case, the efforts perceived by an operator are increased so that he/she delicately manipulates a tool whose interaction efforts with the environment are very weak. The principle remains the same as in the first case. However, the role of the tool and the operator are inverted and the dimensioning of the robot and the sensors must be adapted to lower forces. Such a functionality was implemented by [Cagneau 2008] in the context of laparoscopic surgery, on the surgical comanipulation robot MC2E^{TM}, in order to increase the perceived efforts of interaction of the tool on soft and fragile tissues.

Tremor filtering: Tremor filtering methods exploit the rigidity of the robot, the sensibility of the force sensors and an adequate viscosity control, in order to facilitate fine manipulation of tools by suppressing the tremors of the operator. This approach is mostly used in the field of surgical robotics, such as the Steady-Hand project [Kumar 2000, Mitchell 2007] where it is applied on retina surgery (see Figure 1.13). This modality have also been used on industrial comanipulation. [Erden 2011] applied it to a comanipulation robot to suppress the vibrations of the torch (attached to the end-effector) during a welding task.

Programming by demonstration *PbD* was part of the evolution from manually preprogrammed robots to more flexible user-based techniques to program them (see Figure 1.14). The concept is associated with the *observation-based learning systems* according to the *robot programming classification* proposed by [Biggs 2003]. Also referred to as *imitation learning*, *PbD* is a powerful tool for enhancing and accelerating the training process of a robot to acquire new skills. Its core idea is to overcome the major obstacles for natural and intuitive robot programming, such as high programming and high technical skills or tedious trail-and-error approaches. *PbD* allows to create a more collaborative environment in which humans and robots join in the teaching-learning process.

[1] Human Universal Load Carrier

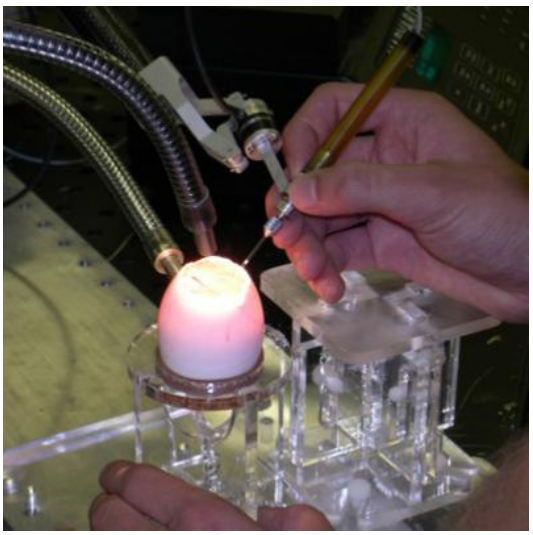


Figure 1.13: SteadyHand Project.

History *PbD* started to be applied in the field of manufacturing robots where it appeared as a promising solution to automate the tedious *manual programming* of robots while reducing the costs involved in the development of robots in industry. As a first approach to *PbD*, *symbolic reasoning* was commonly adopted in robotics [Segre 1985], using *teach-in*, *guiding* or *play-back* methods. Later, approaches using all demonstrated movements were developed to take into account the variability of human motion and the noise inherent to sensors (used to capture the movements). In [Muench 1994] it was suggested to use machine learning techniques to define a discrete set of basic motor skills. At this point, several key-issues of *PbD* were raised:

- how to generalize a task,
- how to reproduce a skill in a novel situation,
- how to evaluate a reproduction attempt and
- how to better define the user role during learning.

As stated in [Billard 2008], **the evolution of *PbD* techniques are related to the evolution of training interfaces**. Traditional ways of guiding/teleoperating robots were progressively replaced by more user-friendly interfaces, such as vision systems [Kang 1995], data gloves [Tung 1995] or kinesthetic teaching (i.e. by manually guiding the arm of the robot arms through the motion) [Inamura 2006]. These interfaces presented new opportunities to develop and extend *PbD* methods. As first suggested by [Muench 1994], the field progressively moved from only copying the demonstrated movement to generalizing from a set of demonstrations. Machine learning tools were incorporated in *PbD* approaches to address both the generalization

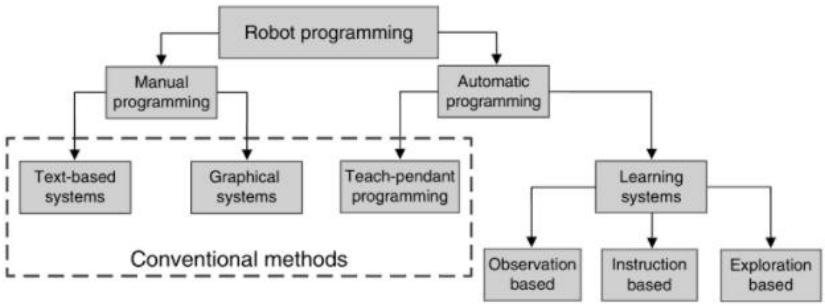


Figure 1.14: Classification of robot programming methods (taken from [Vakanski 2017]) *Manual programming systems* involve text-based programming and graphical interfaces. In *text-based programming* the user develops a program code using either a controller-specific programming language or extensions of a high-level multipurpose language, such as C++ or Python. The *graphical programming systems* employ graphs, flowcharts, or diagrams to create a program code. On the other hand, the conventional *automatic programming systems* employ a teach-pendant (i.e., hand held device with switches as shown in Figure 1.15) for guiding the robot links through a set of states to achieve desired goals. The robot joint positions recorded during the teaching phase are used to create a program code for task execution. *Learning systems* inspire from the way humans acquire knowledge: in *exploration based systems*, a robot learns a task with gradually improving the performance by autonomous exploration (often based on *reinforcement learning techniques*); *instructive systems* utilize a sequence of high-level instructions by a human operator for executing preprogrammed robot actions (often based on gesture, language and multimodal communication approaches); *observation-based systems* learn from observation of another agent while executing the task – we can place here the *Programming by Demonstration* approach.

of demonstrations and the adaptation of the movement to new situations. Between the used techniques there are: *Artificial Neural Networks (ANNs)* [Billard 1999], *Radial-Basis Function networks (RBFs)* [Kaiser 1996], *Hidden Markov Models (HMM)* [Lee 1996, Lee 2007, Calinon 2011] and *GMM* [Calinon 2007a, Raiola 2015] (see Figure 1.16).

Learning interfaces During the observation process of the learning phase, the robot must be able to record the movements of the human operator and the changes in the environment. Recent interfaces used in *PbD* were categorized by [Vakanski 2017] as follows:

- Kinesthetic teaching: the robot axes are manually manipulated by the operator in order to show the robot a desired movement, as shown in Figure 1.18.
- Direct control: the robot axes are guided through a control panel such as teach-pendant devices shown in Figure 1.15.

Figure 1.15: Teach-pendant programming interface (smartPAD) for a Kuka robot.

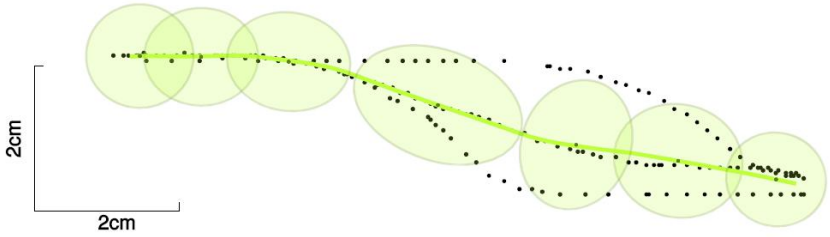


Figure 1.16: Example of a *GMM* (in green) trained on a data set of three demonstrated trajectories (in black).

- Teleoperation: the human operator controls a master robot which manages a slaved robot placed at a separate environment 1.17.
- Sensor based: such as vision, haptics, force, magnetic, and inertia.
- Virtual reality/augmented reality (VR-AR) environment.

Approaches In the literature, there are two kind of approaches to represent a task:

- low-level representation of the skill – known as *trajectories encoding* [Calinon 2007a] – and
- high-level representation of the skill – known as *symbolic encoding* [Pardowitz 2007, Ekvall 2006, Alissandrakis 2007].

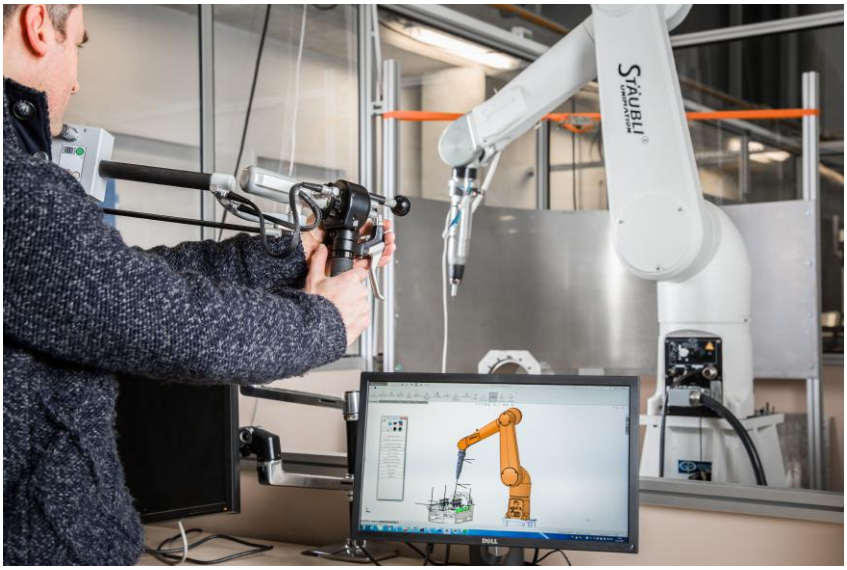


Figure 1.17: Teleoperation of a Staubli robot via a Virtuose 6D haptic device.

Table 1.1: Advantages and drawbacks of symbolic and trajectory level representation of a skill. Taken from [Billard 2008].

	Span of the generalization process	**Advantages**	**Drawbacks**
Symbolic level	Sequential organization of pre-defined motion elements	Allows to learn hierarchy, rules and loops	Requires to pre-define a set of basic controller for reproduction
Trajectory level	Generalization of movements	Generic representation of motion which allows encoding of very different types of signals/gestures	Does not allow to reproduce complicated high-level skills

When learning a skill at a *trajectory level*, human movements can be encoded in either joint, task or torque space. The encoding may correspond to cyclic motion, discrete motion or to a combination of both. In *symbolic encoding*, the skill is decomposed in a sequence of action-perception units. Some advantages and drawbacks of trajectory and symbolic level skill representation are presented in Table 1.1.

Other approaches use *Dynamic Movement Primitives (DMPs)* to encode the dynamics of the movement during the observation phase [Ijspeert 2003, Stulp 2013]. This is a robust strategy to face dynamical changes in the environment. Recently, it

was stated in [Calinon 2013] that *an efficient prior assumption in robot learning from demonstration is to consider that skills are modulated by external task parameters.* Under this theory, they proposed a *Task-Parameterized Gaussian Mixture Model (TP-GMM)* for representing skills based onboth rotations and translations in Cartesian space [Calinon 2013, Calinon 2014]. The proposed TP-GMM approach was applied in [Calinon 2018] to a wider range of affine transformations, including constraints in both configuration and operational spaces, as well as priority constraints. The interested reader is referred to [Billard 2016] for more details on *PbD* approaches.

Commercialized solutions Simple functions based on Programming by Demonstration and using kinesthetic teaching, are already being used in commercialized *cobot*s such as Baxter and Sawyer (Rethink Robotics®), Yumi (ABB®), LBR iiwa (Kuka®) and ISybot®(see Figure 1.18).

(a) Sawyer. (b) Yumi.

(c) LBR iiwa. (d) ISybot.

Figure 1.18: Kinesthetic teaching with several commercialized *cobot*s.

Conclusion Programming by Demonstration is used to learn and generalize a skill to be reproduced later by the robot in the same or different conditions. In this thesis we will use *PbD* in a different context. We aim at programming *motion guidance assistance* on a *cobot*. In this case, the *cobot* use an encoded displacement trajectory (cartesian positions and orientations) obtained by kinesthetic teaching,

not to reproduce it but to guide the user through it – in a passive way – during a comanipulation task. Previous works on this subject will be addressed in Section "Related works". *Motion guidance assistance* functionality is explained next.

Motion guidance assistance *cobot*s are capable of generating rigid constraints while being intrinsically passive. *Motion guidance* consists in limiting the end-effector *DOF* by hardware or software means, so that the tool can only perform specific motions [Book 1996]. Thus, the human technical gesture gains in speed and accuracy, while requiring less co-contraction effort[2] and concentration. This concept has been used in industrial applications, teleoperation and medicine. In [Colgate 2003] an overview of the use of *Intelligent Assistance Devices (IADs)* and virtual guides in industrial applications is presented. [Ryden 2013] uses virtual guides to teleoperate an underwater robot. In [Park 2001] and [Becker 2013], it is suggested that accuracy and stability of robotic end-effectors is essential for surgical tasks and could be increased using motion guidance assistance. Also, this functionality may facilitate physical rehabilitation [Reinkensmeyer 2004].

Different approaches exist to apply motion guidance and constrain the robot end-effector. *Mechanical techniques*, like continuously variable transmission systems (CVT) [Peshkin 2001], are advantageous since they are intrinsically passive. However, most of these methods do not allow to set stiffness constraints which are important features in comanipulation tasks. In addition, the implementation of such systems could be cumbersome or complex for manipulator robots with six *DOF*. Conversely, *automatic control approaches* can be easily applied to any kind of industrial robots available in the market and they allow to set up the robot compliance. Among these automatic methods, *virtual fixtures* [Rosenberg 1993] are more suitable to describe constrained movements. The specification tools used to define them, which are forward kinematics and virtual fixtures Jacobian, have the advantage of being widely used in robotics. In this thesis we will refer to virtual fixtures as *Virtual Guides* or *Virtual Guides Assistance*.

Although virtual guides could be functionally equivalent to guides in the real world, there are many advantages inherent to virtual guides because they are defined by software rather than physically implemented. In fact, when applied on a workstation, the guides only interact with the user and not with the workspace. So the workstation geometry does not impose constraints on the placement or configuration of virtual guides. In addition, **a virtual guide has no mass, no physical or mechanical constraints, requires no machining time or maintenance and can be easily modified.**

The idea of virtual guides is to separate the Cartesian space in two: one part of the degrees of freedom being controlled in position and the other in effort (hybrid control position/force [Raibert 1981]). The degrees of freedom being controlled in effort correspond to the displacements allowed to the operator, and the degrees of

[2] humans tend to constrain the antagonistic muscles to stiffen their movements, thus facilitating the appearance of *Musculoskeletal disorders (MSDs)*

freedom controlled in position correspond to the locked directions. Also, virtual guides can be assimilated to a set of ideal mechanical links (i.e., energy-neutral mechanisms) that would be connected to the robot's effector, with the objective of giving the tool movement a particular law. [Joly 1997] introduced the notion of virtual mechanisms. The force control of the robot is calculated from the simulation of a passive virtual mechanism, connected to the effector of the robot by a fictitious spring / damper system as illustrated in Figure 2.3. The advantage is that the stability of the system is independent of the chosen virtual mechanism. In this thesis we focus on the **motion guidance assistance** functionality using **virtual guides** based on the **virtual mechanisms** concept proposed by [Joly 1997].

Conclusion on cobots properties We have explained several properties and functions of *cobot*s that are useful in a human-robot comanipulation context. In this thesis, we propose to use *Programming by Demonstration* to intuitively program *Virtual Guides Assistance*. Next, we expose the limits of the existing approaches and we present the related works and the contributions of this thesis.

1.1.3 Intuitive, Iterative and Assisted Virtual Guides Programming

As explained before, using virtual guides allows to reduce the physical effort and cognitive overload of the user during the execution of a task. This kind of assistance could be used in industry on manipulation tasks (insertion, assembly), cutting, drilling and polishing. As a reminder, the assistance superpose a synthetic force to forces perceived by the user through an haptic control interface of the robot. The applied force constraints the user's movements and so those of the controlled robot through a particular trajectory, a surface or restrained volume, allowing to assure motion guidance during the task accomplishment.

A known issue with virtual guides assistance is to program the virtual guides on a robot. At least one of the following requirements should be met:

- expert knowledge of the task,
- high technical expertise,
- modeling of the task,
- high programming skills.

Some of these requirements restrict the usefulness of virtual guides to scenarios with unchanging constraints. Yet, many industrial applications require multiple operations to perform a task for which it would be necessary to easily generate and modify the guides in order to adapt to different tasks and situations:

- changes on the part to work on (polish, sand, cut),

- high variability due to low production volumes,
- transporting an object to one of multiple possible positions or
- different assembly sub-tasks to perform based on the tools availability.

Moreover, in an industrial context, virtual guides assistance is used by human operators with the gesture expertise to realize a task but often without modeling or programming skills. It is then less efficient and cost-effective for a factory to relay on robotics engineers to program the virtual guides and modify them when there is a change on the task or the environment.

For these reasons, we propose a **kinesthetic teaching framework** to program virtual guides assistance in an **intuitive and flexible way** so it can be used by **non-robotics experts** and **easily reprogrammed** when needed.

However, there are also some limitations when using kinesthetic teaching. The remaining inertia and articular friction of *cobot*s – even after gravity compensation and inertia masking techniques being applied – can disrupt the fluent execution of movements and hinder their manipulation. It is for example difficult to impose a rectilinear trajectory because the robot tends to follow curved trajectories due to joint space friction. Also, it is sometimes difficult to perform slow and small displacements precisely. In order to keep the desired trajectory, the user must provide additional effort and concentration. Most *PbD* approaches using kinesthetic teaching need several demonstrations to encode a trajectory. This could be exhausting for the user and contradictory to our approach since we are using *PbD* techniques to program virtual guides in order to assist the user later on the execution of a task. For these reasons we suggest that **the operator must be assisted during the programming phase**. To this aim, we propose to **use virtual guides programming in an iterative way so only one demonstration without assistance is needed**. After the first demonstration, the assistance is activated and the user is able to iteratively refine the virtual guides or modify them whenever is needed. Elementary pre-programmed paths (primitives), such as straight lines, circles or straight corners, can also be used to start the programming phase and be assisted from the beginning.

This iterative programming approach could also be used to reduce the cognitive load of users and increase their confort during the teaching process. Complex trajectories in space, including cartesian position and orientation of the tool, require high levels of concentration and effort, which could lead to trajectory encoding errors while using the kinesthetic teaching technique. Thus, we also propose to **simplify the programming process by uncoupling cartesian positions and orientations during the demonstrations**. The same idea can be used when other features are needed for the comanipulation task execution and though must also be taught to the *cobot*, such as speed, variable compliance and force.

Next section presents the works related to the context of this thesis, to the formerly raised limitations on the scientific fields of interest – Programming by

demonstration and Virtual Guides Assistance – and to the approaches proposed in this thesis.

1.2 Related works

Virtual guides have been featured in several works using different definitions, applications and implementation methods. In this related works section we will cover and classify some relevant works based on four main questions:

- How to *define* a virtual guide?
- How to *create* a virtual guide?
- How to *enforce* a virtual guide?
- How to *modify* a virtual guide?

For each question, we will compare these works with our implementation of virtual guides assistance. For the first three questions we inspire from the exhaustive classification given by [Bowyer 2014a].

1.2.1 Virtual Guides Definition

In this thesis, *virtual guides* are used to enforce virtual constraints on the movements of *cobot*s, in order to assist the user during a collaborative task. Virtual guides have been first introduced by [Rosenberg 1993] as *Virtual Fixtures*. The fundamental concept is that virtual fixtures can reduce mental workload, task time and errors during teleoperated manipulation tasks. After Rosenberg's initial work, the use of virtual fixtures has been extended to robotic surgery under the name of *active constraints* [Davies 2006] and to industrial applications in the context of *Intelligent Assist Devices* [Colgate 2003]. Nowadays, virtual fixtures have been featured in several different works, but unfortunately "there is currently no definitive concept which unifies the field" [Bowyer 2014a].

Teleoperation and Hands-on Device approaches The first criteria to classify the virtual guides definition is done according to the user-robot interaction. Generally, virtual guides have been used in teleoperation [David 2014, Xia 2013] or comanipulation contexts [Lin 2006, Dumora 2014, Vitrani 2017].

In *teleoperated* systems, the user controls a master robot that manages the movement of a slave robot which accomplishes the task in a separate environment [Joly 1995, Aarno 2005, Abbott 2005, Bowyer 2013]. Thus, as shown in Figure 1.17, the human user is physically separated from the robot carrying the tool.Teleoperation offers benefits such as motion scaling and the possibility to operate in restricted and unsafe environments. For example, [Ryden 2013] used virtual guides to teleoperate an underwater robot and [David 2014] proposed a supervisory control system using

Figure 1.19: In [David 2014] virtual guides are created on the fly into a physical engine, using linear interpolations. Since the guides are not programmed into the real robot controller, slight variations in their respective positions are possible. Also, in the context of co-manipulation tasks it would be more natural to program virtual guides in the real workspace rather than in a simulated one. This is one of the advantages of the hands on approach compared to the teleoperation.

virtual guides to speed up a disk-cutter insertion process, as illustrated in Figure 1.19. In some of these teleoperation applications, virtual guides are created into a physical engine and though environment modeling is required, which could be a limitation. In addition, since the guides are not programmed into the real robot controller, one drawback of this method is the possible position errors induced by the fact that the model may be mis-referenced with the reality. Moreover, the required information about the environment is interpreted at the slave robot level, then transmitted to the master device to be finally presented to the user either by visual or haptic feedback. This makes the process poorly intuitive.

In *hands-on devices* approaches, the user is allowed to directly interact with the robot through physical contact [Restrepo 2017, Becker 2013, Pezzementi 2007a], as shown in Figure 1.20. This kind of interaction is more intuitive thanks to the direct feedback the user has while manipulating the robot carrying the tool. Also, the user is better integrated in the process compared to the case where the user interacts with the workspace through a teleoperated robot. However, security of interaction must be guaranteed by the comanipulation system since the user and the robot share the workspace. This aspect has been studied by [Lamy 2011, Makarov 2013, Maurice 2017] and corresponds to a whole new field that will not be treated in this thesis.

Another example of hands-on interaction is presented by [Dumora 2014] in the context of big objects comanipulation. A library of virtual guides was hard

Figure 1.20: Scara *cobot* conceived by *CEA*. This image shows an example of *hands-on* interaction where the user directly manipulates the robot.

programmed on the robot so it could detect the intention of the human collaborator and activate the right assistance from the library. This approach was implemented to assist an operator when the task and the environment are unknown. For applications where erroneous or uncontrolled tool positions present a significant danger, a hands-on robot is likely to be safer than a teleoperated one because the user can regulate it and the actuation system can be conceived to be more compliant. Moreover, when applying motion guidance functionality to a hands-on device, the fixtures will constrain both the user and the tool with perfect correspondence between them. In a teleoperation system, the constraint could be apply to the master, to the slave or to both, which could have considerably different properties [Abbott 2003]. **Our work falls in the hands-on interaction category** [Restrepo 2017, Raiola 2017a]. In the approach presented in this thesis, human operators directly manipulate *cobot*s during tasks execution or programming.

Impedance and admittance constraints Another category of the virtual guides definition can be established according to the control scheme implementation. *Impedance* and *admittance* control are used because they are both suitable for environment/user interaction and object manipulation [Hogan 1989]. They are both based on the reciprocal mechanical properties of impedance and admittance/compliance. In *impedance* control, when a constraint is about to be violated, the controller applies a nullifying force to the robot to block motion. The control variable is the force of resistance to external motions imposed by the environment. On the other hand, an *admittance* control acts like a filter on the user's intended motion, where only the components of motion respecting the constraints are transmitted

to the robot's actuators.

A difference between both control schemes is that devices controlled in impedance must be highly backdriveable (high transparency) so that the user is able to move them freely when the controller force is null. Inversely, devices controlled in admittance are highly stiff in response of external forces and only move when commanded by the controller (not backdriveable). The high mechanical stiffness and non-backdrivability of a typical admittance-controlled robot allow for slow and precise motions, making it highly suitable for tasks that require accuracy near human physical limits, such as microsurgery [Burschka 2005].

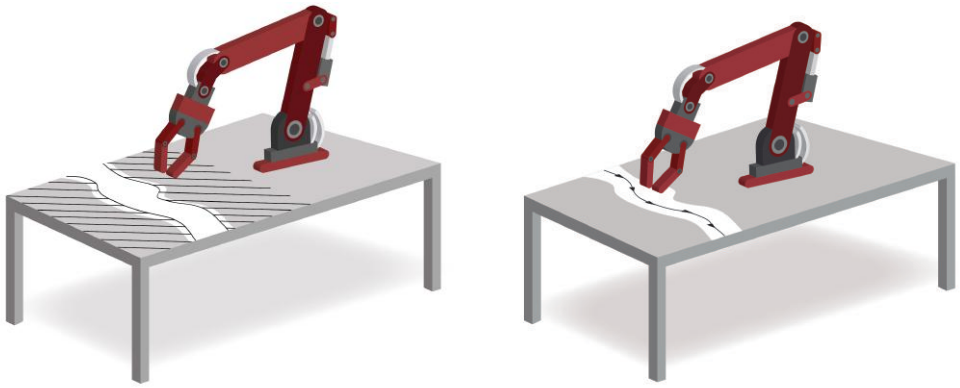
An implementation of impedance control is presented in [Joly 1995], where a passive virtual mechanism is connected to the robot end-effector by a spring-damper system. Impedance controllers have also been used by [Pezzementi 2007a] where they are called "proxies". Implementations of admittance constraints are presented in [Marayong 2003, Bettini 2004] where anisotropic matrices are used to attenuate the non-preferred force components. These methods require sensing external inputs, such as the force or the velocity applied by the user on the robot end-effector. This is not required on impedance control architectures.

The approach presented by [Kosuge 1995] also uses the term virtual mechanism to impose to the robot the behavior of an *ideal virtual tool* by specifying its dynamics. However, this approach is different from the virtual mechanism definition presented by Joly [Joly 1995], who demonstrated in [Joly 1997] that the control law proposed by Kosuge [Kosuge 1995] is not robust to a poor identification of the system. **Our work is based on the impedance control scheme defined by [Joly 1995].**

Regional and guidance constraints The type of assistance offered by the virtual guides are either used to limit the user to a task-specific pathway – *guidance constraints*) [Marayong 2003, Burghart 1999a, Bettini 2004] – or to limit the user to move the robot within a safe region – *forbidden region constraints* [Abbott 2003]. We present both categories in Figure 1.21.

Regional constraints allow to separate the workspace into two regions: free and forbidden. In the free region, the robot must not apply any force to the tool, it is said to be as transparent as possible. Conversely, in the forbidden region, the robot must be able to apply forces at the border of these regions so the tool does not to penetrate into them. The interest of this type of virtual guides is demonstrated for comanipulated orthopedic applications [Françoise 2013b]. Regional constraints present some other advantages such as: task simplification [Rosenberg 1993, Payandeh 2002], reduced risk of tools damaging protected regions/obstacles [Li 2005a], and avoidance of kinematic singularities [Turro 2001b].

On the other hand, guidance constraints are more intrusive upon the user than regional constraints. However, the ability to constrain positions precisely, allowed by guidance constraints, is important in many industrial and medical applications. One example is the application of these constraints on the Steady-Hand robot in the context of ocular micro-surgery [Marayong 2004] (see Figure 1.13). The surgeon



(a) Virtual guide defined as regional constraint. (b) Virtual guide defined as a guidance constraint.

Figure 1.21: Regional and guidance constraints (in black).

is assisted during an extreme precision task execution.

Conclusion on virtual guides definition The particular implementation of virtual guides used in this thesis is based on *impedance control*. This implementation is general and can be extended to both regional and guidance constraints. **In our work we are interested in generating trajectories that will be used as *guidance constraints* in a *hands-on comanipulation* context**. Table 1.2 shows a summary of our virtual guides definition.

Table 1.2: Summary of our virtual guides definition

Human-robot interaction		Control scheme		Constraints	
Teleoperation	Hands On	Impedance	Admittance	Regional	Guidance
	✔	✔			✔

1.2.2 Virtual Guides Creation

There are many possible solutions to create virtual guides. In most works of the literature, virtual guides creation is done *a priori* by a separate process which usually involves a human operator. There are some methods where constraints are generated autonomously based on medical image processing [Navkar 2012, Li 2003, Kumar 2003, Park 2011]. However this methods are mostly applied to well know tasks where geometry models can be obtained with the aid of a vision system. So the creation of guides is done automatically online but there is actually some complex beforehand preparation. This is for example the case of collaborative robotic surgery. In the context of this thesis we do not explore this kind of automatic creation processes.

We relay on the human expert knowledge of the task and we seek for an accelerated and intuitive method using less of extra material as possible.

Usually, the way to create virtual guides is highly related to the final application. The chosen *constraint representation* directly affects the geometrical form of the constraints and the choice of the *constraint evaluation method*. Figure 1.22 shows a virtual guides general framework explaining the relationship between definition, creation, evaluation and enforcement of virtual guides (constraints).

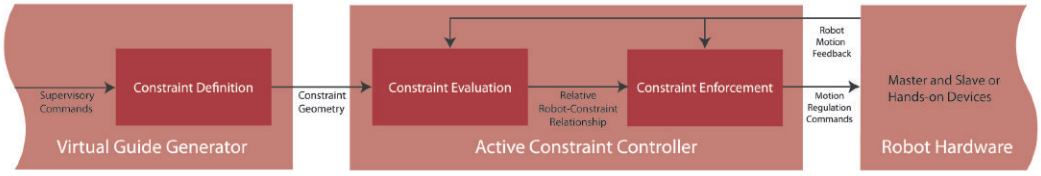


Figure 1.22: Virtual Guides general framework.

Virtual guides have often been limited to pre-defined *geometric shapes* [Marayong 2003] or combinations of shapes [Aarno 2005, Kuang 2004], well-defined *geometric models* [Joly 1995, Dumora 2014] or defined through *high level tasks models* [Xia 2013]. The use of un-reconstructed raw data (i.e., *point clouds*) is currently morewidespread within the literature [Yamamoto 2012, Rydén 2012] since constraints can be produced and maintained effectively in real time. Some simpler methods use *points* [Bettini 2001, Prada 2005, Marayong 2003], *lines* [Burghart 1999b, Srimathveeravalli 2007] and *planes* [Rosenberg 1993, Seung 2010], which can be efficiently represented in memory. However, these approaches highly restrain the creation of virtual guides for complex tasks. Point constraints are limited to simple positioning-like tasks. With linear constraints, the number of following trajectories that can be represented is limited. Some works tried to overcome this limitation by combining multiple short, linear constraints to create more complex guides [Aarno 2005].

A more flexible approach uses parametric curves as sinusoidal functions or splines to create guides [Bettini 2004, Kwok 2013, Marayong 2004, Li 2005b]. The main advantage is that they are geometrically flexible and though are able to describe complex tool trajectories with still reduced computationally cost. **Our work focus on parametric curves representation [Restrepo 2017]. Their flexibility also allows simple and local modifications**.

Parametric surfaces are an extension of parametric expressions from curves into 3D surfaces [Burschka 2005]. In general, nonuniform rational B-splines (NURBS) are used to create both regional and guidance constraints [Ikits 2003]. *Hyperplanar constraints* are an effective method to create constraints but being geometrically rigid, it is difficult to apply them to complex environments. As for lines, they can be combined to overcome their flexibility limits, but once a certain level of complexity is reached, efficiency is degraded [Kapoor 2006]. A more complex method uses *polygonal mesh constraints*, mostly for applications where the constraint profiles are extracted from "real world" surfaces [Li 2007, Ren 2008]. However, for some applications, the

degree of flexibility offered by meshes is not worth the increased complexity. Finally, *volumetric primitive constraints* can be combined in an effective way to represent complex regional and guidance constraints [Prada 2009, Kuang 2004]. Other approaches for creating virtual guides use *explicitly described constraints* [Turro 2001a] or *artificial neural networks constraints* [Ren 2008, Cretu 2003]. However, the former are highly inflexible and the latter are still very challenging and little widespread.

Virtual Guides creation using Programming by Demonstration Recently, *PbD* has appeared as a promising solution to program robots in a fast and simple way when the task is known by the user. Several *PbD* approaches, allow the operator to directly manipulate the robot end-effector to teach a desired movement – *kinesthetic teaching*. This technique takes inspiration from the way humans learn new skills by imitation to develop methods to transmit tasks to robots (see Figure 1.23).

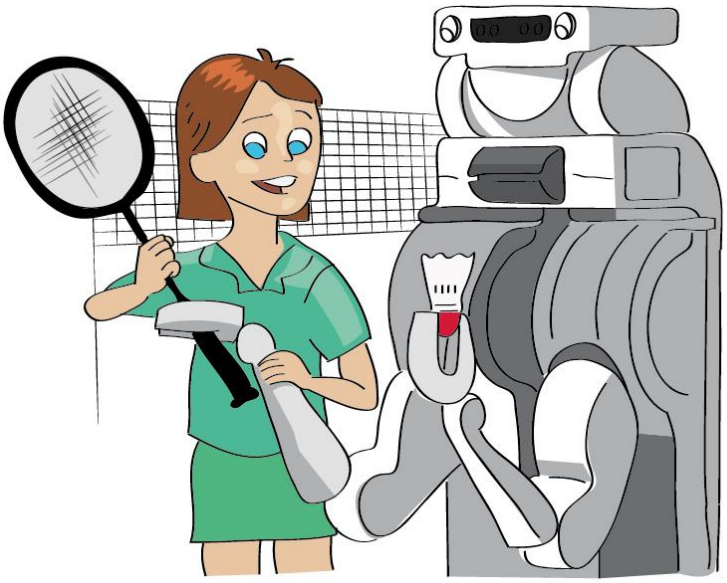


Figure 1.23: Programming by Demonstration analogy with humans learning by imitation. Inspired from the original Willow Garage image under BY-NC Creative Commons license.

In *PbD*, teaching a path usually involves modeling the demonstrated set of trajectories and retrieving a generalized representation of the data set suitable for reproduction by a robot. In [Calinon 2010] and [Vakanski 2012], some stochastic methods allow the extraction of a mean trajectory using a set of demonstrations and a rate of variability associated to each portion of the curve. In [Calinon 2008] the authors used *GMM* to encode a set of demonstrations and then reproduce the task using Gaussian Mixture Regression (GMR). *GMR* allows to retrieve smooth generalized trajectories with associated variance matrix describing the variations and correlations between demonstrations. In [Rozo 2014] *PbD* was used to teach the

robot a desired path and the force needed to perform a human-robot cooperative transportation task. These approaches use *PbD* for transmitting new skills to a robot so they can be reproduced later. In this thesis, we get inspired from these works and **use kinesthetic teaching to program displacement (cartesian position and orientation) trajectories on a *cobot* in order to create virtual guides.**

Generating virtual guides using *PbD* has also been explored by [Aarno 2005]. Their adaptive approach used *HMM* to model and detect optimum guides obtained by demonstration and represented as a sequence of linear guides. Here, the automatically applied guide corresponds to the most probable state. Another interesting work about virtual fixtures and *PbD* has been conducted by [Yoon 2014]. In this work, the authors *personalize* virtual guides based on a set of demonstrations provided by the users in order to match their preferences about the assistance. In our work [Raiola 2017a], we proposed a framework for multiple probabilistic virtual guides where kinesthetic teaching was used to create virtual guides. This probabilistic framework involves modeling a demonstrated set of guides with *GMM* and retrieving a generalized representation of the data set using *GMR*. Unfortunately, with these stochastic approaches, a compromise must be made between the number of demonstrations – time and effort demanding – and the level of information in the training data. In a comanipulation context, requesting multiple demonstrations of a single task could be exhausting for the user. **Therefore, we believe that a *PbD* approach in this context, should be capable of teaching a task to the robot from as few demonstrations as possible.** From the user perspective, the objective is that the robot start performing the tasks right away to gradually improve its performance while being monitored by the user. Besides, must of the literature approaches give a high level of autonomy to the robot to decide what to do when changes occur in the environment or in the task. **In this thesis, we relay on the human operator gesture expertise and choose him/her to be the master of the teaching and the execution of the task**.

Virtual guides evaluation After the *definition* and *creation* of a virtual guide, a way to *evaluate* it must be determined. In general, the proximity between the robot tooltip and the virtual guide geometry is evaluated. However, virtual guides evaluation highly depends on their final application, and so we will not present a wide review of these evaluation methods in this thesis. In Chapter 3: "Virtual Guides Construction" we will explain our virtual guides evaluation method based on Akima splines and quaternion interpolations. As stated before, parametric curves such as splines are a flexible way of defining trajectory constraints while allowing simple and local modifications.

1.2.3 Virtual Guides Enforcement

There is a wide variety of methods that are described within the literature for *enforcing* virtual guides. In general, the input of these methods is the current

relative configuration of the Constrained tool geometry (ctg) and the virtual guide geometry. Based on this relative configuration, the virtual guide enforcement module decides if, and how, motion regulation should be applied (see Figure **??**). We next explain the most used approaches in the literature based on the classification presented in [Bowyer 2014a].

Simple functions of constraint proximity This method uses a simple function of the proximity between the *ctg* and the virtual guide in order to compute the effect of the virtual guide on a device. The work of [Prada 2009,Gibo 2009,Seung 2010] used linear functions to enforce virtual guides. Generally, linear functions are described by the mechanical analogy of a spring system, with stiffness k_p, between the *ctg* and the virtual guide. A linear function for a Cartesian pure spring impedance controller takes the form:

$$f_p = k_p(p_{const} - p_{ctg})$$

where f_p is the constraint force vector, k_p is a linear gain as in a classic proportional position controller, p_{const} is the closest point on the constraint geometry to the *ctg*, and p_{ctg} is the closest point on the *ctg* to the constraint. The magnitude of k_p has an effect on how much the constraint acts on the robot. A positive value generates an attractive constraint while a negative value generates a repulsive one. The force f_r is applied to the robot using the Jacobian and the principle of virtual work [Seung 2010, Prada 2009]. The stability of this method is detailed in [Abbott 2006, Joli 2007].

Linear enforcement methods have been improved for various applications by adding derivative terms on the function [Ammi 2007, Ortmaier 2006]. If a user moves against a constraint at a higher difference of velocity, then the force from the controller in order to slow them down will be bigger. By introducing a derivative term (viscosity), the controller is able to respond to this situation. This takes the following form:

$$f_{pd} = k_p(p_{const} - p_{ctg}) + k_d(\dot{p}_{const} - \dot{p}_{ctg})$$

where f_{pd} is the constraint force vector, and k_d is a derivative gain. An extension to viscoelastic constraints to 6-DOF using "eigenscrews" was presented in [Zhang 2012].

Proxy and linkage simulation A proxy is a completely simulated virtual object, which dynamic properties are defined according to the controller and the virtual guides requirements. The proxy is attached to the *ctg* by an elastic or viscoelastic virtual linkage as illustrated in Figure 1.24. If the *ctg* violates a constraint the virtual linkage will generate a force.

Proxies are the foundation of some more complex methods such as: *implicit-force* and *modified-damping control* [Ho 1995], *virtual mechanisms* [Joly 1995] and *pseudo-admittance control* [Abbott 2007]. *Implicit-force* and *modified-damping control* are two proxy-based algorithms where a proportional-derivative position controller is used to drive the *ctg* towards the desired position/proxy. The difference between the

implicit force and *modified-damping* variants lies in how the velocity of the proxy is computed.

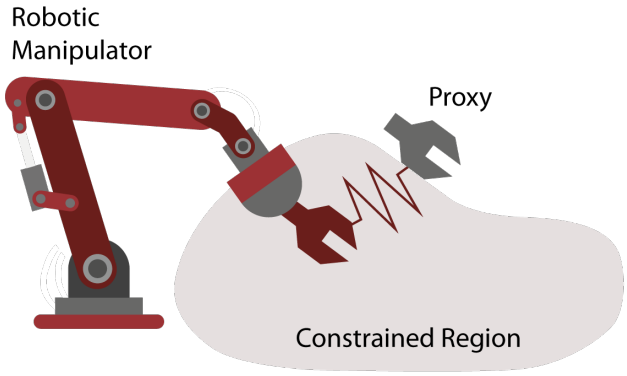


Figure 1.24: Proxy and linkage simulation for virtual guides enforcement. An elastic linkage between the robot end-effector and the proxy is simulated.

Virtual mechanisms were formulated by [Joly 1995] as a way to constrain motion in teleoperation tasks. This approach simulates the kinematics of an ideal mechanism – with no mass – attached to both the master and slave arms via virtual linkages. A composition of Virtual mechanisms was used later so the dynamics of master and slave motions, in both constrained and unconstrained directions, are fully controllable [Micaelli 1998].

Pseudo-admittance control was developed for admittance controlled robots to obtain quasi-statistic transparency and tremor attenuation. To achieve this, [Abbott 2003] conceived a proxy-based method using three control laws to define the forces applied to master and slave devices and the simulated proxy. The master forces are based on its separation from the slave device, the slave forces are based on its separation from the proxy, and the proxy forces are based on the error between the master and slave. In [Pezzementi 2007b] this method was used to apply velocity virtual guides rather than just position constraints.

Non-energy storing virtual guides A limitation in common proximity-based approaches is that they require at least a small amount of constraint violation before any motion regulation is applied. Moreover, many enforcement techniques lead to potential energy storage on the constraint level when the *ctg* is forced to move against them. *Simulated plasticity* was proposed by [Kikuuwe 2008] so constraints dissipate and never store energy. However, the system could become instable when the tool crosses the constraint boundary. A solution to this drawback was proposed in [Kikuuwe 2006].

Potential fields Potential fields were initially proposed in robotics as anti-collision approaches [Khatib 1986]. For virtual guiding proposes, potential fields are used by allocating low potential values to the areas where the robot is actively encouraged to

be – targets –, and allocate high potential values to the restricted areas – obstacles. By calculating the negative gradient of the field, a force vector can be extracted for robot positions which will point towards targets and away from obstacles. When this force is applied to the robot, it generates motion in the desired directions. Later research by [Turro 2001a] showed that potential field virtual guides could be used in teleoperated systems with force-reflecting master devices. Other works have been done in microscale telemanipulation [Ammi 2007] and in intracellular injection [Ghanbari 2010] applications.

Reference direction virtual guides In this enforcement method, a compliance matrix is used to force the robot to be more compliant when moved in a direction that respects a constraint and less compliant when not. For a given time and position, the entire space of possible directions of motion is divided into two complementary subspaces: "preferred" (i.e., those which correspond to a direction permitted by the active constraints) and "non-preferred" (i.e., those which correspond to a direction restricted by the active constraints), as shown in Figure 1.25. The preferred directions can be computed in a variety of different ways, allowing different constraints to be represented, such as *guidance constraints* [Bettini 2001, Prada 2009, Marayong 2003] and *regional constraints* [Burghart 1999b, Xia 2008].

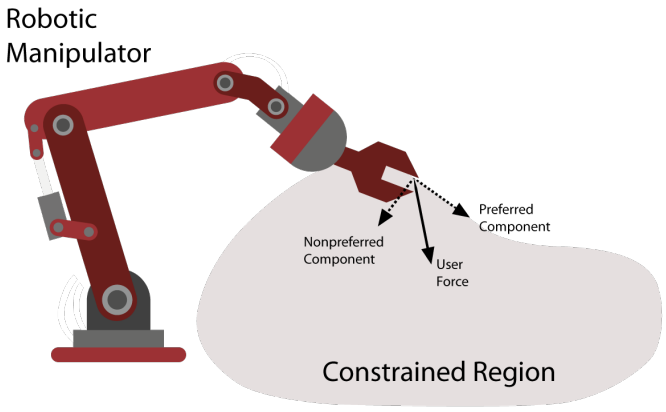


Figure 1.25: Reference direction virtual guides. The force of the user is decomposed into preferred – permitted by the constraint – and nonpreferred – resisted by the constraint – components.

Conclusion The virtual guide enforcement method used in this thesis falls in the *proxy and linkage simulation* category. The concept of virtual mechanisms proposed by [Joly 1997] are used to enforce virtual guides assistance for comanipulation tasks. More details about our virtual guides implementation are given in Chapter 2: "Virtual Guides Definition via Virtual Mechanisms".

1.2.4 Virtual Guides Modification

It was stated above that Programming by Demonstration could provide a natural and intuitive way to create virtual guides. However, the previously mentioned *PbD* approaches cannot be modified both interactively and locally by the user. If the task or the environment change, the user has to do a new set of demonstrations with the robot to obtain a new task representation. In a *cobot* comanipulation context, it could be exhausting and painful to repeat several times a demonstration in order to encode a task or modify it. Moreover, it would be contradictory to our approach since we look forward using *PbD* techniques to program virtual guides which will be used to assist a human operator on the execution of a task.

Adaptive approaches *Adaptive virtual guides* methods have been proposed in the literature to allow the generalization of a task to unknown environments or situations that have not been explicitly preprogrammed on the robot. The controller decides when and how to apply the constraints based on some knowledge of the task, of the hardware or based on some previous information of the users. For example, Hidden Markov Models (HMM) have been used to classify and interpret users' intention. Some applications of adaptive virtual guides are presented by [Rozo 2014] and [Aarno 2005]. In the first approach, this is applied on tasks where initial and end points are more relevant than the trajectory itself. In the second, the fixtures are made flexible and adaptive by decomposing the trajectory into straight lines. The probability that the user is following a certain trajectory is estimated and used to automatically adjust the compliance of the virtual guide. However, in both approaches, several demonstrations are needed to learn the first trajectory which could be tedious for the human operator. Moreover, the robot decides when and how to modify the trajectory which could be counter-intuitive. An emerging approach, known as *Dynamic active constraints* proposes to continuously modify the virtual guide geometry as a result of changes in the physical environment or task being undertaken. Much of the literature has focused on applications for surgery [Taylor 2016] since additional material such as vision systems are needed to update the model of the virtual guide. For this reason, we do not explore this approach.

Incremental approaches *PbD* incremental learning approaches have been used to reduce the number of demonstrations by gradually refining the task knowledge as more examples become available. These incremental learning methods use various forms (verbal and non-verbal) to guide the attention of the robot to the important parts of the demonstration or to particular mistakes produced during the reproduction of the task [Calinon 2007b]. Such incremental and guided learning is often referred to as *scaffolding* or *moulding* the knowledge of the robot. Usually, these methods are used for learning complex tasks within the household domain from as few demonstrations as possible [Saunders 2006]. In previous work Appendix

A: "Co-manipulation with a Library of Virtual Guiding Fixtures", we applied incremental training of *GMM* to create virtual guides from data obtained with kinesthetic teaching and iteratively refine them [Raiola 2017a]. However, these kind of approaches are useful for task refinement but not suitable for partial modifications of the task. For example, we noted in [Raiola 2017a] that if a demonstration was too different from a previous one the algorithm will consider it as a new virtual guide and the previous guide would not be modified. Also, the refinement depends on a not user-intuitive parameter such as a likelihood threshold. In this thesis, we inspire from these incremental learning approaches and keep the idea of **iterative refinement**. Yet, **we also seek for an interactive and more intuitive method that also allows local partial modifications of the task**.

Interface-based approaches The approach presented by [Boy 2007], introduced the concept of *collaborative learning* to design ergonomic virtual guides to a tricycle *cobot*. Motion constraint can also be adapted to changes in the environment. In this method, *PbD* is initially used to teach the *cobot* a path to follow. A dedicated Graphical User Interface (GUI) path editor is provided on a PC for off-line definition and modification of guide paths. Similarly, it was suggested in [Mollard 2015] to improve the interaction protocol in a *PbD* context by including a GUI in the programming loop to show to the user the learned information. A relevant difference between the approaches of [Boy 2007] and [Mollard 2015] is that the latter intends to optimize the learning of a task aimed to be automatically reproduced by the robot, while the former approach uses the operator **not only as a part of the teaching phase but as a part of the task execution**.Thus, motion guidance is not implemented in [Mollard 2015]. On the other hand, in [Boy 2007] it is suggested that only *PbD* is needed to design ergonomic guide paths for driving a tricycle *cobot*. Conversely, the results of a user study presented in [Mollard 2015], confirmed the interest of combining both processes to reduce the overall programming time and effort and to increase precision. Indeed, there is a important complexity gap between both tasks, which can explain why the *GUI* was more influent on a specific task.

Assisted approaches Inspired from the idea of *collaborative learning* and taking into account the context of this thesis, we propose a programming framework to assist the user throughout the teaching process. To this aim, we suggest to **use virtual guides programming in an iterative way so only one demonstration without assistance is needed**. Under this point of view, the work in [Tykal 2016] appears to be much closer to our motivations. This method was proposed to ease *kinesthetic teaching* by combining the idea of incremental learning through warping several demonstrations with virtual tool dynamics [Kosuge 1995] to assist the user during teaching. However, the assistance is gradually increased based on the demonstrations accumulated up to that moment. Therefore, several demonstrations are still needed to refine the task before having the correct assistance. Moreover, after each iteration, it is the robot who chooses an assistance and not the operator who

decides where to refine the trajectory, which, might be counterintuitive to the human. In this thesis, we suggest that the operator be the master of the programming and decides when and where a trajectory modification has to be done. Also, the approach presented in [Tykal 2016], as other *PbD* incremental methods mentioned above, are conceived for task refinement but not for partial modifications.

Our iterative and assisted programming approach could also be used to reduce the cognitive load of users and increase their comfort during the teaching process. Complex trajectories in space, including cartesian position and orientation of the tool, require high levels of concentration and effort, which could lead to trajectory encoding errors while using the kinesthetic teaching technique. Thus, we also propose to **simplify the programming process by uncoupling cartesian positions and orientations during the demonstrations**. To this aim, we should first address the matter of n-dimensional virtual guides creation according to our constraint definition. Since more details on our virtual guides implementation and evaluation are given in Chapter 2 and at the beginning of Chapter 3, we will present the related works on 6-*DOF* constraints definition in Chapter 3: "Virtual Guides Construction".

1.3 Contributions

In the literature review from the previous Section, we identified the following challenges in the use of Virtual Guides in the context of human-robot comanipulation:

1. *Human-Robot Interaction* improvement and roles allocation.
2. Limited use of *Virtual Guides Assistance*.
3. Enhancement of the *flexibility of Virtual Guides Programming*,

This thesis presents a novel **Virtual Guides Programming Framework** that allows the human worker to create and modify Virtual Guides by demonstration through an **iterative** method based on **kinesthetic teaching** and interpolation functions. Thanks to this approach, the human worker is able to iteratively modify the guides by directly manipulating the end-effector, making the process more intuitive plus reducing its painfulness. Our approach allows local refinement of virtual guiding paths through physical interaction with the robots. The user is able to modify a specific cartesian keypoint of the guide or re-demonstrate a portion. Our approach was also extended to 6D Virtual Guides, where *XSplines* are defined via Akima interpolation (for translation) and spherical cubic interpolation of quaternions (for orientation). The user can initially define a Virtual Guide and then use the assistance in translation to only concentrate on defining the orientations along the path. We demonstrated that these innovations provide a novel and intuitive solution to increase the human worker's comfort during the programming and execution of human-robot comanipulation tasks, in two industrial scenarios with a collaborative robot.

Publications Some of the contributions of this thesis have been published in an international conference and a journal, listed hereafter:

S. Sánchez Restrepo, G. Raiola, P. Chevalier, X. Lamy, and D. Sidobre. *Iterative virtual guides programming for human-robot comanipulation.* IEEE International Conference in Advanced Intelligent Mechatronics (AIM), 2017. Pages 219-226.

G. Raiola, **S. Sánchez Restrepo**, P. Chevalier, P. Rodriguez-Ayerbe, X. Lamy, S. Tliba, F. Stulp. *Co-manipulation with a library of virtual guiding fixtures.* Autonomous Robots Journal, Special issue on Learning for Human-Robot Collaboration, 2017. https://doi.org/10.1007/s10514-017-9680-7.

This doctoral work was rewarded with the **Demenÿ-Vaucanson Award 2016**.

More information about the scientific communications and supplementary material related to this doctoral thesis can be found at: susanres.fr.

1.4 Outline

The organization of this thesis is detailed hereafter :

Chapter 1 presented the context of our research: *programming of human-robot comanipulation tasks in industrial environments using virtual Guides with collaborative robots*. Through an analysis of the existing solutions and limitations of the approaches proposed by the literature, we justified the objectives of the thesis. The contributions of this doctoral work were also exposed.

Chapter 2 explains the extension of the virtual mechanisms concept as 6D Virtual Guides in a comanipulation context. The control scheme is presented and the stability is addressed. This forms the basis of our work.

Chapter 3 details the construction of 6D Virtual Guides using kinesthetic teaching and *XSplines* (translation and orientation), defined via Akima and quaternion interpolations. The control scheme presented in Chapter 2 is used for the 6D Virtual Guides enforcement.

Chapter 4 presents the concept of *Iterative Virtual Guides Programming Framework*, which allows the user to create and modify the 6D Virtual Guides – explained in Chapter 3 – in an intuitive way while being assisted by the robot.

In Chapter 5, we validate the interest of using Virtual Guides in a comanipulation context and we evaluate the impact of our *Iterative Virtual Guides Programming Framework*, with two different experiments.

Finally, Conclusions and Perspectives are detailed in Chapter 6.

Chapter 2

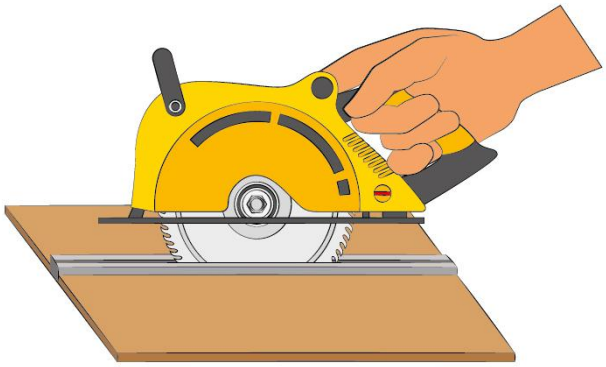
Virtual Guides Definition via Virtual Mechanisms

Contents

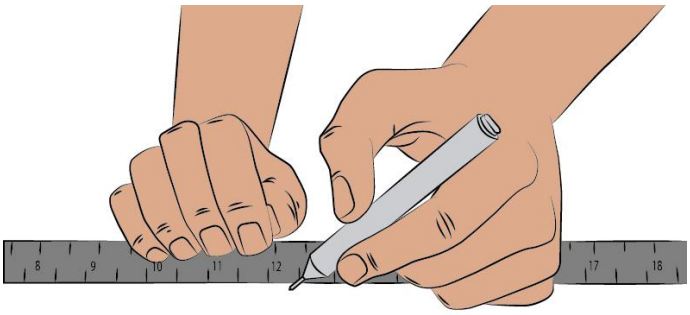
This chapter details the theory of Virtual Mechanisms for the implementation of Virtual Guides Assistance. In Section 2.1 the general definition of Virtual Mechanisms is presented and then implemented in Section 2.2. In Section 2.3 the stability of the impedance controller is proven by proving its passivity via the analysis of the dissipated energy. We discuss in Section 2.4 the problem related to kinematic singularities of the Virtual Mechanism caused by the unpredictability of users demonstrations and a solution using Jacobian normalization.

2.1 Definition of Virtual Mechanisms

Virtual Guides Assistance allows the worker to relieve mental processing and physical effort for task accomplishment, reducing the level of concentration devoted to the task while increasing the human-robot interaction ergonomy. Virtual guides are functionally equivalent to fixtures in the real world. An example of a non-virtual, but real guide, is a straight edge system clamped to a wood panel to aid in making a perfectly straight cut with a circular saw (see Figure 2.1(a)). During both virtual and real guiding, the user is responsible for the progress of the task, and particularly benefits from the accurate positioning of the tool (cobot end-effector / circular saw) offered by the guide system (computer-generated guide / straight edge system), in the form of haptic feedback. In other words, when using the virtual guides, the cobot becomes a tool that improves human capabilities and enhances the task efficiency in terms of execution time, mental workload, safety and precision. A virtual guide can also be compared with a ruler (see Figure 2.1(b)), which helps the user to draw lines with minimal effort and high precision.



(a) Straight edge system to cut straight with a circular saw



(b) Ruler used as a guide to draw a straight line

Figure 2.1: Examples of real guides.

As explained in the previous Chapter, in this thesis we are interested in using

virtual guides to assist human operators during industrial comanipulation tasks with a cobot, both for task programming and task execution. To this aim, we define Virtual Guides through the concept of Virtual Mechanisms introduced in [Joly 1995]. Thus, the cobot end-effector is virtually connected to the virtual mechanism through a spring-damper system. The result is a confined motion of the cobot end-effector if the virtual mechanism possesses less *DOFs* than the cobot.

Figure 2.2: Direct manipulation of the *Cobomanip* cobot designed by CEA-List and Sarrazin to assist the operator during handling operation.

Originally, virtual mechanisms have been introduced to enhance safety in teleoperating robotic systems [Joly 1995]. They were used for the remote inspection of pipes during the maintenance of nuclear sites. The camera movements were constrained along the pipe through a cylindrical virtual mechanism. In this thesis, we use the same concept of virtual mechanisms but in a comanipulation context where the user manipulates the robot directly, as shown in Figure 2.2.

2.2 Implementation of Virtual Mechanisms

Virtual mechanisms can be implemented using multiple *DOFs* to constrain movement in 1D paths, surfaces or volumes [Joly 1997] (see Figure 2.3). In our work, we implement the concept of virtual mechanisms (as defined in 1.2) to constrain the movements of the cobot to a 6D path defined by position and orientation constraints obtained by kinesthetic teaching. The cobot end-effector and the virtual mechanism are coupled by a spring-damper system which corresponds to a *proportional-derivative* controller whose coupling gains are the stiffness K and the damping B, as shown in Figure 2.4. If the cobot end-effector moves, the virtual mechanism is pulled along the

path in the direction of the movement; also, the virtual mechanism pulls the cobot end-effector towards the path, since the linking acts in both directions. The general effect is that the cobot end-effector can be moved easily along the constraining path, but not away from it.

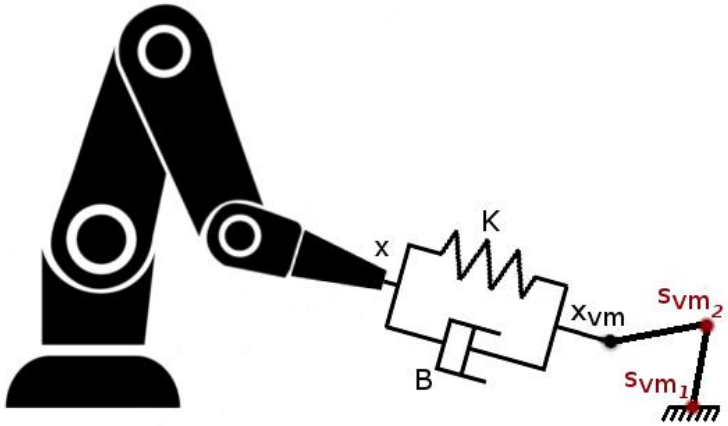


Figure 2.3: Virtual linking between a robot and the virtual mechanism using a spring-damper system. X_{vm} and X represent the pose in Cartesian space of the virtual mechanism and the robot end-effector respectively. The s_{vmi} represent the degrees of freedom of the virtual mechanism. For example, as illustrated here, with a 2 degrees of freedom mechanism the robot can be constrained to move in a cartesian plane.

Notation The Cartesian pose and velocity of the virtual mechanism and the cobot end-effector are described by $\{X_{vm} \in \mathbb{SE}(3), \dot{X}_{vm} \in \mathbb{R}(6)\}$ and $\{X \in \mathbb{SE}(3), \dot{X} \in \mathbb{R}(6)\}$ respectively. The Cartesian poses are defined by the translational and rotational components of both virtual mechanism and cobot end-effector displacement as:

$$X_{vm} \triangleq \begin{Bmatrix} X_{vm,trans} \in \mathbb{R}^3 \\ X_{vm,rot} \in \mathbb{SO}(3) \end{Bmatrix}$$

$$X \triangleq \begin{Bmatrix} X_{trans} \in \mathbb{R}^3 \\ X_{rot} \in \mathbb{SO}(3) \end{Bmatrix}$$

To avoid singularity in the representaiton of rotations $(X_{vm,rot}, X_{rot})$, we use

unit quaternions [Dam 1998]. A quaternion q can be considered to be the association of a scalar $w \in \mathbb{R}$ and a vector $\mathbf{a} \in \mathbb{R}^3$:

$$q \triangleq \begin{bmatrix} w \\ \mathbf{a} \end{bmatrix}$$

The relation between a quaternion and a rotation will be defined in the next chapter.

The Cartesian velocities are defined by twists $\in \mathbb{R}(6)$ describing the instant movements of the robot end-effector relative to the robot base. We choose to reduce the twists in the center of the robot end-effector. The twists are then formed by the translational and rotational components of the time derivatives of both virtual mechanism and cobot end-effector displacements:

$$\dot{X}_{vm} \triangleq \begin{Bmatrix} v_{vm} \in \mathbb{R}^3 \\ \omega_{vm} \in \mathbb{R}^3 \end{Bmatrix}$$

$$\dot{X} \triangleq \begin{Bmatrix} v \in \mathbb{R}^3 \\ \omega \in \mathbb{R}^3 \end{Bmatrix}$$

The current position of the virtual mechanism is described by its parameterized space by the parameter $s_{vm} \in \mathbb{R}$, and the evolution of the virtual mechanism is described by $\dot{s}_{vm} \in \mathbb{R}$.

The direct geometric and kinematic models of the virtual mechanism are defined by L_s and J_s respectively. The geometric model allows to determine the pose of the virtual mechanism X_{vm} according to the configuration of its links (in this case represented by s_{vm}), while the kinematic model allows to determine the velocity of the virtual mechanism $\dot{X}_{vm}$ according to the evolution of it $\dot{s}_{vm}$.

Geometric model:

$$X_{vm} = L_s(s_{vm}) \tag{2.1}$$

Kinematic model:

$$X_{vm} = f(s_{vm}) \tag{2.2}$$

$$\dot{X}_{vm} = J_s \dot{s}_{vm} \tag{2.3}$$

where $J_s[6\times1]$ is the virtual mechanism's Jacobian, as defined in [Wisama Khalil 1999].

In Chapter 3: "Virtual Guides Construction" we will describe how to implement the geometric and kinematic models through users demonstrations in both

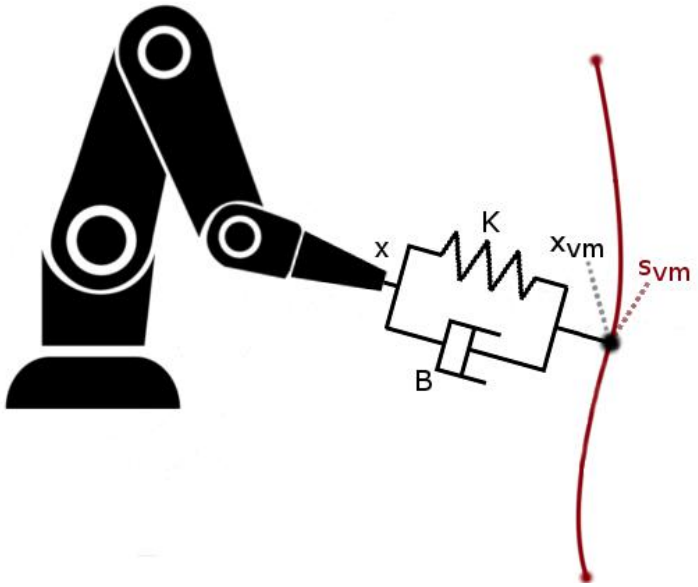


Figure 2.4: Virtual mechanism representation. The red curve represents the possible configurations of the virtual mechanism in the Cartesian space X_{vm}, and because of the spring-damper system linking, it represents the allowed configurations of the robot end-effector X. The current position of the virtual mechanism is described by its parameterized space by the parameter $s_{vm} \in \mathbb{R}$

translational and rotational components.

Force on the cobot end-effector The force F_c applied by the spring-damper system on the cobot is given by:

$$F_c = K(X_{vm} - X) + B(\dot{X}_{vm} - \dot{X}) \tag{2.4}$$

Where $K \in \mathbb{R}(6 \times 6)$ and $B \in \mathbb{R}(6 \times 6)$ are diagonal matrix. $K_{trans} \in \mathbb{R}(3 \times 3)$ represents the stiffness gain in translation and is a symmetric definite positive matrix.

The notation $(X_{vm} - X)$ may be abusive for rotations. More adequate angular error computation can be used without restrictions.

$$X_{vm} - X \triangleq \begin{Bmatrix} X_{vm,trans} - X_{trans} \\ \delta(q_{vm}, q) \end{Bmatrix}$$

We will give more details about the orientation representation on Chapter 3: "Virtual Guides Construction".

Gains tuning – K and B – is similar to the tuning of a cartesian PD position loop.

The Jacobian of the cobot is given by J $[6 \times n]$ where n represents the number

of *DOF* of the robot. We use the transposed Jacobian J^T to transform the forces into a torque reference for the controller [Wisama Khalil 1999]. The torque applied to the joints of the cobot is described by τ_c:

$$\tau_c = J^T F_c \tag{2.5}$$

Force on the virtual mechanism The behavior of the virtual mechanism impedance is given by:

$$\tau_{vm} = K_s(s_{cons} - s_{vm}) + B_s(\dot{s}_{cons} - \dot{s}_{vm}) \tag{2.6}$$

Where $s_{cons} \in \mathbb{R}$ and $\dot{s}_{cons} \in \mathbb{R}$ represent the reference position and the velocity along the desired path, respectively. The gains $K_s \in \mathbb{R}$ and $B_s \in \mathbb{R}$ represent a stiffness-damping coupling and define the impedance of the virtual mechanism.
Gains tuning – K_s and B_s – is similar to the tuning of a cartesian PD position loop and is done independently from the tuning of the coupling gains – K and B.
The virtual mechanism being ideal, the efforts applied on it are null. The equilibrium of the applied forces is given by:

$$J_s^T F_c = \tau_{vm} \tag{2.7}$$

Virtual boundaries constraints The purpose of "*virtual stops*" is to limit the permissible displacements of the tool. They can be defined in a complementary manner to a virtual mechanism, for example imposing limits on the travel of the links that constitute it. More generally, they can be described by surfaces delimiting an area of the working space in which the effector must remain confined.

In our virtual guides implementation, it is possible to specify the stiffness-damping coupling – K_s, B_s – between the virtual mechanism specification and a reference position s_{cons} along the desired path (see Figure 2.6). It is then possible to create virtual boundaries at the extremities of the path by applying to s_{cons} the following law:

- If $s_{vm} \in [0, s_{max}]$ then $s_{cons} \leftarrow s_{vm}$.
- If $s_{vm} > s_{max}$ then $s_{cons} \leftarrow s_{max}$.
- If $s_{vm} < 0$ then $s_{cons} \leftarrow 0$.

With these boundary constraints the user will feel a repulsive force (spring-damper effect) when he/she reaches the beginning or the end of the path.

Control law Using equations (2.3), (2.4), (2.6) and (2.7), we obtain:

$$J_s^T(K(X_{vm} - X) + B(J_s\dot{s}_{vm} - \dot{X})) = -B_s\dot{s}_{vm} + K_s(s_{cons} - s_{vm}) + B_s\dot{s}_{cons} \tag{2.8}$$

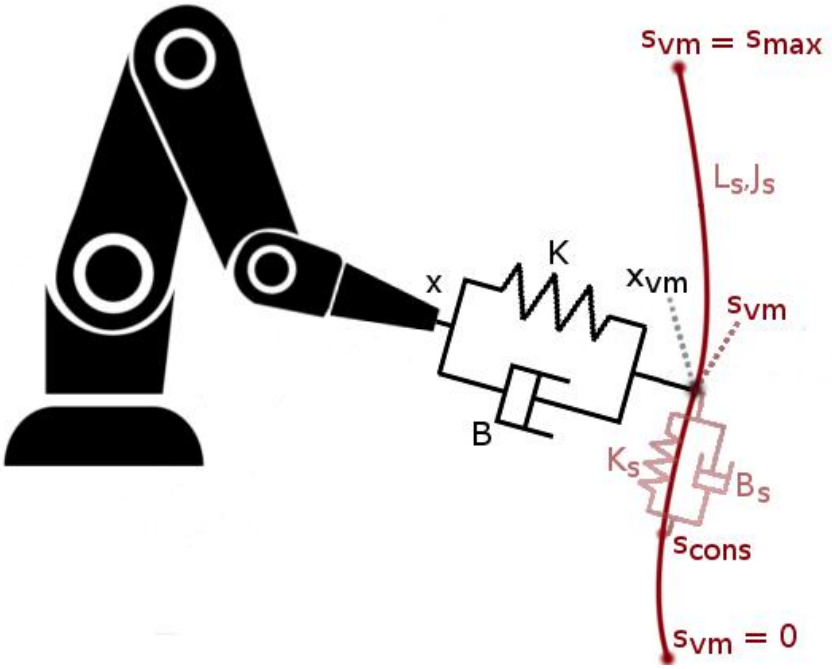


Figure 2.5: Lateral view of the physical analogy of a virtual mechanism. The desired path is illustrated in red. The pose X of the cobotic system is linked to the pose X_{vm} of the virtual mechanism by a spring-damper system. The current position of the virtual mechanism is s_{vm}.

By solving equation (2.8) with respect to $\dot{s}_{vm}$, we obtain a first order dynamical system that expresses the evolution of the virtual mechanism:

$$\dot{s}_{vm} = (B_s + J_s^T B J_s)^{-1}(-J_s^T(K(X_{vm} - x) - B\dot{X})) + K_s(s_{cons} - s_{vm}) + B_s\dot{s}_{cons} \tag{2.9}$$

The matrix $J_s^T B J_s$ is symmetric, positive, definite because J_s has full column rank and B is symmetric, positive, definite.

s_{vm} can be determined at every instant by integrating the controller state equation (2.9) in real time. Then, since s_{vm} and $\dot{s}_{vm}$ are known, X_{vm} and $\dot{X}_{vm}$ can be computed based respectively on equations (2.2) and (2.3). Finally, the driving force can be computed using (2.4).

From equations (2.1), (2.3), (2.4), (2.5), (2.6) and (2.9) we obtain the control law scheme presented in Figure 2.6.

The gains specifications of the coupling – K and B – are independent from the virtual guide specification – L_s, K_s and B_s. Gains tuning is similar to the tuning of a cartesian PD position loop. The higher the gains, the more the behavior of the robotic system will tend to the one defined by the virtual mechanism. In general,

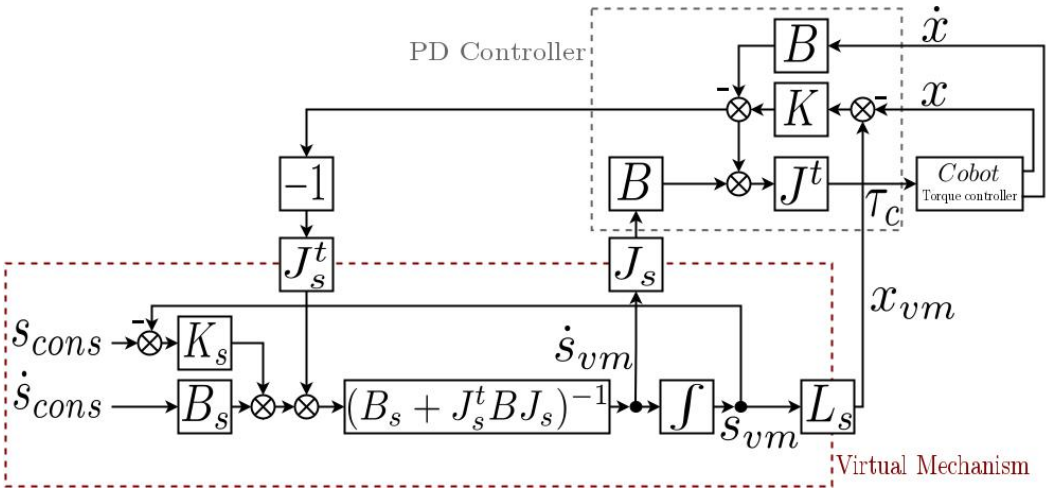


Figure 2.6: Control law scheme of a 1-*DOF* virtual mechanism.

the spring K is chosen as stiff as possible.

2.3 Stability during Interaction

In comanipulation tasks, robots and humans work hand in hand sharing their workspace. In this case safety is a major issue. According to [Van Damme 2010], the *safety* of a robotic system is mainly enforced through three different factors:

- *Safe element design* inherent from the mechanics of the robot : for example the use of padded and rounded link shapes to avoid injuries caused by accidental contacts with the robot, and low weight components to reduce impact forces in eventual collisions.

- *Passive compliance* devices to allow the robot to be more compliant (i.e. less stiff) thus more safe. In case of contacts with the environment or with humans, part of the energy is absorbed by the robot. Some examples are: torque/velocity limitation devices, elastic actuators [Pratt 1995] or variable stiffness actuators [Tonietti 2005].

- *Active compliance.* Compliance can also be ensured by the control law applied to the robot. According to [Siciliano 2009], there are two main kind of *compliant* controllers: *force* and *impedance* controllers.

In the impedance control scheme presented above, a virtual spring-damper system is used to control the robot movements through a virtual mechanism previously defined. This controller generates forces for the robot end-effector based on the position and velocity of the virtual mechanism. In this section, we study the passivity of this controller to assure that it does not lead to the instability of the

robotic system [Khalil 1996]. This represents an important step to prove that the co-manipulation with virtual guides is *safe*.

Stability proof using a passivity criterion It was proven by [Hogan 1988] that the passivity of the system guarantees the stability of the controlled system when it interacts with any passive environment, including a human operator.

In principle, a human operator is able to supply energy to a mechanical system, thus it would not be passive. However, the results presented in [Hogan 1989] present another point of view : humans are able to modulate the mechanical impedance of their hands according to the task they want to accomplish, thanks to the co-contraction of the antagonistic muscles distributed on each of the articulations, and thanks to the richness of the postures that they can confer on their shoulder, arm and wrist.

The variability of the impedance that the hand can adopt is very important: it is found that the resulting stiffness in the hand can evolve over at least two orders of magnitude. In his experiments, [Hogan 1989] notes that all the impedances represented by the hand have as a common point to be passive. It is also found that the man is able to adapt the impedance of his hand with a delay of about one second.

[Colgate 1988] introduced a passivity criterion to ensure a stable interaction between the controlled robot and an unknown and variable environment. A system is considered passive when it does not provide more energy than it has received. In the time domain, this property is often reflected as:

$$\forall t \geq t_0 \geq 0, \quad \int_{t_0}^{t} P_e(\tau) d\tau \geq E_{cobot}(t) - E_{cobot}(t_0) \tag{2.10}$$

where P_e is the power supplied to the controller and E_{cobot} is the internal energy of the cobotic system at the instant t. This equation reflects the idea that the cobot cannot give to the environment more energy than it stored at the instant t_0, when cutting the supply energy, the system should evolve towards a minimum energy state.

The advantage of this criterion is that the stability of the system in interaction with any passive environment can be proven by only taking into account the system, regardless of the environment. The passivity of the virtual mechanism controller was proven by Joly in [Joly 1995] by using the mechanical analogy of the system and studying the energy dissipation.

Passivity proof Considering the system in Figure 2.4, the supplied power of the controller is given by:

$$P = \tau_{vm}^T \dot{s}_{vm} - \tau_c^T \dot{q}, \tag{2.11}$$
$$= F_c^T \dot{X}_{vm} - F_c^T \dot{X}. \tag{2.12}$$

Where q represents the joint velocities of the cobot.

$$P = F_c^T(\dot{X}_{vm} - \dot{X}), \tag{2.13}$$
$$= (K(X_{vm} - X))^T(\dot{X}_{vm} - \dot{X}) + (\dot{X}_{vm} - \dot{X})^T B(\dot{X}_{vm} - \dot{X}). \tag{2.14}$$

By integrating both sides of the equation above, we obtain the following energy balance:

$$\underbrace{\int_0^t P\,dt}_{\text{supplied energy}} = \underbrace{E_{pot}(t) - E_{pot}(0)}_{\text{stored spring energy}} + \underbrace{\int_0^t (\dot{X}_{vm} - \dot{X})^T B(\dot{X}_{vm} - \dot{X})\,dt}_{\text{dissipated energy}}. \tag{2.15}$$

In this equation E_{pot} represents the potential energy associated to the spring K. The controller is proven passive because both E_{pot} and the dissipated energy are positive (B is positive definite), implying that the controller do not supply more energy than the initial one (as defined in equation 2.10):

$$\int_0^t P\,dt >= -E_{pot}(0). \tag{2.16}$$

We invite the interested reader on the passivity of the system to consult the more detailed proof presented in [Joly 1997].

2.4 Kinematic Singularities

By looking at the dynamical system of equation (2.9), it is clear that the term $({J_s}^t B J_s)$ has to be invertible in order to avoid singularities on the evolution of the virtual mechanism. In other words, we must guarantee for all values of s_{vm} that the Jacobian J_s is never null. In our implementation, the gain B_s used (together with K_s) to create virtual boundaries, avoids the zero division in the evolution law of the virtual mechanism. However, since the Jacobian J_s is created through user demonstrations (as will be explained in Chapter 3: "Virtual Guides Construction"), if we use *PbD* recording time as the Virtual Guides parametrization the Jacobian represents the velocity of the virtual mechanism and it is possible that it is not normalized. These velocity variations could affect the users interaction with the virtual mechanism. To overcome this problem, we propose to *normalize* the kinematic model of the virtual mechanism.

Jacobian Normalization We can parametrize the kinematic model of the virtual mechanism using an approximation of the arc-length $l_{vm} \in \mathbb{R}$ of the path that defines the virtual mechanism, instead of the phase $s_{vm} \in \mathbb{R}$ which could depend on the – variable – sampling time of the demonstrations.

Thus, we can replace the phase s_{vm} with the arc-length approximation l_{vm}, to rewrite the kinematics of the Virtual Mechanism as:

$$X_{vm} = f(l_{vm}), \tag{2.17}$$
$$\dot{X}_{vm} = J_s(l_{vm})\dot{l}_{vm}, \tag{2.18}$$
$$\text{with} \quad ||J_s(l_{vm})|| = 1. \tag{2.19}$$

In Chapter 3: "Virtual Guides Construction" we will explain how to define the arc-length approximation for both translational and rotational components of Virtual Guides. A constraint model coupling both movement components will be presented together with the corresponding Jacobian normalization. Finally, the geometric and kinematic models of 6D Virtual Guides will be defined.

2.5 Conclusion

In this chapter, we presented how the virtual mechanisms proposed by [Joly 1995] can be used to implement Virtual Guides as 6D paths, for comanipulation applications. These virtual mechanisms work as an impedance controller for the robot, allowing movement along the preferred directions and prohibiting the movements along the restricted ones. The passivity of the system is proven by studying the dissipated energy of the system which proves also its stability. We also presented the problem of singularities of the kinematic model of the virtual mechanism when the virtual guides are defined through users demonstrations. A solution using Jacobian normalization is proposed. In the following chapter we will see how Akima spline and quaternion interpolations can be used to describe the kinematics of Virtual Guides using demonstrations through kinesthetic teaching.

Chapter 3

Virtual Guides Construction

Contents

In the previous Chapter we explained how to use virtual mechanisms to implement Virtual Guides in a comanipulation context. In order to create these Virtual Guides, we have to implement the kinematic equations of the virtual mechanism described by (2.2) and (2.3). As explained in Chapter 1: "Introduction", there are two main ways to do it:

- through geometric modeling and
- through demonstrations.

Geometric modeling is the most common method in the literature. It was used by [Joly 1995] in the context of teleoperation tasks. An important advantage of this method is that it can lead to a precise definition of the shape of the virtual guide. However, to implement the kinematic equations of the virtual mechanism, this method implies a good knowledge and understanding of the task. In addition, the geometric model has to be defined in the operational space of the robot, which could require a transformation from the task reference to the robot reference and

though be less trivial. A most recent approach proposes to define the kinematics of Virtual Guides using a set of demonstrations, which could be provided by the user through kinesthetic teaching [Calinon 2007a]. Users are able to physically manipulate and guide the robot in order to perform the desired movements. These movements can be recorded by the robot and be used to define a model of the Virtual Guide. A great advantage of this method is that the definition is done directly in the human-robot workspace, which is ideal in a comanipulation context. Moreover, a non-expert user may specify new virtual guides through lead-through programming without any prior knowledge about the task or about its geometric model, as done in [Stulp 2013, Tykal 2016]. The *PbD* approach is also a promising solution to reduce the programming time of the tasks and enhance the flexibility of the Virtual Guides Assistance presented in Chapter 2: "Virtual Guides Definition via Virtual Mechanisms".

For these reasons and for the ones enlighted in Chapter 1: "Introduction", we use kinesthetic teaching to create Virtual Guides from demonstrations. In previous work Appendix A: "Co-manipulation with a Library of Virtual Guiding Fixtures", we proposed a framework for *Multiple Probabilistic Virtual Guides* where kinesthetic teaching was used in the programming process. This probabilistic framework involves modeling a demonstrated set of guides with *GMM* and retrieving a generalized representation of the data set using *GMR* [Raiola 2017a]. Unfortunately, with these stochastic approaches, a compromise must be made between the number of demonstrations – time and effort demanding – and the level of information in the training data. In a comanipulation context, requesting multiple demonstrations of a single task could be exhausting for the user. In this thesis, we propose to use an **iterative and assisted framework to program virtual guides through kinesthetic teaching, with only one demonstration and the possibility to refine or modify the guide later in an intuitive way**. This framework will be presented in Chapter 4: "Iterative Virtual Guides Programming".

In this Chapter, we will explain how to define the geometric and kinematic models of Virtual Guides using interpolation functions. We will first address the Virtual Guides construction as Position Constraints and as Orientation Constraints separately and then propose a coupled solution position-orientation to define *6D Virtual Guides* through *displacement splines*. The approaches presented in Chapter 2: "Virtual Guides Definition via Virtual Mechanisms" and in this Chapter, are the base of the programming framework in Chapter 4: "Iterative Virtual Guides Programming".

3.1 Geometric and Kinematic Models of Virtual Guides

When the user manipulates the cobot and creates a displacement of the end-effector, the force applied by the cobot on the virtual mechanism is expressed by a new value of the phase s_{vm} by integrating the dynamical system defined in equation (2.9). We can use the direct geometric model L_s (2.1) and the kinematic model J_s (2.3) to

compute the corresponding *pose* X_{vm} and *velocity* $\dot{X}_{vm}$ of the virtual mechanism for a given value of the phase $s_{vm} = s$. These models are defined in this thesis through *interpolation functions* and their derivatives.

In the following Sections of this Chapter, we will describe the interpolation functions used for both Position Constraints – Section 3.2 – and Orientation Constraints – Section 3.3 – construction. We will then propose a common parametrization to generate *6D Virtual Guides* through *displacement splines* in $\mathbb{SE}(3)$ – Section 3.4.

3.2 Virtual Guides as Position Constraints

In this Section we address the Virtual Guides construction as position constraints, thus we only focus on the translations of the cobot end-effector $X_{trans} \in \mathbb{R}^3$ (the orientation of the end-effector being fixed or free). During the kinesthetic teaching, the user is able to show the cobot the desired trajectory by manually moving its end-effector. The Cartesian position X_{trans} of the cobot end-effector is recorded on user demand (e.g. when he or she pushes a button to record the current robot end-effector position) or continuously with a determined sampling time. In both cases, the position X_{trans} and a recording parameter t are stored as a list of points M_T:

$$M_T(t_i) = \{t_i, (x_{trans,i}, y_{trans,i}, z_{trans,i})\}_{i=0:N-1} \tag{3.1}$$

where N is the number of points.

When the recording of the trajectory is done by user demand, the recording parameter starts at $t_0 = 0$ and increases monotonically along with the recorded points. When the recording is done continuously, the recording parameter corresponds to the recording time. Thus, in both cases, the parameters t_0, t_1, $t_2 ... t_{N-1}$ satisfy a monotonic condition:

$$t_0 < t_1 < t_2 ... t_{N-2} < t_{N-1} \tag{3.2}$$

In order to create a Virtual Guide using the list M_T, we propose to use an interpolation function that passes exactly through the recorded points so the demonstrated trajectory is encoded in a precise and smooth way. As suggested in Chapter 1: "Introduction" and as we will see in Chapter 4: "Iterative Virtual Guides Programming", using interpolation functions to describe the geometric and kinematic models of Virtual Guides presents the following advantages:

- Only one demonstration without Virtual Guides Assistance is needed.
- The recorded trajectory depends entirely on the demonstration and not on any automatic algorithm.

- Virtual Guides defined through interpolation functions can be easily modified.

Next, we present some interpolation methods, among them the Akima Spline interpolation method that will be used in this Chapter for Virtual Guides construction in Cartesian space.

3.2.1 Interpolation in R^3

To build a path through several waypoints, different solutions are available. It is for example possible to *interpolate* these points or to *approximate* them. In addition, depending on the type of interpolation or approximation, different properties, such as continuity of derivatives, may vary among the built paths. A non-exhaustive list of these methods and their properties is presented below.

Bezier curves This is an approximation method that uses control points instead of interpolation points, so the created curve does not pass through the specified waypoints but approaches them (see Figure 3.1). The degree of these curves is proportional to the number of control points and can increase very rapidly.

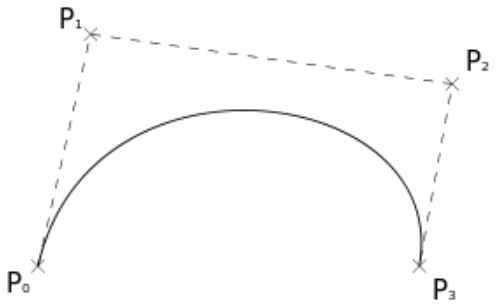


Figure 3.1: Example of a Bezier curve with 4 control points: $P(t) = P_0(1-t)^3 + 3P_1t(1-t)^2 + 3P_2t^2(1-t) + P_3t^3$

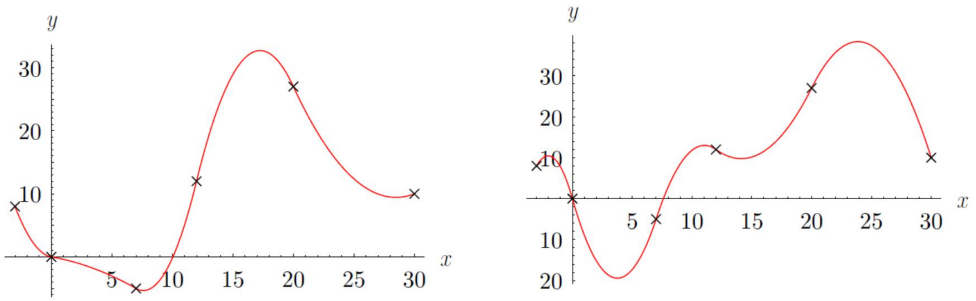
B-Splines This is also an approximation method [De Boor 1972]. It allows to overcome some disadvantages of the Bezier curves. For example, the degree of the curve remains constant even when adding a point. Also, it is possible to make a local modification of the curve without changing its natural aspect. With B-Splines, it is possible to interpolate the waypoints by playing on the degree of the knots. However, B-Splines imply complex calculus of the basic functions, and it would be less intuitive for the user to manage the knots and the waypoints at the same time in order to modify the curve.

Quadratic interpolation This is one of the simplest methods for interpolating points. It consists in connecting the waypoints by pairs using two degree polynomials.

In order to obtain a smooth interpolation, two conditions are imposed on each curve connecting two waypoints:

- The curve must go through the two points that it connects.
- The tangent to the first point must be equal to the tangent to the second point of the previous curve (the first slope of the first curve is arbitrarily defined).

The main disadvantage of this method is that depending on the choice of the initial slope, the interpolation can be completely different, as shown in Figure 3.2. In addition, this interpolation is not "natural", the created curve does not correspond to what the user might expect.



(a) Quadratic interpolation with initial slope $m_0 = -5$.

(b) Quadratic interpolation with initial slope $m_0 = 5$.

Figure 3.2: Example of the quadratic interpolation with different initial slopes.

Cubic interpolation Unlike *Quadratic interpolation*, this method involves curves of degree three. This makes it possible to impose an additional condition: the tangent to the second point. Different types of cubic interpolations make it possible to calculate the slopes according to the desired characteristics of the curve. General requirements are function continuity and passing through all given points. There could also be additional requirements such as continuity of higher derivatives.

The most common cubic interpolation methods are: natural, monotonic and Akima. *Natural cubic interpolation* is a piecewise cubic curve with continuous second derivative. The resulting curve is piecewise cubic on each interval, with matching first and second derivatives at the supplied data-points. The second derivative is chosen to be zero at the first and last points. A disadvantage of this method is that the curves could oscillate in the neighborhood of an outlier, as shown in Figure 3.3. *Monotonic cubic interpolation* is a variant of cubic interpolation that preserves monotonicity of the data set being interpolated by modifying the tangents definition [Fritsch 1980]. This method does not present oscillations in the presence of outliers. The *Akima cubic interpolation* method [Akima 1970] is also based on piecewise cubic polynomials. It differs from the previous methods by the conditions imposed at the data points, which make the interpolation function robust to local

changes. In Figure 3.3 we show a comparison of these three cubic interpolation methods and a linear interpolation. We can see that the monotonic and the Akima curves present less oscillations than the natural curve.

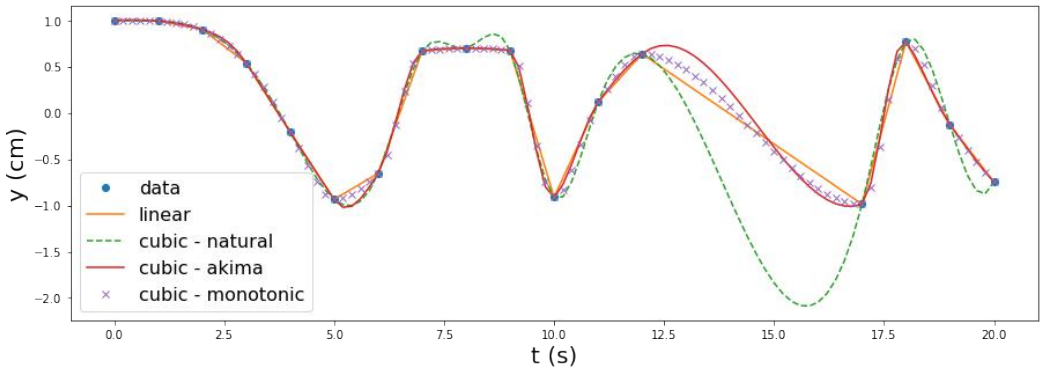


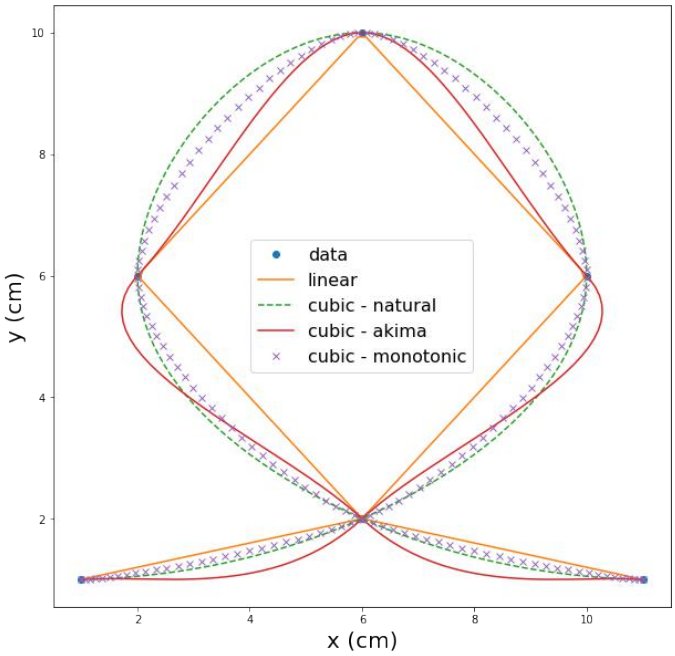
Figure 3.3: Comparison of different interpolation methods.

We can see in Figure 3.4 that when it comes to an interpolation between 4 points, the best interpolation method to obtain circles is the natural spline in comparison with the monotonic and the Akima curves. However, to create Virtual Guides by demonstration it is not desirable to use splines that could present oscillations. Also, when modifying a point on a natural cubic interpolation, the whole curve could be modified. For these reasons, from this point, we do not consider natural splines in our choice. Finally, in Figure 3.4(b), we can see that when we rotate the 4 given waypoints on the plane, by definition, the resulting monotonic curve resembles to the linear interpolation curve. On the other hand, the Akima spline is closer to the desired circular curve. Indeed, when creating Virtual Guides, it is important to avoid straight angles since they could generate undesirable behaviors of the control scheme.

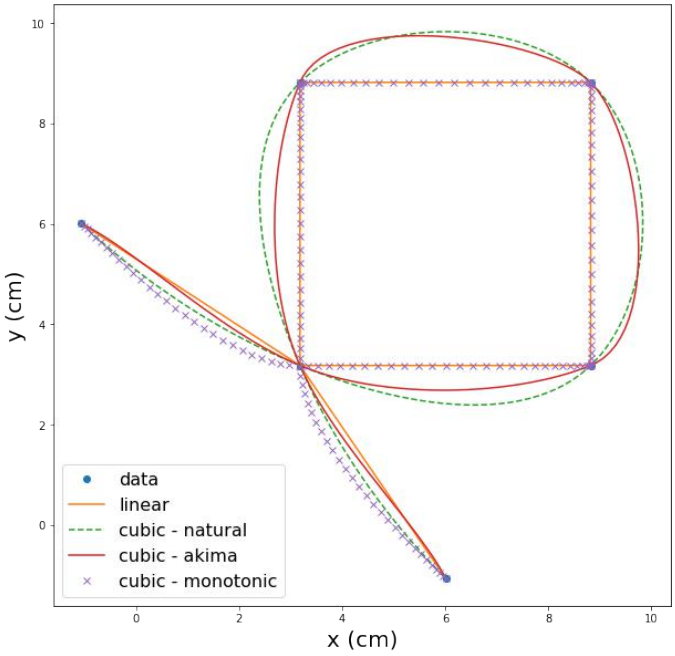
To resume, some advantages of the Akima spline interpolation method are:

- It yields to a smooth natural-looking curve.
- There is no need to solve large equation systems. It is therefore computationally very efficient.
- It reduces oscillatory effects.
- Local changes do not affect the interpolation beyond neighbor points.
- Akima spline points are intuitive to use in Virtual Guides modification.

Besides, the Akima spline interpolation method provides a curve of class C^1 at least, and as stated in [Joly 1997], to define virtual mechanisms it is possible to use any curve defined by a parametric function of class C^1. When it comes to interpolation, a compromise must be done between the smoothness of the curve and



(a) Interpolation of 4 waypoints. The natural cubic interpolation gives the closest resulting curve to a circle.



(b) Interpolation of 4 waypoints rotated on the plane. The monotonic interpolation spline resembles to the linear interpolation curve.

Figure 3.4: Comparison of different interpolation methods to create a circular curve.

the robustness to local modifications. For example, for path planning it is better to guarantee C^2 continuity in order to ensure continuous acceleration. For Virtual Guides implementation we need at least C^1 continuity and for our particular use of Virtual Guides we need for sure to locally modify the splines.

For all these reasons we decided to use Akima spline interpolation to define the geometric and kinematic models of the virtual mechanisms used to implement Virtual Guides. Next, we give more details of this interpolation method.

3.2.1.1 Akima Spline Interpolation

The Akima Spline interpolation method was defined by [Akima 1970]. It is a continuously differentiable sub-spline interpolation, built from piecewise third order polynomials and applicable to successive intervals of the given points. Only data from the next and previous two neighbor points are used to determine the coefficients of the interpolation polynomial. Thus, the slope of the curve is locally determined at each given point, by the coordinates of five points centered on the studied point. In consequence, this spline type creates a smooth and natural curve between the waypoints and always passes directly through them.

Slope The slope r of the curve at each given point is determined locally by the coordinates of five points, with the point in question as a center point, and two points on each side of it. A polynomial of degree three representing a portion of the curve between a pair of given points is determined by the coordinates of the two points and the local slope.

With the five data points 1, 2, 3, 4 and 5, the slope r of the curve at point 3 is determined by:

$$r_3 = \frac{(|m_4 - m_3|m_2 + |m_2 - m_1|m_3)}{(|m_4 - m_3| + |m_2 - m_1|)} \tag{3.3}$$

where m_1, m_2, m_3, m_4 are the slopes of the line segments $\overline{12}$, $\overline{23}$, $\overline{34}$ and $\overline{45}$, respectively as defined by (3.4).

$$m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \tag{3.4}$$

To determinate r, two assumptions are made:

- the slope of the curve at point 3 should approach that of the line segment $\overline{23}$ when the slope of $\overline{12}$ approaches that of $\overline{23}$,
- the slope is invariant under a linear-scale transformation of the coordinate system.

Under the definition (3.3), the slope r_3 of the curve at point 3 only depends on the slopes of the four line segments, being thus independent from the interval widths. Moreover, this definition implies that:

- $r_3 = m_2$ when $m_1 = m_2$ and $m_3 \neq m_4$.
- $r_3 = m_3$ when $m_3 = m_4$ and $m_1 \neq m_2$.

which are the desired conditions.

It also follows from (3.3) that, when $m_2 = m_3$, $r = m_2 = m_3$. However, when $m_l = m_2 \neq m_3 = m_4$, the slope r_3 is undefined under equation (3.3). In this special case r is chosen such that:

$$r_3 = \frac{m_2 + m_3}{2}$$

Interpolation between two points A portion of the curve between a pair of consecutive data points is defined in such a way that the curve will pass through the two points. At these points the slopes are determined by the procedure described above. For example, the third-degree polynomial for the line segment $\overline{12}$ is determined by:

$$y = p_0 + p_1(x - x_1) + p_2(x - x_1)^2 + p_3(x - x_1)^3 \tag{3.5}$$

where:

$$\begin{aligned} p_0 &= y_1 \\ p_1 &= r_1 \\ p_2 &= \frac{[3\frac{(y_2 - y_1)}{(x_2 - x_1)} - 2r_1 - r_2]}{(x_2 - x_1)} \\ p_3 &= \frac{[r_1 + r_2 - 2\frac{(y_2 - y_1)}{(x_2 - x_1)}]}{(x_2 - x_1)^2} \end{aligned}$$

The first derivate of the function is determined by:

$$\dot{y} = p_1 + 2p_2(x - x_1) + 3p_3(x - x_1)^2 \tag{3.6}$$

Estimation of extra points At each end of the curve, two more points have to be estimated from the given points. Let N be the number of given points. As presented in [Akima 1970], the extra points are estimated by assuming that all of the last three points (x_{N-2}, y_{N-2}), (x_{N-1}, y_{N-1}), (x_N, y_N) and the extra points (x_{N+1}, y_{N+1}), (x_{N+2}, y_{N+2}) lie on a curve expressed by:

$$y = g_0 + g_1(x - x_N) + g_2(x - x_N)^2, \tag{3.7}$$

where g_i are constants. Assuming that

$$x_{N+2} - x_N = x_{N+1} - x_{N-1} = x_N - x_{N-2},$$

Then,

$$\begin{aligned} x_{N+1} &= x_{N-1} + x_N - x_{N-2}, \\ x_{N+2} &= 2x_N - x_{N-2}. \end{aligned}$$

The ordinates y_{N+1} and y_{N+2} can be determined from (3.7), by using x_{N+1} and x_{N+2}, respectively. According to [Akima 1970], it gives:

$$\begin{aligned} \frac{(y_{N+2} - y_{N+1})}{(x_{N+2} - x_{N+1})} - \frac{(y_{N+1} - y_N)}{(x_{N+1} - x_N)} &= \frac{(y_{N+1} - y_N)}{(x_{N+1} - x_N)} - \frac{(y_N - y_{N-1})}{(x_N - x_{N-1})} \\ &= \frac{(y_N - y_{N-1})}{(x_N - x_{N-1})} - \frac{(y_{N-1} - y_{N-2})}{(x_{N-1} - x_{N-2})} \end{aligned}$$

Then,

$$\begin{aligned} y_{N+1} &= (2m_{N-1} - m_{N-2})(x_{N+1} - x_N) + y_N, \\ y_{N+2} &= (2m_N - m_{N-1})(x_{N+2} - x_{N+1}) + y_{N+1}. \end{aligned}$$

Next, we will see how to use the Akima spline interpolation definition to create *Position Constraints*.

3.2.2 Position Constraints Construction through Akima Splines

The above Akima Spline definition, *slope*, the *interpolation function* and the *extra points*, can be applied to the list of M_T points (3.1) where t values increase monotonically (3.2). In a first stage we extend the monodimensional definition of the Akima splines to multidimensional spaces.

3.2.2.1 Multi-dimensional Akima interpolation

A multi-dimensional Akima spline $MDSpline$ is composed of several Akima splines, each corresponding to an additional degree of freedom. For example, a $MDSpline$ representing a translation in Cartesian space will consist of three splines: one for x axis - $spline_x(t)$ -, one for y axis - $spline_y(t)$ -, and one for z axis - $spline_z(t)$. The synchronization of the axes in the Cartesian space is provided by their curve parameter.

Indeed, for each point, the three axis have the same corresponding parameter t. Thus, each waypoint will be reached at the same value of t:

$$x_{trans} = spline_x(t),$$
$$y_{trans} = spline_y(t),$$
$$z_{trans} = spline_z(t)$$

The first derivative of the interpolation function can also be calculated for each axis to compose $MD\dot{S}pline$, using equation (3.6).

3.2.2.2 Parameterization

If we use time as the parameter for the curve, the Jacobian J_s will correspond to the velocity of the virtual mechanism. As explained in Chapter 2: "Virtual Guides Definition via Virtual Mechanisms", since the trajectory is shown by demonstration, the velocity is variable and could be null. Jacobian variations could affect the user's interaction with the virtual mechanism. In order to guarantee a normalized Jacobian, it is desirable to evaluate the Akima spline at points based on the arc-length of the curve instead of based on the recording sampling time. For that matter, we propose to separate spacial and temporal aspects of the trajectory.

Let t be the time, s be the path-length curvilinear parameter and P_T the list containing the Akima spline points.

$$P_{T,i} = \{x_{trans,i}, y_{trans,i}, z_{trans,i}\}$$

with $i = 0 : N-1$, where N is the number of waypoints. When the points are recorded manually, $t = i$.

The Akima spline curve $MDSpline$ parametrized with time is defined by f:

$$f : \left| \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{R}^3 \\ t & \longmapsto & P_T \end{array} \right.$$

The transformation function from the the arc-length curvilinear to time parameterization is defined by g:

$$g : \left| \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{R} \\ s & \longmapsto & t \end{array} \right.$$

where g corresponds to a monotonic cubic interpolation function [Fritsch 1980]. This kind of interpolation is optimal for the transformation function since the resulting curve does not present oscillations in the presence of outliers.

We approximate the computation of s by:

$$s_{i+1} = s_i + \|P_{T,i+1} - P_{T,i}\|_2$$

where $s_0 = 0$ and $i = 0 : N - 1$.

The Akima spline curve is now defined as a composition of the initial curve parameterized with time (f), and the space transformation function (g) as:

$$f(t) = f(g(s)) = f \circ g(s) \tag{3.8}$$

The Virtual Guide is now described by the following data list:

$$M_T(t_i, s_i) = \{t_i, s_i, x_{trans,i}, y_{trans,i}, z_{trans,i}\}_{i=0:N-1}$$

3.2.2.3 Geometric and Kinematic models

We can now define the geometric model of the virtual mechanism L_s (2.1) as:

$$L_s : \left| \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{R}^3 \\ s & \longmapsto & MDSpline(s) \end{array} \right.$$

and:

$$X_{vm,trans} = MDSpline(g(s)) \tag{3.9}$$

We can notice that the parameter s_{vm} of the virtual mechanism corresponds to the spline parameter s. We can then write:

$$X_{vm,trans} = MDSpline(g(s_{vm})) \tag{3.10}$$

The kinematic model of the virtual mechanism J_s (2.3) can be defined as the derivative of the multi-dimensional spline $MDSpline$. Finally :

$$J_s(s_{vm}) = MD\dot{S}pline(g(s_{vm})) \tag{3.11}$$

and :

$$\dot{X}_{vm,trans} = J_s \dot{s}_{vm} \tag{3.12}$$

3.3 Virtual Guides as Orientation Constraints

3.3.1 Related Works

Programming by Demonstration literature In *PbD* approaches, multivariate Gaussians are widely used to encode robot behaviors. Such approaches do not provide

the ability to properly describe end-effector orientation, as the distance metric in the space of orientations is not Euclidean. In [Zeestraten 2017], the authors present an extension of common probabilistic imitation learning techniques to Riemannian manifolds. This work shows the importance of being able to represent end-effector orientations from users demonstrations, coupled with cartesian positions. However, it does not address virtual guides creation. Thus, the work of [Zeestraten 2017] would be useful to extend *Probabilistic Virtual Guides* [Raiola 2017a] to a more general framework using also orientation representation.

Virtual Guides literature Due to the task-dependent role of virtual guides, most experiments have tried to address the most general scenarios for their applications. Many of these applications mainly consisted in general tasks such as path following, targeting and object avoidance exercises in two dimensional environments [Bettini 2001], [Nolin 2003] and in more complex 3D environments [Prada 2005], [Kuang 2004], [Raiola 2017a]. Fewer applications have addressed the implementation of orientation virtual guides [Li 2007, Bowyer 2013].

Traditional virtual guides implementations deal with rotation and translation separately in $\mathbb{R}^3$ space, the interconnection between them is not usually involved in the virtual guide design. The method presented in [Li 2007] uses *preferred directions virtual guides*, with an admittance control architecture, to constrain the user to follow a curve, surface or orientation. In the experimental section of this work, rotational and translational constraints were implemented separately. *Autonomous error compensation* was used in [Castillo-Cruces 2010] to overcome human difficulties in simultaneously controlling the position and orientation of a 6-DOF robot under the vision-based *preferred directions virtual guides* presented in [Li 2007]. The system uses computer vision as a sensor for providing a reference trajectory, and the virtual guide control algorithm then provides haptic feedback for implementing direct shared manipulation. The authors found that in a system with both position or orientation reference direction fixtures, only position or orientation would be effectively constrained at a time. They stated that this is due to the translational and rotational components of motion being decoupled from each other in such a way that the user, focusing on moving one, will not notice an error in the other. These works, have addressed orientation virtual guides but for a different type from those used in this thesis. Also, they address translation and orientation separately.

In the case of Dynamic Virtual Guides (DVG), the work of [Bowyer 2014b] extended dynamic frictional constraints to be able to constrain the position or the orientation of a tool. In contrast to our definition of Virtual Guides (Chapter 2: "Virtual Guides Definition via Virtual Mechanisms"), *DVG* are applied to environments which deform or move over time (e.g., soft tissue in the context of robot assisted surgery), and though the constraints are not based on virtual mechanisms but on elasto-plastic friction models. However, the definition of the position constraints in $\mathbb{R}^3$ and orientation constraints in $\mathbb{SO}(3)$ are done independently, so there is no synchronization of the translational and rotational movements, which is one of the

objectifs of our work: to propose a 6D constraint model that, in addition, can be easily obtain through kinesthetic teaching. The authors used a simulated version of *The Steady-hand Game* where users must simultaneously control both the position and orientation of the ring. Results showed that the assistance provided by full 6-*DOF* dynamic frictional constraints (the constraints in position and orientation are applied simultaneously but are not coupled) led to the best average user performance compared to cases without any assistance, or with position and orientation constraints separately. It was also found that constraining only position is beneficial even when orientation is a factor in the task. The likely cause of this is that the cognitive load involved in positioning the ring is reduced to such a degree that the user has the time to actively consider and control the orientation as required for the task.

In more recent work [Bowyer 2015], the authors extended their approach to n-dimensions and applied translational and rotational dynamic friction constraints using the same formalism. They repeated the task defined in their previous publication (*The Steady-hand Game*) and found out that the new approach can be of significant benefit to tasks requiring simultaneous accurate movements in multiple dimensions. The full dissipative controller, which employed energy redirection between all six dimensions, resulted in the best user performance when compared with the uncoupled dissipative controller.

In [Zhang 2012], the structure of geometric and dynamic constraints of reference tasks is analyzed using the screw theory. Virtual guides using screw theory unifies rotational and translational constraints into one set. The spatial compliance/stiffness matrices synthesis for admittance and impedance controlled devices was also studied. In more recent work [Zhang 2014], the authors studied the application of virtual guides in deforming environment, and proposed a novel framework of *DVG* for admittance-type devices. A framework for *DVG* in the Euclidean Group $\mathbb{SE}(3)$ was proposed to enhance the surgical operation accuracy of admittance-type medical robotics in the deforming environment. This approach unites rotation and translation in a compact form. The *DVG* can improve the dynamic properties of human-robot cooperation in low-frequency deforming environment, and maintain synergy of orientation and translation during the operation.

The difference between the approach of [Zhang 2014] and [Bowyer 2015] is that the former is applied on *dynamic constraints* based on the concept of *preferred directions virtual guides* presented in [Li 2007], and the latter n-dimensional dissipative control strategy is proposed to enforce dynamic constraints employing a new technique called "energy redirection", both in a robot assisted surgery context. These last works show the importance of coupling translation and orientation for Virtual Guides construction. However, their approaches are based on *Dynamic Virtual Guides* which are different from our non-dynamic definition based on virtual mechanisms. Nevertheless, our work is similar to [Zhang 2014], in the choice of orientation representation and distance definition in $\mathbb{SE}(3)$.

Next, we present our implementation of Orientation Constraints using *interpola-*

tion functions in $\mathbb{SO}(3)$.

3.3.2 Interpolation in SO(3)

In this section we address the Virtual Guides construction in $\mathbb{SO}(3)$, focusing on the orientation of the cobot end-effector. As explained before for the *Position Constraints* Construction, during the kinesthetic teaching, the user is able to show the cobot the desired movements by manipulating its end-effector. The orientation X_{rot} of the cobot end-effector is recorded by user demand (e.g. by pressing a button to record the current robot end-effector orientation) or continuously with a determined sampling time. In both cases, the orientation $X_{rot} \in \mathbb{SO}(3)$ and a recording parameter $t \in \mathbb{R}$ are stored as a list of points M_R:

$$M_R(t_i) = \{t_i, X_{rot,i}\}_{i=0:N-1} \tag{3.13}$$

where N is the number of recorded points.

When the movements recording is done by user demand, the recording parameter starts at $t_0 = 0$ and increases monotonically along with the recorded points. When the recording is done continuously, the recording parameter corresponds to the recording time. Thus, in both cases, the parameters t_0, t_1, $t_2 ... t_{N-1}$ satisfy the monotonic condition (3.2).

In order to create Orientation Constraints using the list M_R, we propose to use an interpolation function that matches exactly the recorded orientations in order to encode in a precise and smooth way the demonstrated movements. We previously stated the advantages of using interpolation functions for Position Constraints construction. The same advantages apply for Orientation Constraints. Next, we present some possible representations of orientations and the methods to interpolate them in order to define the geometric and kinematic models of the virtual mechanism from equations (2.1) and (2.3), defined in Chapter 3: "Virtual Guides Construction".

3.3.2.1 Orientation Representation through Unit Quaternions

The most common representations of orientations are rotation matrices and Euler angles. However, they both present several disadvantages for using them for orientation interpolations:

- The rotation matrix is not an optimal representation because it uses nine parameters to represent the three degrees of freedom of a rotation. Matrix would be more suitable when representing all the other transformations such as translation, scaling, projection and shearing.
- When using Euler angles, rotation must be expressed as the angles about three explicit axes, where the order matters. Describing a general rotation

in this way is not natural or intuitive for the user. Also, it is possible to encounter gimbal lock [Shoemake 1985]. It could be troublesome to uphold the mathematical constraints on this representation during calculations. Finally, to interpolate Euler angles, the coordinates of each basis axis must be interpolated independently. Thereby the interdependencies between the axes are ignored.

Among all others, the axis-angle representation is the most intuitive one since a rotation is defined through an angle θ about an axis represented by a unit vector $\hat{\mathbf{v}}$. The vector $\mathbf{v} = \theta\hat{\mathbf{v}} \in \mathbb{R}^3$ is known as the angle-axis representation of the rotation. However, when interpolating directly using $\mathbb{R}^3$ vectors (designed by $i = 1 : N$), we can find some inappropriate behavior of the resulting curve: the interpolation occurring in the 2D vector space leads to a coupled interpolation between θ_i and $||\hat{\mathbf{v}}_i||$. Firstly, this leads to the angle θ_i to compensate the crushing of the vector (as illustrated by the blue curve in Figure 3.5. This phenomenon results in an undesired rotation (see Figure 3.7). Besides, it induces an irregular step interpolation in the orientation space as shown in Figure 3.6. In Subsection 3.3.2.2 we will explain the *SLERP* interpolation method which allows to overcome the just mentioned issues of interpolating directly using $\mathbb{R}^3$ vectors.

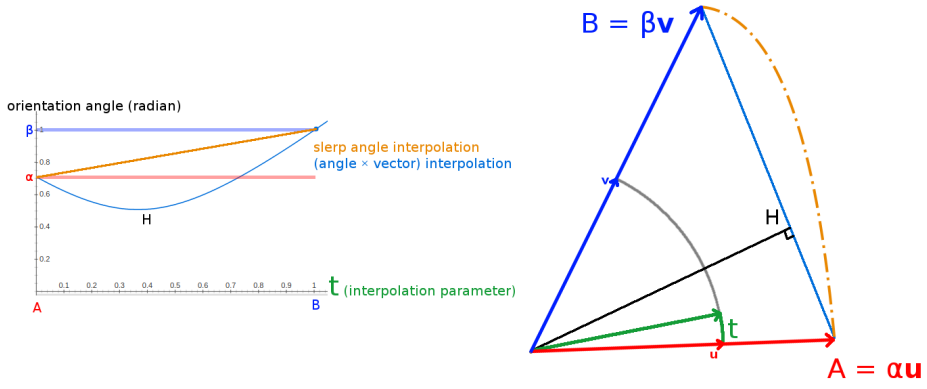


Figure 3.5: Comparison between interpolation into the angle-vector representation vs. the SLERP interpolation method. The figure on the right represent the plane between the two interpolated angle-vectors. By using linear interpolation directly into this space the distance between the unit circle and the interpolated angle-vector does not evolve linearly (as shown by the blue curve on the left). On the contrary, the angle-vectors obtained by SLERP interpolation represented by the dotted orange path allows to keep a linear interpolation of the rotation angle and of the direction vector.

We find that the best choice for our context of application is the quaternion representation. The quaternion representation is compact with a more natural geometrical interpretation and a parameterization of rotation that is not dependent

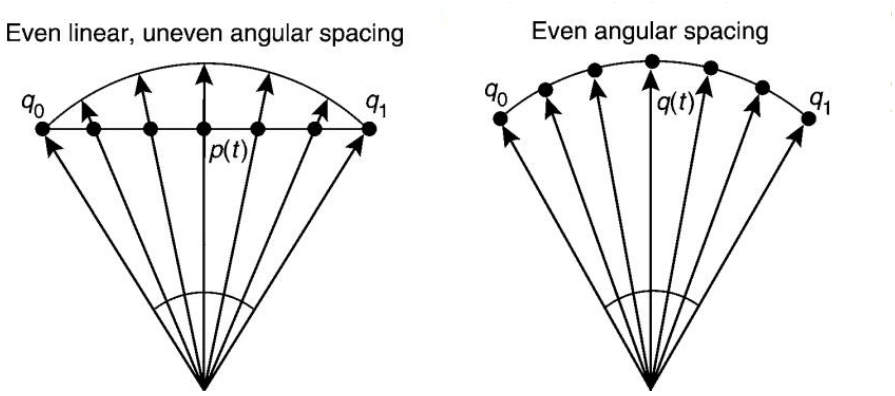
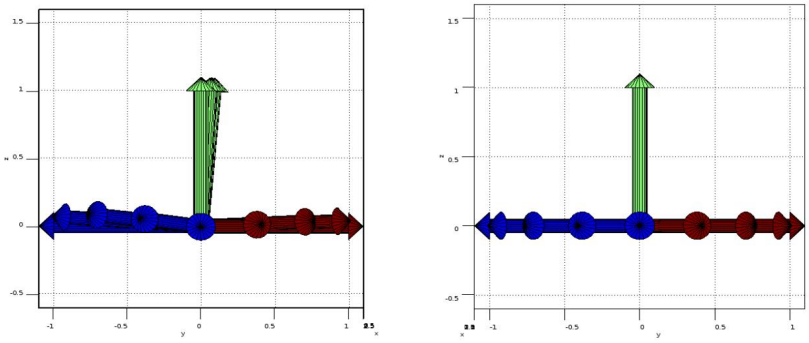


Figure 3.6: Comparison of the LERP (left) against the SLERP (right).



(a) Linear interpolation of axis-angle representation using vectors

(b) Slerp interpolation

Figure 3.7: Difference between linear vector interpolation and SLERP interpolation. We represent a rotation of pi/2 rad around the z axis. The problem we find is that the interpolation in the angle-vector representation produces intermediate rotations that do not remain on the (x,y) plane. This effect is more pronounced when the rotation angle increases.

on the coordinate system. Also, quaternion rotations have the unique property that the angle between any two vector representations (i.e. $arccos(q1 \cdot q2)$) is exactly half the minimum angle between the two rotations that they represent. Moreover, the set of unit quaternions can be considered as a unit 3-sphere, $\mathbb{SO}(3) \triangleq \mathbb{R}^4$ and therefore the geodesic distance between any two quaternions on $\mathbb{SO}(3)$ is also exactly half the minimum angle between the two rotations. Subsequently, the direction of a geodesic between two quaternions gives the most direct rotation between them. It is this property that makes quaternions pertinent for interpolation between rotations. A larger description of the advantages of using quaternions representation is presented

in [Funda 1990].

Next, we present some quaternion notations useful to the reading of this chapter.

Notations A quaternion q can be considered to be the association of a scalar $w \in \mathbb{R}$ and a vector $\mathbf{a} \in \mathbb{R}^3$:

$$q \triangleq \begin{bmatrix} w \\ \mathbf{a} \end{bmatrix}$$

Considering the three complex variables $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$, such as:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1 \quad \mathbf{ij} = \mathbf{k} \quad \mathbf{jk} = \mathbf{i} \quad \mathbf{ki} = \mathbf{j}$$

We can also define the quaternion q by:

$$q = w + i\mathbf{i} + j\mathbf{j} + k\mathbf{k}$$

In vectorial form:

$$q = \begin{bmatrix} w \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} w \\ i \\ j \\ k \end{bmatrix}$$

Relation with the angle-axis representation: The quaternion q corresponding to the rotation through an angle θ along the axis $\hat{\mathbf{v}}$ is defined by:

$$q = \begin{bmatrix} w \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} w \\ i \\ j \\ k \end{bmatrix} = \begin{bmatrix} \cos(\frac{\theta}{2}) \\ v_x \sin(\frac{\theta}{2}) \\ v_y \sin(\frac{\theta}{2}) \\ v_z \sin(\frac{\theta}{2}) \end{bmatrix}$$

The **conjugate** of a quaternion is denoted q^* and is defined by:

$$q^* = \begin{bmatrix} w \\ \mathbf{a} \end{bmatrix}^* = \begin{bmatrix} w \\ -\mathbf{a} \end{bmatrix}$$

The **norm** of a quaternion is denoted $\|q\|$ and is defined by:

$$\|q\| = \sqrt{w^2 + i^2 + j^2 + k^2}$$

The **multiplicative inverse** of a quaternion is denoted q^{-1} and is defined by:

$$q^{-1} = \frac{q^*}{\|q\|}$$

where the division of a quaternion by a real-valued scalar is just component-wise division.

If $\|q\| = 1$, the q is called a unit quaternion and in this case $q^{-1} = q^*$. The set of unit quaternions constitutes a unit sphere in four-dimensional space. *We will use these group of quaternions to represent rotations.*

The mapping between rotations and quaternions is unambiguous with the exception that every rotation can be represented by two quaternions q and $-q$.

The logarithm of a unit quaternion is defined by:

$$\log(q) = \begin{bmatrix} 0 \\ \hat{\mathbf{v}}\theta \end{bmatrix} \tag{3.14}$$

where $\hat{\mathbf{v}}$ is the unit vector representing the axis of rotation through an angle θ.

As presented in [Hanson 2005], the instantaneous angular velocity $\hat{\omega} \in \mathbb{R}^3$ of an unit quaternion q with first derivative $\dot{q} \in Q$, is defined by:

$$\begin{bmatrix} 0 \\ \hat{\omega} \end{bmatrix} = 2\dot{q} \cdot q^{-1} \tag{3.15}$$

Next, we first address the interpolation between two orientations using spherical linear interpolations of quaternions. Then, we explain the interpolation of a list of orientations using spherical cubic interpolation of quaternions.

3.3.2.2 Interpolation between two orientations

The interpolation between two given orientations can be done using the axis-angle or the quaternions representations. In this thesis, we will use unit quaternions to interpolate orientations and though we will not discuss interpolations using other representations of rotations.

SLERP *SLERP* of quaternions generates constant motion along the geodesic linking between two unit quaternions. A detailed definition of *SLERP* is given in [Shoemake 1985, Dam 1998]. Here, we present the analogous quaternion formula of the Euclidean expression for linear interpolation: $x(t) = x_0 + t(x_1 - x_0)$.

As the interpolation parameter t uniformly varies between 0 and 1, the values $Slerp(t)$ are required to uniformly vary along the circular arc from p to q, where p and q are unit quaternions. *SLERP* can be written in an exponential form [Dam 1998] as:

$$Slerp(t; p, q) = p(p^{-1}q)^t \tag{3.16}$$

An example of the application of (3.16) is shown in Figure 3.8.

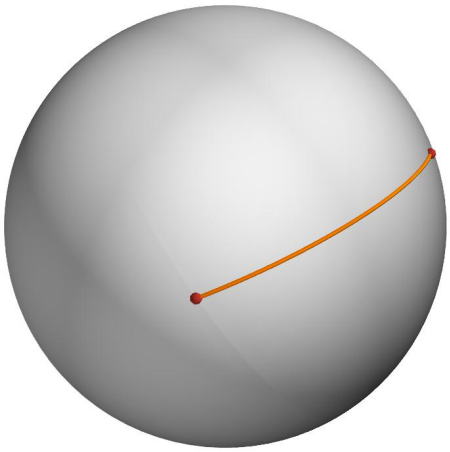


Figure 3.8: Example of the *SLERP* interpolation between two unit quaternions on the sphere.

The first derivative along t [Dam 1998] is given by:

$$\dot{Slerp}(t; p, q) = p(p^{-1}q)^t log(p^{-1}q) \tag{3.17}$$

Although p and $-p$ represent the same rotation, the values of $Slerp(t; q, p)$ and $Slerp(t; q, -p)$ are not the same. Indeed, geometrically the geodesic is a 4D-circle, so it can be followed from q in two directions, arriving firstly in p or firstly in $-p$. This is a consequence of $\mathbb{SO}(3)$, because in this set a rotation about some direction of 2π returns to the same orientation of origin. The shorter path can be established simply by comparing the distance between p and q with the distance between $-p$ and q. As explained in [Eberly 2002], it is customary to choose the sign σ on p so that $q \cdot (\sigma p) \geq 0$. In other words, the angle between q and σp is acute. This implementation choice avoids extra spinning caused by the interpolated rotations.

3.3.2.3 Interpolation over a Series of Orientations

In the set of unit quaternions, the *SLERP* interpolation curve of two quaternions is a *geodesic* (i.e., the shortest interpolation path between two quaternions on the $4D$ unit sphere). However, the simple juxtaposition of the successive geodesic interpolations joining a series of orientations presents some limitations (see Figure 3.9):

- the curve is not smooth at the control points,
- the angular velocity is not constant and
- the angular velocity is not continuous at the control points.

A reparametrization can easily ensure continuity across the entire interpolation, but fails to fix the lack of smoothness at the control points [Dam 1998]. The smoothness is understand here as the continuity of, at least, the first derivative (C^1 continuity).

SQUAD The Spherical Cubic Interpolation called *Squad* – spherical and quadrangle – was presented by [Shoemake 1985]. It takes inspiration from Bezier curves, but involves spherical linear interpolations instead of simple linear interpolations.

The evaluation of *SQUAD* uses an iteration of three *SLERP* similarly to the Casteljau algorithm [Farin 2014]:

1. Imagine four unit quaternions p, a, b, and q as the ordered vertices of a quadrilateral.
2. Interpolate the quaternion c along the "edge" from p to q using $Slerp(t; p, q)$.
3. Interpolate the quaternion d along the "edge" from a to b using $Slerp(t; a, b)$.
4. Now interpolate the edge interpolations c and d to get the final result e.

The end result (final interpolation) is denoted *SQUAD* and is given by:

$$Squad(t; p, a, b, q) = Slerp(2t(1-t)); Slerp(t; p, q), Slerp(t; a, b)) \tag{3.18}$$

We can use (3.16) to obtain the exponential form of *SQUAD*:

$$Squad(t; p, a, b, q) = Slerp(t; p, q)(Slerp(t; p, q)^{-1} Slerp(t; a, b))^{2t(1-t)} \tag{3.19}$$

An example of the application of equation (3.19) is shown in Figure 3.9.
The derivative of *SQUAD* in equation (3.19) is defined in [Eberly 2002] as:

$$\dot{Squad}(t; p, q, a, b) = \frac{d}{dt}[UW^{2t(1-t)}] \tag{3.20}$$

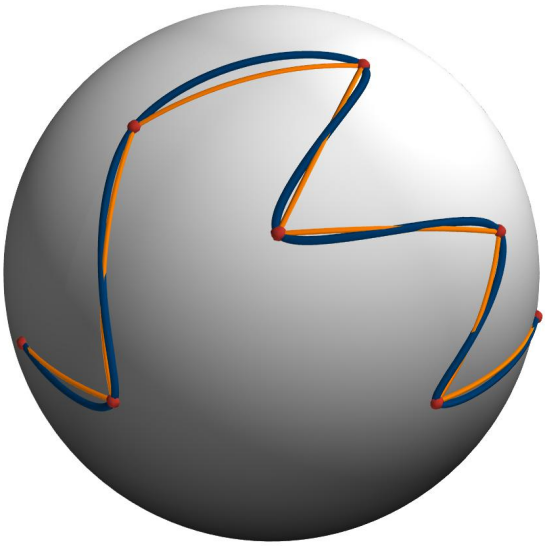


Figure 3.9: Example of the *SLERP* (orange) and the *SQUAD* (blue) interpolations between eight unit quaternions on the sphere. We can see that in contrast to the *SLERP* curve, the *SQUAD* curve is smooth at the control points (red).

Where

$$\begin{aligned} U(t) &= Slerp(t; p, q) \\ V(t) &= Slerp(t; a, b) \\ \dot{U}(t) &= U(t)log(p^{-1}q) \\ \dot{V}(t) &= V(t)log(a^{-1}b) \\ W(t) &= U(t)^{-1}V(t) \end{aligned}$$

where $S\dot{q}uad$ is not an unit quaternion.

It is also shown in [Eberly 2002] that the derivatives of *SQUAD* at the endpoints are:

$$S\dot{q}uad(0; p, a, b, q) = p[log(p^{-1}q) + 2log(p^{-1}a)] \tag{3.21}$$

$$S\dot{q}uad(1; p, a, b, q) = p[log(p^{-1}q) - 2log(q^{-1}b)] \tag{3.22}$$

3.3.3 Orientation Constraints Construction through Spherical Cubic Interpolation

Given a sequence of N unit quaternions $\{q_n\}_{n=0:N}$, we want to build an interpolation curve between those quaternions, subject to the following conditions:

- the spline pass through the control points

- the first derivatives are continuous at the control points

To this aim, the idea is to chose intermediate quaternions a_n and b_n to allow control of the derivatives at the endpoints of the spline segments. More precisely, let $S_n(t) = Squad(t; q_n, a_n, b_{n+1}, q_{n+1})$ be the spline segments. By definition of *SQUAD*, the last quaternion of a previous segment $n-1$ is equal to the first quaternion of the current segment n:

$$S_{n-1}(1) = q_n = S_n(0)$$

To obtain continuous derivatives at the endpoints we need to match the derivatives of two consecutive spline segments:

$$\dot{S}_{n-1}(1) = \dot{S}_n(0) \tag{3.23}$$

From equation (3.22) we can write:

$$\dot{S}_{n-1}(1) = q_n[log(q_{n-1}^{-1}q_n) - 2log(q_n^{-1}b_n)]$$

and

$$\dot{S}_n(0) = q_n[log(q_n^{-1}q_{n+1}) + 2log(q_n^{-1}a_n)]$$

The derivative continuity equation (3.23) provides one equation in the two unknowns a_n and b_n, so there is one degree of freedom. It is suggested in [Dam 1998] and [Eberly 2002] to use an average T_n of "tangents", so $\dot{S}_{n-1}(1) = q_n T_n = \dot{S}_n(0)$, where:

$$T_n = \frac{log(q_n^{-1}q_{n+1}) + log(q_{n-1}^{-1}q_n)}{2} \tag{3.24}$$

With this two equations (3.23), (3.24), a_n and b_n can be determined as follows:

$$a_n = b_n = q_n exp\left(-\frac{log(q_n^{-1}q_{n+1}) + log(q_n^{-1}q_{n-1})}{4}\right) \tag{3.25}$$

and $b_n = a_{n+1}$.

Thus, $S_n(t) = Squad(t; q_n, a_n, a_{n+1}, q_{n+1})$.

The expression of *SQUAD* is not defined in the first and last interval since q_{n-1} appears in the expression for a_0 and q_{n+1} appears in the expression for a_n. Therefore, it is necessary to define bound values for a_0 and a_n. This choice could

have an impact on the resulting interpolation curve continuity class and can be avoided during the implementation. Indeed, we could add two points, before the beginning and after the end of the useful path, to ensure continuity order on the desired portion.

Squad extrapolations In order to avoid discontinuities when implementing the Virtual Guides using the *SQUAD* interpolation curve, it is important to provide the extrapolations of the Virtual Guides.

For $n = -1$, we define the interpolation as:

$$q(t) = interpolation(t; q_{-1}, q_0) = Slerp(t; q_0, q_1) \tag{3.26}$$

with t the interpolation parameter between two quaternions.
and its derivative by:

$$\dot{q}(t) = \dot{Slerp}(t; q_0, q_1) \tag{3.27}$$

For $n = N + 1$, we define the interpolation as:

$$q(t) = interpolation(t; q_{N-1}, q_N) = Slerp(t; q_{N-2}, q_{N-1}) \tag{3.28}$$

where N represents the number of interpolation quaternions as though the quaternion $q(s = s_{max})$.
and its derivative by:

$$\dot{q}(t) = \dot{Slerp}(t; q_{N-2}, q_{N-1}) \tag{3.29}$$

3.3.3.1 Parameterization

As presented in Chapter 2: "Virtual Guides Definition via Virtual Mechanisms" and in Section 3.2: "Virtual Guides as Position Constraints", singularities can appear on the virtual mechanisms when the Jacobian J_s is not normalized. These singularities can disturb the interaction between the user and the Virtual Guides controller. Since the list of orientations recorded by the cobot are obtained by users demonstrations, we can not guarantee the normality of J_s.

In order to guarantee a normalized Jacobian, it is desirable to evaluate the *SQUAD* at orientations based on the arc-length of the curve instead of based on the recording sampling time. For that matter, we propose to separate spacial and temporal aspects of the trajectory.

Let t be the time, s_θ be the arc-length quaternion curvilinear parameter and P_R the list containing the *SQUAD* orientations waypoints $q_{rot} \in \mathbb{SO}(3)$.

$$P_{R,i} = \{q_{rot,i}\}$$

with $i = 0 : N-1$, where N is the number of orientation waypoints. When the waypoints are recorded manually, $t = i$.

The *SQUAD* curve parametrized with time t is defined by f_θ:

$$f_\theta : \left| \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{SO}(3) \\ t & \longmapsto & P_R \end{array} \right.$$

The transformation function from the the arc-length curvilinear to time parameterization is defined by g_θ:

$$g_\theta : \left| \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{R} \\ s_\theta & \longmapsto & t \end{array} \right.$$

Where g_θ corresponds to a monotonic cubic interpolation function [Fritsch 1980]. This kind of interpolation is optimal for the transformation function since the resulting curve does not present oscillations in the presence of outliers.

In the context of rotations in $\mathbb{SO}(3)$, the natural metric is equal to the angle between two rotations. Specifically, given two rotation quaternions r and p, the product rp^{-1} is also a rotation by an angle $\theta \in [0, \pi]$ about some axis. We chose to use the metric of quaternion $d(r, p)$ defined by:

$$d(r, p) = \|\log(r^{-1}p)\| = \theta \tag{3.30}$$

where the log function is defined in (3.14).

We approximate the computation of s_θ by:

$$s_{\theta,i+1} = s_{\theta,i} + d(q_i, q_{i+1})$$

where $s_{\theta,0} = 0$ and $i = 0 : N-1$. The computation of $d(q_i, q_{i+1})$ is done using (3.30).

The result of this parameterization is an equal arc length quaternion curve subdivision, thus a normalized Jacobian J_s.

The *SQUAD* spline can now be defined as a composition of the initial curve parameterized with time f_θ and the space transformation function g_θ as:

$$f_\theta(t) = f_\theta(g_\theta(s_\theta)) = f_\theta \circ g_\theta(s_\theta) \tag{3.31}$$

The Virtual Guide is now described by the following data list of length N:

$$M_R(t_i, s_i) = \{t_i, s_i, q_{rot,i}\}_{i=0:N-1}$$

Geometric and Kinematic models The above definitions of the *interpolation function* for quaternions can be applied to the list of M_R points (3.13) where t values

increase monotonically (3.2).

$$M_R(t_i) = \{t_i, x_{rot,i}\}_{i=0:N-1}$$

where N represents the number of points.

We can now define the geometric model of the virtual mechanism L_s (2.1) as:

$$L_s : \left| \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{SO}(3) \\ s_\theta & \longmapsto & Squad(s_\theta; \phi) \end{array} \right.$$

where ϕ represents the other function parameters needed to compute $Squad$.

and:

$$X_{vm,rot} = Squad\left(g_\theta(s_\theta), \phi\right) \tag{3.32}$$

We can notice that the parameter s_{vm} of the virtual mechanism corresponds to the spline parameter s_θ. We can then write:

$$X_{vm,rot} = Squad\left(g_\theta(s_{vm}), \phi\right) \tag{3.33}$$

The kinematic model of the virtual mechanism J_s (2.3) can be defined as the angular velocity which can be obtained using the *SQUAD* derivative. In this thesis, we decided to use the discrete derivative of *SQUAD*, since the implementation of the derivative defined in equation (3.20) and detailed in [Dam 1998], resulted in undesired behaviors of the angular velocity. For example, we were unable to obtain constant rotation velocity when applying *SQUAD* between a list of uniformly spaced orientations (i.e., same angle between orientations). We believe, this could come from the implementation of the chain rule and the product rule for quaternions.

We defined the discrete derivative by:

$$\dot{Squad}(t; \phi) = \frac{Squad(t + \varepsilon; \phi) - Squad(t - \varepsilon; \phi)}{2\varepsilon} \times \left(\frac{1}{s_{max} - s_{min}} \right) \tag{3.34}$$

In our implementation, a value of $\varepsilon = 0.001$ gave good results.

The kinematic model of the virtual mechanism J_s can be defined using the instantaneous angular velocity $\hat{\omega} \in \mathbb{R}^3$ from (3.15), by:

$$J_s = \hat{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{3.35}$$

$$\dot{X_{vm,rot}} = J_s \dot{s_{vm}} \tag{3.36}$$

3.4 6D Virtual Guides

We presented in Section 3.2 and Section 3.3 how to define and construct Position and Orientation constraints. However, robotic applications usually use the pose of the end-effector (or the calibrated tool). We presented in Subsection 3.3.1 the related works to the creation of 6D Virtual Guides and we will present in this Section our new approach based on virtual mechanisms and interpolation functions.

6D Virtual Guides through XSplines To enforce 6D Virtual Guides we use the virtual mechanisms controller presented in Chapter 2: "Virtual Guides Definition via Virtual Mechanisms". However, we must first define the Virtual Guides and implement their geometric and kinematic models. To this aim, we developed the concept of *XSplines*.

XSplines are curves in $\mathbb{SE}(3)$, defined by both position interpolations in $\mathbb{R}^3$ and orientation interpolations in $\mathbb{SO}(3)$ of poses of a robot obtained through kinesthetic teaching. In other words, *XSplines* are a composition of *MDSplines* and *Squads*. Figure 3.10 shows an illustration of this concept. The advantage of *XSplines* is that they are parameterized in a way that allows the synchronization of the translation and orientation movements.

A pose $X \in \mathbb{SE}(3)$ is defined as:

$$X = \left\{ \begin{array}{l} X_{trans} \in \mathbb{R}^3 \\ X_{rot} \in \mathbb{SO}(3) \end{array} \right\} = \left\{ \begin{array}{c} x \\ y \\ z \\ w \\ i \\ j \\ k \end{array} \right\}$$

Parameterization The space parameter s_x is defined to variate depending on both the translational and rotational components of a pose. This parameterization allows s_x to evolve along the curve even if the movement is done only on one of the components. This feature could be very useful when the movement is described by only a rotation along an axis without any translation. In this case, since translation and rotation are coupled via an XSpline, the curve parameter will continue to evolve.

This parameterization uses the previously defined parameterizations in $\mathbb{R}^3$ and $\mathbb{SO}(3)$. To this aim, we define a scaling factor L between the two parameters, which value depends on the geometry of the tool used on the robot.

For two displacements e and u, we define the space parameter s_x by using the distance $d_x(e, u)$ between them, with:

$$d_x(e, u) = \sqrt{s^2 + L s_\theta^2} \tag{3.37}$$

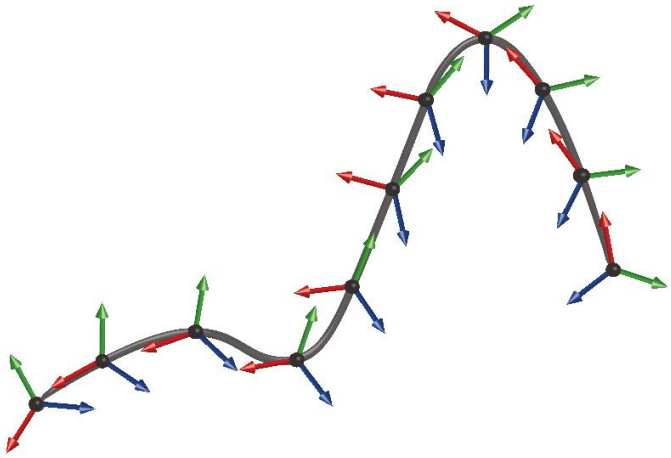


Figure 3.10: Illustration of a *XSpline* where both translations and orientations movements are considered in a single 6D curve.

where:

- s: represents the cartesian translation parameter defined in Subsection 3.2.2.2: "Parameterization",
- s_θ: represents the $\mathbb{SO}(3)$ rotation parameter defined in Subsection 3.3.3.1: "Parameterization" and
- L represents the scaling factor between both parameters.

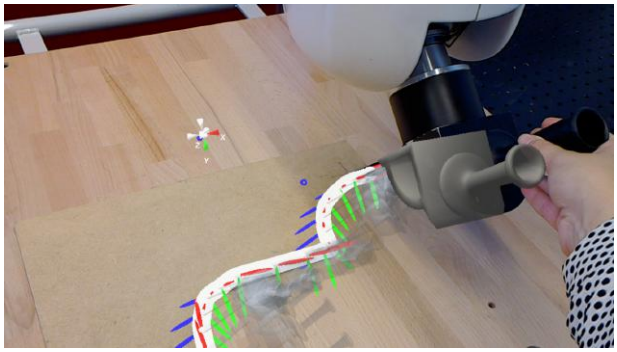
We approximate the computation of s_x by:

$$s_{x,i+1} = s_{x,i} + d_x(x_i, x_{i+1})$$

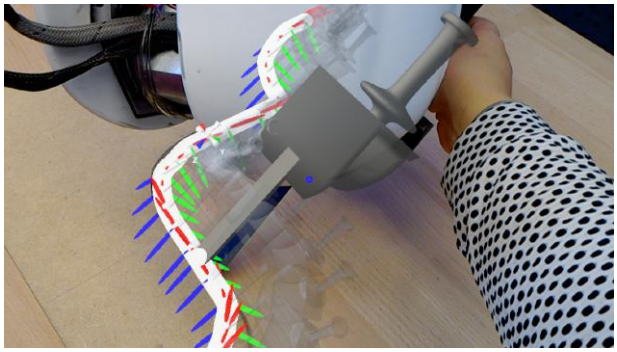
where $s_0 = 0$ and $i = 0 : N-1$. The computation of $d_x(x_i, x_{i+1})$ is done using (3.37).

In our implementation, presented in Chapter 5: "Experimental Evaluation" - Experiment 2, $L = 0.1$ (corresponding to a lever arm of $10cm$) gave good results.

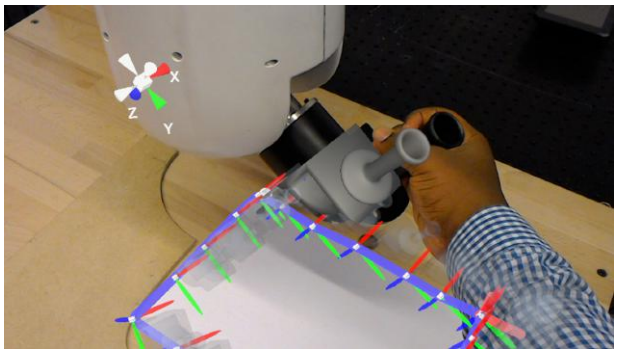
An example of 6D Virtual Guides using XSplines – parameterized as explained above – is shown in Figure 3.11.



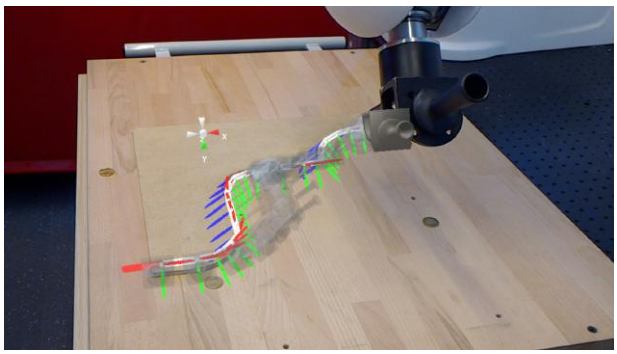
(a)



(b)



(c)



(d)

Figure 3.11: Visualization of 6D Virtual Guides using an Augmented Reality (AR) interface. The XSplines are represented by the pose of the tool attached to the cobot. The translational component of the guide is represented by cartesian points (white points on figure *c)*) and the rotational component of the guide is represented by trihedrons along the path (red-green-blue). We can also see the different orientations of the model of the tool through the path (transparent gray).

Geometric and Kinematic models The above definition of *XSplines* can be applied to a list of M_X poses (3.13) where t values increase monotonically (3.2).

$$M_X(t_i) = \{t_i, x_i\}_{i=0:N-1}$$

where N represents the number of poses.

We can define the geometric model of the virtual mechanism L_s (2.1) as:

$$L_s : \left| \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{SE}(3) \\ s_x & \longmapsto & XSpline(s_x; \lambda) \end{array} \right.$$

where λ represents the other function parameters needed to compute $XSpline$: $\lambda = X_{trans,i}, X_{trans,i+1}, X_{rot,i}, a_i, a_{i+1}, X_{rot,i+1}$.

As explained before, the parameter s_{vm} of the virtual mechanism corresponds to the curve parameter s_x. Then:

$$X_{vm} = XSpline\,(s_{vm}, \lambda) \tag{3.38}$$

with:

$$XSpline(s_{vm}, \lambda) \triangleq \left\{ \begin{array}{c} MDSpline(g(s_{vm})) \in \mathbb{R}^3 \\ Squad(g_\theta(s_{vm}), \lambda) \in \mathbb{SO}(3) \end{array} \right\}$$

The kinematic model of the virtual mechanism J_s (2.3) can be defined using:

$$J_s = \begin{bmatrix} MD\dot{S}pline(g(s_{vm})) \\ \hat{\omega} \end{bmatrix} = \begin{bmatrix} \dot{spline}_x(g(s_{vm})) \\ \dot{spline}_y(g(s_{vm})) \\ \dot{spline}_z(g(s_{vm})) \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{3.39}$$

and :

$$\dot{X}_{vm} = J_s \dot{s}_{vm} \tag{3.40}$$

3.5 Conclusion

In this Chapter we explained how the geometric and kinematic models of Virtual Guides – presented in Chapter 2: "Virtual Guides Definition via Virtual Mechanisms" – can be programmed through kinesthetic teaching and modeled through interpolation functions. To construct Position Constraints we implement multi-dimensional Akima

Spline interpolations. To construct Orientation Constraints in $\mathbb{SO}(3)$ we use Spherical Cubic Quaternion interpolations (*SQUAD*). In both cases we proposed to separate the time and space components of the curves to parameterize them in a way that guarantees the Jacobian normality. We also proposed a 6D Virtual Guides definition through $XSplines \in \mathbb{SE}(3)$, based on both position interpolations – $MDSpline \in \mathbb{R}^3$ – and orientation interpolations – $Squad \in \mathbb{SO}(3)$ – of poses obtained through kinesthetic teaching. The advantage of *XSplines* is that they are parameterized in a way that allows the synchronization of the translation and orientation movements. These guides form the basis of our iterative programming framework presented next in Chapter 4: "Iterative Virtual Guides Programming".

Chapter 4

Iterative Virtual Guides Programming

Contents

In this chapter we present a new framework to program Virtual Guides in an intuitive way by using an iterative kinesthetic teaching approach where the human operator is assisted. The Virtual Guides definition and implementation were presented in Chapter 2: "Virtual Guides Definition via Virtual Mechanisms". Then, Virtual Guides construction using information directly recorded by the robot was explained in Chapter 3: "Virtual Guides Construction". With these elements, we will show how to iteratively refine or modify Virtual Guides both in position and orientation. In Section 4.1 we define the roles assigned to the cobot and the human operator during the task programming and execution. Then, we present in Section 4.2 the definition of our novel Virtual Guides programming framework. Section 4.3 explains the possible interaction modes with the cobot and Sections 4.4 and 4.5 present the iterative Virtual Guides modification.

4.1 Human-Robot Interaction Roles

When seeking for an intuitive, fast and cost-effective method for programming Virtual Guides on a comanipulation robot, the roles of interaction between the cobot and the human operator should be defined. In comanipulation tasks, the division of responsibilities can be done according to the skills of the actors involved in the collaboration. [Dumora 2014] discussed several methods of role assignment between man and machines in the context of long objects comanipulation. We next present a brief summary of these role allocation methods.

- Allocation by imitation: This strategy was proposed by [Reed 2008] where it is used in the context of human-robot interaction. It consists in designing a robot capable of imitating a man who interacts with a human partner. This method is not suitable to our context since it does not take into account the capabilities of the robot, which is the core of the assistance for comanipulation tasks in industrial environments. The interest of associating a robot and a human operator is to be able to carry out tasks that each one alone is not able to carry out. This strategy seems more adapted to analyse human behavior and interaction.

- Economic allocation: This strategy consists in allocating to the machine only the most profitable functions among the automatable ones, compared to the cost of a human operator. Here again, this method does not rely on the limits of the human operator. The goal of our framework is to provide assistance to human workers while programming a cobot, since the working conditions of the operator are a major issue.

- Leftover allocation: This strategy consists in allocating to the machine all the automatable functions. The human operator is assigned the remaining functions, for which automation techniques are not available. This strategy assigns to the robot as many functions as possible, relieving the operator of cognitive and physical overload so he/she can focus on a minimum of subtasks. However, this can be frustrating in cases where the robot performance is less efficient than what the human could achieve, creating a negative psychological feeling for the operator. Furthermore, this strategy does not take into account the capabilities of each actor of the collaboration. The operator is left in the background even though his/her involvement in the system is particularly important because the assistance is aimed to be used by him/her.

- Comparison allocation: This strategy is a pioneer on role allocation. It is based on the comparison of the relative capacities of humans and machines in order to fulfill each function. A function is assigned to the actor with the best ability to perform it. This strategy is also called MABA MABA – "Men Are Better At" "Machines Are Better At" – [Fitts 1951], and it allows to exploit the complementary capabilities of each partner.

In the context of human-robot comanipulation assistance, the MABA MABA strategy seems to be the more pertinent. An example of this strategy criteria is shown in Figure 4.1. Next, we explain how we apply this strategy to our Virtual Guides programming framework.

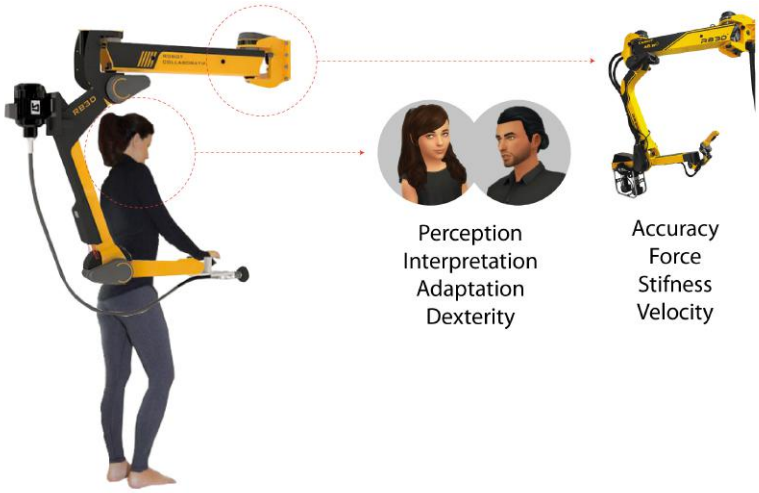


Figure 4.1: Illustration of some of the human-robot complementary skills.

The human operator should master the action plan. In the context of human-robot comanipulation in small industries, the human operator generally knows the task and has the gesture expertise to execute it. Besides this knowledge, he has also the capacity for interpreting visual data in a much more robust and cost-effective manner than commercial cobots. Also, his ability to improvise when facing an unexpected situation (i.e., changes in the environment or the task) allows him to instantly change his action plan. Moreover, dexterity and cognitive abilities of today cobots are very poor compared to those of humans. For these reasons, we entrusted the decision of the action plan to the operator, relying on the cognitive ability of the operator to add *flexibility* to the system.

The human operator should master the motion generation. As presented in Chapter 2: "Virtual Guides Definition via Virtual Mechanisms", a major constraint in human-robot interaction is to ensure the stability. It was shown in [Mussa-Ivaldi F. A. 1985] that the human operator is very complex and difficult to characterize, since the impedance at the point of interaction, i.e. his hand, is very variable. However, it was demonstrated in [Hogan 1989] that humans have the ability of providing energy when necessary while emulating a passive impedance interaction system. For this reason, the role of generating motions is also attributed to the operator.

This role can demand an important physical effort to the operator. This is why we propose an iterative framework for programming the cobot. The operator could

not only be relieved of physical effort but also the cognitive load could decrease thanks to the iterative nature of the process.

Role of the human operator. In summary, the human operator should be the master of both the programming and the execution of the task, in order to benefit from his/her gesture experience and knowledge of the task and the environment. Also, his/her cognitive abilities add flexibility to the system. Finally, the requested physical effort to generate movements are reduced by using the Virtual Guides Assistance presented in Chapter 2: "Virtual Guides Definition via Virtual Mechanisms".

The cobot should be able to refuse the action plan and act according to the human operator's intention. The decision of the action plan was attributed to the operator who knows the task and the environment and has higher cognitive capabilities than the cobot. However, the cobot must be able to force the operator's movements to follow some guide. For example, it is easier for the cobot to prevent a collision with itself thanks to its proprioceptive sensors. The assignation of this role to the cobot allows also the operator to only concentrate on the task and not on the cobot. We also decided that the cobot should act according to the human operator's intention. To do this, they are two approaches:

- Predictive approach: the idea is to *anticipate* the intention of the operator by prediction. In this case, the robot uses information that allows it to act before the operator realizes or tries to realize his/her action.
- Reactive approach: the idea is to *recognize* the intention of the operator. In this case, the robot uses information that comes from the action of the operator who is trying to realize his/her intention.

Since we have defined that the human operator should master the action plan and the generation of the movements, the cobot should then follow instructions rather than applying any automatic algorithm to try to identify the intention of the operator. Moreover, one drawback of the predictive approach is that it could became counterintuitive for the user if it is not well implemented on the cobot. We use then a pure reactive approach, where the intention of the operator is taken into account by explicit actions via physical buttons. Using the force information coming from the human operator interaction with the cobot could be an interesting solution to apply the reactive approach. However, this is left out for future work. Another interesting application of this approach was presented in [Dumora 2014] where naive Bayesian classification is used to recognize the intention of the operator to use a library of assistances in the context of big objects comanipulation.

The cobot should force the movements according to the programmed task. The physical abilities of robots are much higher than those of the operator and also invariable over time. This is why we attribute to the cobot all the physical

tasks. The cobot must then physically assist the operator to facilitate both the programming and the execution of the task. To this aim, we propose to assign to the cobot the task of blocking the degrees of freedom of motion not used by the operator's action plan (i.e., programmed movements) and let free those used, as presented in Chapter 2: "Virtual Guides Definition via Virtual Mechanisms". Thereby, the operator can move the cobot end-effector through the allowed paths, without having to worry about unwanted movements.

Role of the cobot: In summary, the main roles of the cobot are to constrain the movements that do not correspond to the desired programmed task (e.g. a path to follow). The cobot does not automatically predict or detect the intention of the operator, it follows explicit instructions.

Conclusion: It is important to note that our decisions apply only to the programming and task execution in an industrial comanipulation context presented in Chapter 1: "Introduction". The field of human-robot interaction is far larger and choices would be different depending on the application and the type of human-robot interaction. We have reached the solution of dividing the human operator and the cobot work according to their skills. The main roles of the operator are to decide the action plan and generate the movements. In this context, the cobot force is used to avoid undesired motions and to follow the instructions of the user.

Based on the previously defined roles of interaction of the human operator and the cobot, we present next a new framework for programming Virtual Guides.

4.2 Iterative, Intuitive and Assisted Programming of Virtual Guides

In Chapter 1: "Introduction" - Section 1.1.3, we presented the core motivations of our programming framework based on the problematics of the context of this research and on the current limitations of the literature. In this section, we summarize the main features of our framework.

The main objective is to propose different tools to enhace the **flexibility of Virtual Guides programming**. To increase flexibility we think the framework must propose easy and fast **modification tools** and also be **intuitive** so no robotics-experts can use it. This is why we choose to use *PbD* and particularly, **kinesthetic teaching**. However, in the context of **collaborative robotics**, cobots could present some remaining inertia and friction that can make the direct manipulation harder. Since the core idea is to assist human operators during a programming task, we propose to make the programming **iterative**. In this way, users do a first demonstration of the Virtual Guide by directly manipulating the end-effector of the cobot and then the Virtual Guides Assistance is activated. While the assistance is active, the user can **refine or modify** the Virtual Guide in both **position and/or orientation**.

The process is done iteratively until the **user is satisfied** with the result. An important advantage of this iterative approach, beyond the gain of flexibility of the process, is that **the physical effort and cognitive overload of the user can be reduced**. The first demonstration can also be replaced by pre-programmed paths or **simple tools** to rapidly **create basic guides** such as: straight lines, arcs, planar surfaces, among others. In addition, the iterative programming idea can be applied to more complex tasks where velocity, force and stiffness (of the virtual mechanism spring) must be encoded together with the guiding path. Indeed, it is easier for the user to concentrate in only one task at the time (e.g. teaching first position and then orientation, velocity, force or stiffness). If well implemented, this framework can also allow the users to program Virtual Guides faster and increase accuracy. Finally, these tools are thought for the human operators that are the masters of both the task programming and execution. We aim at reducing the penibility of the programming process while increasing the user's satisfaction and comfort.

Implementation The iterative programming framework can be implemented in any kind of collaborative robot that allows safe physical interaction with human operators. It can also be used together with other programming approaches but this highly depend on the final application.

The tools presented in this Chapter, were gradually developed by using three different robots: 2-*DOF*, 3-*DOF* and 6-*DOF*. Experimental validation with users on the 3-*DOF* and 6-*DOF* cobots will be presented in Chapter 5: "Experimental Evaluation". All the components of the framework, from the Virtual Guides Controller ((Chapter 2: "Virtual Guides Definition via Virtual Mechanisms")), passing through the Virtual Guides Construction (Chapter 3: "Virtual Guides Construction"), to the Virtual Guides Iterative Programming tools (presented in this Chapter), where programmed within the proprietary Robotics Framework from the Interactive Robotics Laboratory at *CEA*.

Interfaces To interact with the cobot in order to create, activate or modify Virtual Guides, the human operator can use tangible, graphical and augmented reality interfaces. The core interface of our framework is the robot itself (physical buttons and haptic feedback) since the programming is done manually by demonstration. However, together with the developement of the Virtual Guides tools, a few prototype interfaces were also conceived. In the context of this thesis, the implementation of the framework in different robots and the development of interaction interfaces can be considered as a more technical contribution and so we will not exhaustively discuss this subject on this chapter. However, there are a few important concepts to keep in mind for the interfaces conception:

- They must be as most intuitive as possible since it is the core idea of the framework.

- The ergonomy of the interaction should be considered (e.g. too many buttons or information could annoy the user)
- The user must be able to save/load the Virtual Guides in order to use or modify them later.
- Virtual Guides within the iterative framework can be considered as the addition of a *base guide* (i.e., obtained through the first demonstration or a pre-programmed guide) and a history of *guide modifications*. In this way, the user can undo one or several modifications applied to the Virtual Guide, or reuse a *guide modification* to apply it to another *base guide* later.
- The interface should be programmed using a modular architecture, so new tools can be added easily. It should also be possible to use with any robot (at least those who use the same Robotics Framework).

Finally, visual feedback is an important feature for the use of Virtual Guides. We addressed this subject by developing an Augmented Reality interface to visualize 6D Virtual Guides. We will briefly discuss this in Section 6.2: "Perspectives" of last chapter.

Next, we present three different interaction modes that allow to program and use the Virtual Guides. Further in this Chapter, Virtual Guides refinement and modification will be addressed.

4.3 Interaction Modes

Within the programming framework presented above, the user will need, at some point, to "escape" from a Virtual Guide due to variations in the environment or in the task. If a particularly stiff guidance constraint is in use, this could be difficult. For example, if a physical obstacle blocks the path, the robot can become immobilized. In [Marayong 2004], a process for selecting the optimal constraint compliance value with respect to this problem is reported. The method described searches for a compromise between the level of permissible accidental violation of the constraint and the need for freedom of intentional violation of it. In [Nolin 2003] three methods for deciding when to apply a force scaling in an "explicit", "implicit" or "autonomous" way, were studied. It was stated that the autonomous mode could be surprising for the user. Similarly, [Passenberg 2011] used "interaction forces" between the user and the robot to achieve an autonomous response. By monitoring how much agreement there is between the user and robot they classify intentional violation cases and react accordingly. In our case, we have previously defined the roles of interaction between a user and a cobot, and stated that the role of the cobot was to physically and passively assist the user while following instructions and reacting to users' explicit commands. This is why we propose to define interaction modes based on explicit cues.

Inspired from [Nolin 2003] and [Raiola 2017b], we define three different modes of interaction in our framework.

1. Hard Virtual Guides: The user is constrained to the Virtual Guides without being able to escape them. The spring K from the virtual mechanism, will exert high forces to pull the end-effector back to the Virtual Guide. This is useful in scenarios where there are no changes in the environment or the task and the user benefits from the Virtual Guides assistance to execute a task.

2. Soft Virtual Guides. The user is able to "escape" the Virtual Guides. The force exerted by the virtual mechanism decreases if the distance to the Virtual Guide becomes large. This mode is particularly useful to let the user refine the guides within the iterative programming framework. It is also useful to modify an existing Virtual Guide whenever there are changes in the task or the environment.

3. Null Virtual Guides. This mode is equivalent to operating the robot in gravity compensation mode with no Virtual Guides Assistance. It is used to realize the first demonstration of a new Virtual Guide.

We present in Table (4.1) the differences between the features of the three interaction modes.

	Type of guide		
	Null	Soft	Hard
Create new Virtual Guides	Yes	Yes	No
Refine/modify virtual Guides	No	Yes	No
Attracted to Virtual Guide when close to it	No	Yes	Yes
Attracted to Virtual Guide when far from it	No	No	Yes

Table 4.1: Main functionalities of the three interaction modes.

Soft Virtual Guides implementation: As explained above, the user might need to momentarily escape the Virtual Guide, in order to refine or modify it while the assistance is still active. To this matter, we use the concept *force scaling* presented in [Nolin 2003, Raiola 2015]. It allows the user to get out of a constraint and locally modify the Virtual Guide. In [Nolin 2003], three force scaling functions were proposed: toggle, fade and hold (see Figure 4.2). In this work, no consistent patterns for user preference between scaling modes were found. We decided to use the fade mode since we consider that smooth transitions could be more pleasant and intuitive for the user.

When the user tries to "escape" the guide, the force of the virtual mechanism fades proportionally with the distance between the pose of the virtual guide x_{vm} and the current pose of the cobot end-effector x. The user will feel an attractive force F when escaping or approaching the guide, up to a defined distance d_{max}.

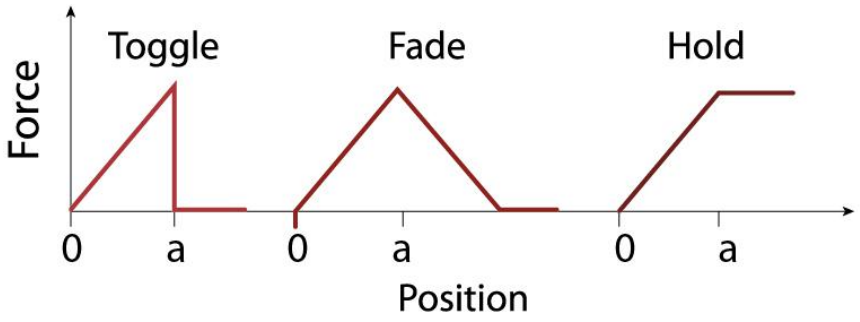


Figure 4.2: Force vs. position profiles for three force scaling methods presented in [Nolin 2003]. Position is the perpendicular distance between the user and the line representing the virtual fixture, where '0' is on the line and 'a' is the activation point.

$$F = \alpha(\varepsilon)F_c$$

Where F_c is the force applied by the virtual mechanism on the cobot, as explained in Chapter 2: "Virtual Guides Definition via Virtual Mechanisms".

In order to produce a quick fading effect at the proximity of the guide, the force fading is not linear as in [Nolin 2003] but follows a smooth curve. For computational efficiency purposes, $\alpha(\varepsilon)$ is defined as a 4th degree polynomial $f(x)$ as follows:

$$f(x) = ax^4 + bx^3 + cx^2 + dx + e \tag{4.1}$$

$$\dot{f}(x) = 4ax^3 + 3bx^2 + 2cx + d \tag{4.2}$$

We take into account the following constraints to solve the equation system of (4.1) and (4.2):

- $f(0) = 1$
- $\dot{f}(0) = 0$
- $f(-1) = f(1) = 0$
- $\dot{f}(-1) = \dot{f}(1) = 0$

We obtain the following polynomial:

$$f(x) = x^4 - 2x^2 + 1 \tag{4.3}$$

And we use it to define $\alpha(\varepsilon)$ as shown in 4.3, where:

$$\varepsilon = \frac{\| x - x_{vm} \|}{d_{max}}$$

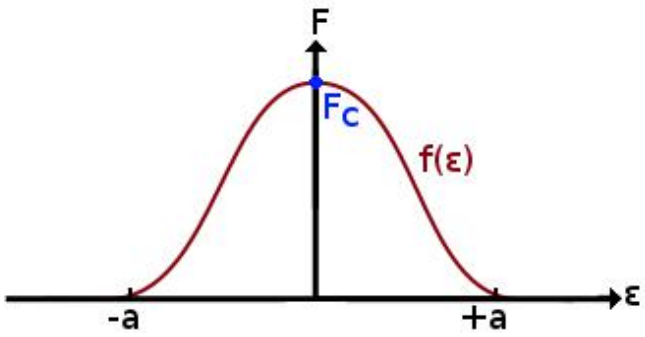


Figure 4.3: Scaling force function using a fourth degree polynomial: $f(\varepsilon) = \varepsilon^4 - 2\varepsilon^2 + 1$. Here, 'a' is the activation point.

The parameter d_{max} can be manually tuned to modify the basin of attraction of the virtual guide. In the case of 6D Virtual Guides, ε should be defined according to the metric (3.37) defined in Chapter 3: "Virtual Guides Construction".

Finally, a more distinctive feature of this *soft* interaction approach is the possibility to go back to the Virtual Guide intuitively without needing to change the control mode.

4.4 Local Refinement and Modification of Virtual Guides

When asking the operator to perform several times the ideal path he has in mind, one realizes that there may be many variations. These variations are due to the forces of interaction with the robot that the operator must compensate, the poor repeatability inherent to human gestures, the variability in the task and, often, simply concentration errors on a trajectory portion. For these reasons and the ones enlightened in Section 4.2, we propose that the operator incrementally modifies the virtual guides while being assisted by the robot. The refinement or modification are done directly on the workspace by manually placing the cobot end-effector to the new desired position, orientation or by doing a partial demonstration of a portion of the Virtual Guide to modify.

4.4.1 Virtual Guides Refinement of Translation

We understand by refinement of a Virtual Guide, the local modification of a point of the guide. Next, we will discuss the refinement of the translational component of a 6D Virtual Guide (or position constraints).

Usually, the shape control of spline curves like the B-Splines is achieved by the modification of its knots values [Juhász 2001]. However, since the translational component of Virtual Guides are constructed via interpolation curves, we propose to modify them by applying a "deformation" method to the the control points of the Akima splines presented in Chapter 3: "Virtual Guides Construction". Next, we present a brief review of deformation methods inspired from the field of Computer Design and Animation [Gain 2008].

4.4.1.1 Spatial Refinement Theory

Spatial deformations modify – or "deform" – an object by warping its ambient space. The deformation can be formulated as a mapping from the world space through a local parameter space, defined by the deformation tools of a particular technique. They are different types of deformations:

- Point-based ($0D$)
- Curve-based ($1D$)
- Surface-based ($2D$)
- Volume-based ($3D$)

In general, all deformations are controlled by positioning and repositioning control points, and possibly by means of simple ways for specifying scaling, tangent orientation or deformation regions. The criteria to analyse a deformation method can depend on different factors:

- Versatility: position, size and shape of the deformation boundary offered by each technique.
- Ease of use.
- Efficiency: interactive feedback, computation cost.
- Correctness: internal validity and external realism of the shape.

In our case, we want to locally modify a point of a Virtual Guide that is implemented as a curve. For this reason, we relay on Point-based ($0D$) deformation methods.

Point-based deformation: The idea is to enable a user to drag object points and have the surrounding surface to adapt smoothly. For instance, pushing or pulling a single object point will create either a dimple or a mound in the object. The deformation of an object is defined by the displacement of points called *constraints*. There are several approaches which we do not discuss exhaustively.

Two of the most commun approaches are the Free-Form and Direct Manipulated Free-Form Deformation (DMFFD), presented by [Sederberg 1986, Hsu 1992]. The issue with this type of deformation is that the regions of influence of constraint points cannot be varied independently, which is not intuitive enough for our iterative programming approach.

On the other hand, Simple Radial Deformations (SRD) approaches offer great simplicity and efficiency among the point-based spatial deformations. In these schemes, deformations are determined by an arbitrary number of constraints, each consisting of a spherical radius of effect r_i centered on a constraint point C_i with an associated displacement δ_{C_i}. When applied to an object, the result is a collection of smooth, possibly overlapping bumps. As the name implies, sample points of *SRD* are parametrized only by their distance from the constraint points and deformations, thus radiate uniformly in all directions. In contrast to the *DMFFD* approach, all constraint parameters (radius r_i, constraint point C_i, displacement vector δ_{C_i}) are completely independent.

Simple Constrained Object Deformation (SCODEF) presented by [Borrel 1994] is part of *SRD* methods. The user defines a set of constraint points, giving a desired displacement and radius of influence for each. Unlike other radial methods, the deformation effect is local, this is why we chose to use it for the refinement of Virtual Guides. We present next how to apply the *SCODEF* to our programming framework.

4.4.1.2 Virtual Guides Refinement via Point-based Deformation

SCODEF was introduced in the field of geometric modeling and interactive shape edition for producing controlled spatial deformations. It can be viewed as the deformation obtained by creating an arbitrary number of possibly overlapping B-spline shaped "bumps" over the space. The location and height of a bump are defined by a *constraint* and its width by the radius of influence of the *constraints*. *SCODEF* is a constraint based deformation in two senses: first, it is defined using constraints; second, the constraints directly influence the final shape of the deformed objects, and this shape can be fine-tuned by adjusting the radius of influence of each constraint point. This deformation method can satisfy an arbitrary number of constraints, unless two conflicting constraints are applied to the same unique point, in which case a best approximation is calculated. For points lying within the regions of influence of several constraints, a weighted combination of each constraint is applied. To obtain local deformation, B-splines functions are used but any other function having a limited range could be used.

We got inspiration from the *SCODEF* concept to locally modify Position Constraints Virtual Guides. We also get inspired by [Raffin 2000], where an extension of *SCODEF* is presented in order to modify the deformation function without changing the deformation model.

Our contribution is to implement the *SCODEF* method to locally modify the

Position Constraints constructed via Akima splines.

Implementation of *SCODEF* In our programming framework, the human operator can modify the Virtual Guide at a specific point of the spline interpolation without modifying the entire curve. The control point (waypoint) to be modified is called a *constraint point* X_k. Once X_k is defined, the operator modifies the interaction mode (Section 4.3) to be able to "scape" the Virtual Guide and displace the end-effector in space, using a *Soft Virtual Guide*. He/she can then specify the new point $X_k^{'}$ to replace X_k.

A vector of displacement $\vec{d}$ between X_k and $X_k^{'}$ is determined:

$$\vec{d} = \overrightarrow{X_k X_k^{'}}$$

A radius r determines the region of influence of the deformation. It seems natural to choose r proportional to the magnitude of $\vec{d}$:

$$r = g \parallel \overrightarrow{X_k X_k^{'}} \parallel$$

This radius of influence r allows an intuitive control of the Virtual Guide modification. The only parameter to tune is g, for which a compromise must be done between the shape of the curve and the locality of the deformation, i.e, the neighbor points that will be also modified.

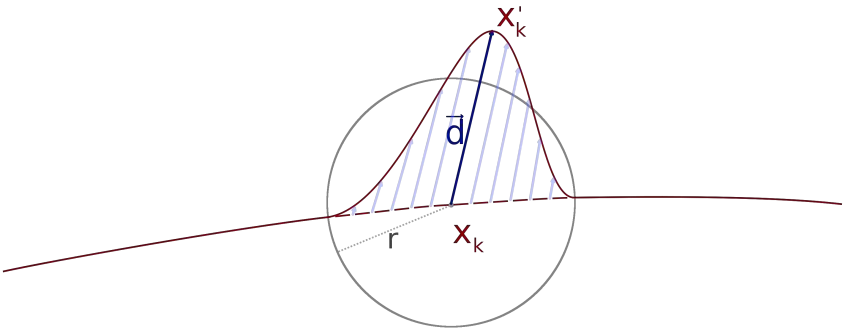


Figure 4.4: SCODEF deformation applied to the constraint point X_k lying on the spline curve. $X_k^{'}$ is the new desired position. In gray, the circle of influence of radius r. In red, the deformed spline. In dark blue, the displacement vector $\vec{d}$ of the constraint keypoint and in light blue the displacement vector of the other keypoints.

A local deformation function F centered at the constraint point X_k and decreasing to zero for points beyond the radius r, must be determined. According to [Raffin 2000], deformation functions can be defined using three required conditions:

- $f(X_k) = 1$ to satisfy the constraint X_k.

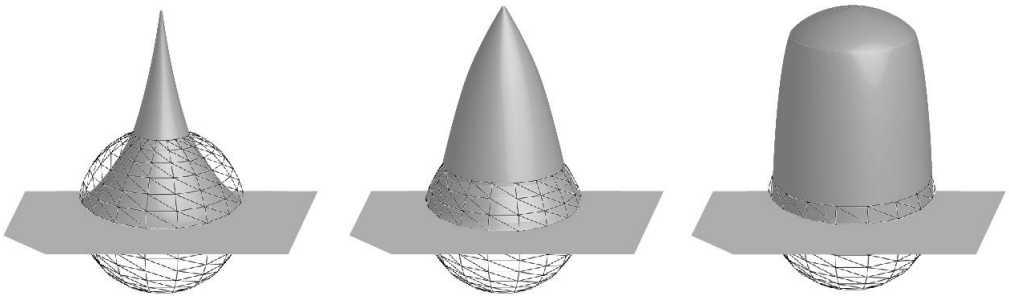


Figure 4.5: Several deformation functions: gate, peak and B-spline. Image inspired from [Raffin 2000].

- $f(r) = 0$ by definition of the radius of influence.
- f is continuous, to ensure the continuity of the deformation.

For example, in Figure 4.5 a gate function, a peak function and the original B-spline function are shown.

In order to obtain a smooth displacement of the Akima spline control points, we apply the fourth degree polynomial $f(x)$ previously defined in equation (4.3). We define x by:

$$x = \frac{(s_{vm,i} - s_{vm,k})}{r}$$

Where $\{s_{vm,i}\}_{i=0:N-1}$ represent the curvilinear abscissa parameter of the Akima spline, and N is equal to the number of interpolation points.

Since the deformation polynomial $f(x)$ is continuous, the *SCODEF* deformation function F is also continuous.

Once the deformation function applied, a list containing the new guide points after the deformation is defined. This provides a new Akima interpolation defining the new Virtual Guide.

To summarize, the deformation steps are:

1. Deformation constraint definition (selecting a guide point and assigning it a new location).
2. Definition of the associate function to the constraint.
3. New Virtual Guide construction via Akima spline interpolation.

Perspective: The concept of the radius of influence greatly helps to intuitively control the locality of the deformation. We previously saw that the *SCODEF* region of influence is circular. It was demonstrated in [Raffin 2000] that different

areas of influence can be defined such as ellipsoids, star-shaped and triangles. Also a curvilinear displacement was proposed to extend the "straight deformations" originally proposed for *SCODEF*. We did not applied this in our work, but there are interesting solutions to explore.

Drawback: The number of control points of the Akima spline can be a limitation for the proposed modification approach. For example in the case where the point that the user wants to modify is "too far" from a control point. In this case, the point to be modified can be added as a control point X_k.

4.4.2 Portion Modification of a Virtual Guide

The previous method is useful to locally modify the trajectory, however sometimes the user needs to modify a whole portion of it. To do so, a partial demonstration of the portion to modify can be done using kinesthetic teaching and the interaction modes presented in Section 4.3.

Initial and final positions of the partial demonstration do not always match a control point of the guide. To merge the new portion with the rest of the guide, we propose to use the previously explained point deformation method to modify the closest points on the base guide for matching the first and last points on the new guide portion.

Let X_1 and X_2 be the constraint points of the current guide. $X_1^{'}$ and $X_2^{'}$ are the initial and final points of the new guide portion, respectively (see Figure 4.6). The constraint points are defined as the two base guide's points which are closest to $X_1^{'}$ and $X_2^{'}$, respectively. We obtain X_1 and X_2 by calculating the distance from $X_1^{'}$ and $X_2^{'}$ to the base guide's control points. Then we select the two control points of minimum distance. Displacement vectors $\vec{d_1}$ and $\vec{d_2}$ are determined between X_1 and $X_1^{'}$, and between X_2 and $X_2^{'}$. A radius of influence r must be defined for both constraints. Again, we choose the radius proportional to the magnitude of the displacement vector.

$$\vec{d}_1 = \overrightarrow{X_1X_1^{'}}; \qquad r_1 = g_1 \parallel \vec{d}_1 \parallel$$

$$\vec{d}_2 = \overrightarrow{X_2X_2^{'}}; \qquad r_2 = g_2 \parallel \vec{d}_2 \parallel$$

The radius $r1$ and $r2$ allow a more intuitive control of the deformation. The only parameters to tune are g_1 and g_2, where a compromise must be done between the shape of the curve and the area of influence of the deformation, i.e, the neighbor points that will be deformed.

A local deformation function $F(x)$ is defined, as done previously in Subsection 4.4.1.2, using the polynomial of equation (4.3).

The deformation function $F(x)$ is applied to both X_1 and X_2 to obtain the new control points, including $X_1^{'}$ and $X_2^{'}$.

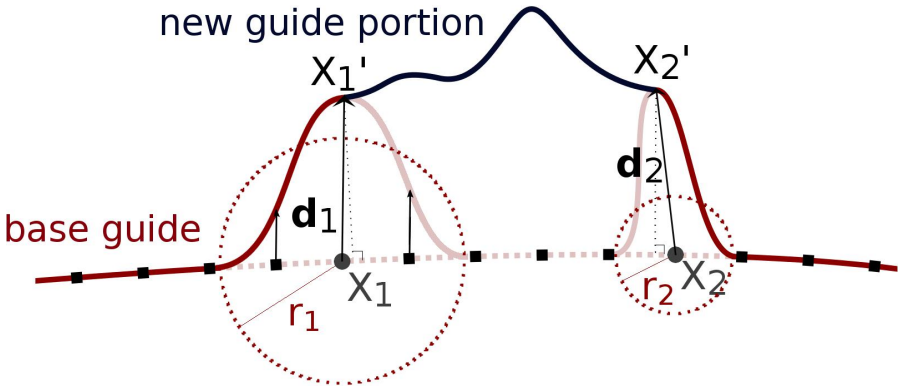


Figure 4.6: Local refinement applied to the constraint points X_1 and X_2 lying on the base guide. $X_1^{'}$ and $X_2^{'}$ are the initial and final point, respectively, of the new guide portion.

The new control points of the Virtual Guide are stored in a vector defined by the modified base guide's control points before X_1 and after X_2, along with the new guide portion's control points. Finally, we perform a new Akima spline interpolation to define the new Virtual Guide.

4.5 Iterative Programming of 6D Virtual Guides

During the execution of a complex task in space, it would be difficult to concentrate in both translation and orientation of the robot. In this Section, we propose to realize the trajectory programming in two phases to reduce the cognitive load of the user and enhance the programming experience by separating the translation and the rotation programming.

The first time we record both translation and orientation. The second time we use the virtual mechanism's translation only in order to just modify the orientation.

1. The user does a first demonstration of the trajectory (Positions) using the cobot in gravity compensation mode.

2. A Position Constraint Virtual Guide is constructed and activated.

3. While being guided through the Position Constraint trajectory, the user does a demonstration of the tool orientations. The orientation of the tool x_R is recorded without adding new keypoints to the curve. For translation, we use the position of the virtual mechanism x_{vm} instead of the position of the tool since we want to keep the initial position information recorded in step 1. We could also just record the orientation of the tool and fusion both informations of translation and orientation. This is just an implementation issue.

4. A 6D Virtual Guide is constructed and activated.

4.5.1 Refinement of Orientation Components

The approaches presented in Subsections 4.4.1 and 4.4.2 can be applied to $\mathbb{SE}(3)$. Thus, the orientation components can also be refined locally or modified on a portion of the 6D Virtual Guide. This functionality can be used iteratively using the *Soft* interaction mode presented in Section 4.3.

4.6 Conclusion

In this chapter we presented a new intuitive, iterative and assisted framework for programming 6D Virtual Guides. First, we defined the roles allocation: the human operator masters the action plan and the cobot assists the human passively and under explicit demand. Next, we proposed a solution to edit Virtual Guides using three interaction modes (*Soft*, *Hard* and *Null* Virtual Guides): guides can be locally refined or modified by the user in both Cartesian position and orientation. Especially we proposed to use Virtual Guide Assistance through an iterative process to relieve the users. This method aims at reducing the cognitive load of the human operator and enhance the programming experience by allowing the programmation of the translation and the rotation iteratively for 6D Virtual Guides. Two experiments showing the advantages of using Virtual Guides Assistance and our programming framework are presented in the following chapter : Chapter 5: "Experimental Evaluation".

Chapter 5

Experimental Evaluation

> *Les conclusions théoriques n'ont de valeur que s'il est montré qu'elles permettent effectivement d'obtenir des résultats concrets.*

Luc Joly, PhD Thesis, 1997

Contents

In this chapter we present two main experiments conducted using the *Virtual Guides Assistance* and the *Iterative Programming Approach* explained in Chapter 2: "Virtual Guides Definition via Virtual Mechanisms", Chapter 3: "Virtual Guides Construction" and Chapter 4: "Iterative Virtual Guides Programming". The main objective of the following experiments is to show the influence of the *Virtual Guides Assistance* on the execution of a comanipulation task and the influence of our *Iterative Approach* on the programming of a comanipulation task, both in terms of time and accuracy. We will also show that our approaches improve the user experience when programming or executing a task with a cobot.

(a) 3-DOF ISybot cobot (b) 6-DOF ISybot cobot

Figure 5.1: Back-drivable cobots with screw-and-cable transmissions

Both experiments were conducted on back-drivable cobots with screw-and-cable transmissions [Garrec 2010] developed by *CEA* and ISybot[1] (see Figure 5.1). The main characteristics of the two versions we used are summarized in Table 5.1.

Table 5.1: Main characteristics of ISybot cobots: PK0 and PK2

Cobot	**Number of axis**	**Weigth** (kg)	**Payload** (kg)	**Working radius** (m)
PK0	3	40	14	1.2
PK2	6	50	8	1.2

5.1 Statistic analysis

In this section we define some useful concepts to better understand the analysis of the results obtained in the next sections of this chapter.

[1] http://www.sybot-industries.com/

We will use two types of variables for the statistic analysis of our experiments:

- Dependant variable: A variable measured by the experimenter to be analyzed (quantitative).
- Independent variable: A variable manipulated by the experimenter to determine its effects on the dependent variable (qualitative).

In general, a statistic analysis combines several independent variables but just one dependent variable. In this chapter, the standard deviation of a dependent variable will be represented as $\sigma_{variable}$.

Hypothesis test The main purpose of statistics is to test a hypothesis. A hypothesis is an educated guess about something in the world around us. It should be testable, either by experiment or observation. A statistical hypothesis test is a method of statistical inference. Commonly, two statistical data sets (groups) are compared. A hypothesis is proposed for the statistical relationship between the two data sets, and this is compared as an alternative to an idealized null hypothesis that proposes no relationship between two data sets.

- Null hypothesis (H0): there is no difference between the two data sets.
- Alternative hypothesis (HA): there is a significance difference that is not due to random sampling.

To decide between the null hypothesis and the alternative hypothesis, it is useful to identify two conceptual types of errors (type I and type II), and specify some limits, for example, how much type I error is allowed. This is called the significance level (α), understood as the probability threshold below which the null hypothesis will be rejected. Common values are $\alpha = 5\%$ and $\alpha = 1\%$.

- Type I error: reject H0 when it is true, it is called α.
- Type II error: accept H0 whereas it is false, it is called β.

The more we reduce α, the more we increase β, however, $\beta \neq 1 - \alpha$.

ANOVA An ANOVA (Analysis of variance) test is a way to find out if survey results or experiment results are significant. In other words, they help us figure out if we need to reject or accept the null hypothesis. Basically, we test groups to see if there is a difference between the means of the measured Dependant Variable.

An F-statistic is computed for each tested hypothesis. It is any statistical test in which the test statistic has the asymmetric F-distribution under the null hypothesis. The probability, assuming the null hypothesis is true, of observing a result at least as extreme as the test statistic, is called the p-value. In other words, the p-value represents the probability the results could have happened by chance.

- If p-value $> .05$, then the risk is too great to reject H0, and we declare the absence of significant difference according to the conditions.
- If p-value $< .05$, then we can reject H0 and declare that our conditions lead to significant differences.
- If $.05 <$ p-value $< .10$, we can consider that the observed phenomenon is a trend that could become significant if we increase the number of subjects of the study.

Repeated measures ANOVA A "repeated measures ANOVA" is an ANOVA test where the same group of participants is being measured over and over again, thus they are not independent. In this case, we can not use the same approach as in a regular ANOVA test. Indeed, each repetition measure is naturally correlated to the subject. This is why we define the repetitions as a factor of the ANOVA test. This kind of factor, based on an internal condition, is what we call a *within factor* as opposed to *between factors*, which are based on external conditions of the experiment (such as subjects' age or gender).

Post-hoc tests An ANOVA test tells us if the null hypothesis is true or may be rejected, but in the latter case, it does not tell us where the difference is. To determine which groups differ from another we need to make complementary tests, called post-hoc tests.

One of the most used post-hoc tests after an ANOVA is the Tukey Test [Tukey 1949]. However, this test makes the assumption that the studied groups are independent, which is not the case for the repeated measures ANOVA groups.

We chose to use separate paired-samples t-tests[2] [Student 1908] and a sequentially acceptive step-up Benjamini procedure [Benjamini 1995] based on the conservative Bonferroni correction. This approach is specially adapted to the "repeated measures ANOVA test". Bonferroni laws says that if we do N comparisons[3], the type I error cannot be superior to $N \cdot \alpha$. We consider $FWER$ as the probability that a family of comparisons contains at least one type I error. If we want to keep $FWER = \alpha$, then we should simply use a threshold of α/i, where i represents the current comparison. The Benjamini procedure proposes to use an adaptive threshold of $\alpha(N + 1 - i)/N$, which is less conservative than the initial threshold correction and avoids error type II when N increases.

We will use this formalism to analyse the data of our experiences. The interested reader is referred to [Lomax 2013] for further details on statistic tests and post-hoc analysis.

[2] A t-test, also called t-Student test, evaluates if there is a significant difference between two groups distributions. When the groups are dependent from each other, (which is the case for groups of a repeated measures ANOVA) we use a paired t-test.

[3] A comparison is an independent t-test for each combination of factors.

5.2 Experiment 1: Virtual Guides Assistance advantages

In Chapter 2: "Virtual Guides Definition via Virtual Mechanisms" and Chapter 3: "Virtual Guides Construction" we presented our virtual guides implementation for a comanipulation context using Akima splines. These virtual guides were then used in Chapter 4: "Iterative Virtual Guides Programming" as part of our iterative programming approach. The main objective of this first experiment is to show the advantages of virtual guides assistance on a comanipulation task with a cobot, to validate the interest of using them on our cobot programming approach.

We designed a user study in order to analyse the participants' peformance of a comanipulation task with a cobot (time and accuracy) and to observe how novice users perceive the Virtual Guides Assistance (helpful and intuitive).

The following hypotheses were tested:

- H1: *Virtual Guides Assistance* reduces the execution time of the task.
- H2: *Virtual Guides Assistance* improves the users' task performance.
- H3: *Virtual Guides Assistance* is perceived as helpful by the users.
- H4: *Virtual Guides Assistance* is easy to use.
- H5: *Virtual Guides Assistance* is more useful when the task requires higher level of attention.

In order to verify these hypotheses, we designed a specific comanipulation experience using the approaches explained in Chapter 2: "Virtual Guides Definition via Virtual Mechanisms", Chapter 3: "Virtual Guides Construction" and Chapter 4: "Iterative Virtual Guides Programming".

5.2.1 Task definition

This first experiment seek to simulate a sanding task where an operator and a robot collaborate to **clean** a metal sheet. The general task is to follow a sweeping trajectory with the 3-DOF ISybot cobot of Figure 5.1(a) appended with an industrial sander tool. The sander tool is thus used to erase the 2 red marks shown in Figure 5.2.

It was necessary to choose a task where no technical expertise is needed in order to minimize the impact of different skill levels of participants. Also, to execute a real sanding task experiment, security measures would have been harsher and dust and noise could have disturbed the user. Finally, since we wanted to analyse the effects of virtual guides only in a trajectory level (without measuring forces), we sticked to an erasing task. However, the system could be used in more dynamic and complex scenarios.

In order to verify our hypotheses, two tasks are studied:

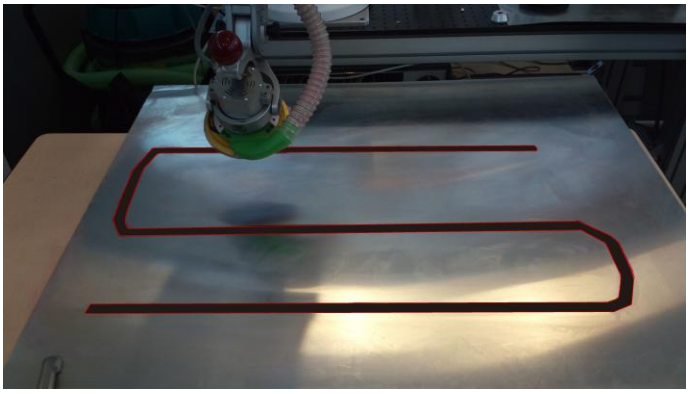


Figure 5.2: Sweeping trajectory (in red). In order to reproduce the same trajectory for each participant and each repetition a black tape canvas was used.

1. *Case A* consists on cleaning the metal sheet while following the sweeping trajectory,

2. *Case B* consists in performing the same task with an obstacle blocking the trajectory.

The obstacle in *Case B* is a symbol that represents changes in the task, as for example a new metal sheet to sand that has a different form or holes to avoid. It also demands the user a higher level of attention compared to *Case A*.

In both cases, we compare two *Assistance Modes*:

1. *Gravity Compensation Assistance*,

2. *Virtual Guides Assistance*.

For the first *Mode*, users are able to move the robot freely. For the second *Mode*, an expert user programs the virtual guide aimed to assist the participants during the execution of the task, as explained in Chapter "Virtual Guides Definition via Virtual Mechanisms". The expert user also modifies a portion of the guide generated for *Case A*, using the iterative approach explained in Section 4.4.2, in order to realize *Case B*.

Programming virtual guides by an expert user To program a virtual guide for *Case A - Virtual Guides Assistance Mode*, the expert user uses a function that takes two non-adjacent vertices of a rectangle and generates a set of points describing a sweeping trajectory filling the rectangle. These vertices are shown by demonstration to the cobot and recorded using the lower green button placed on the 3rd axis of the cobot, as shown in Figure 5.3. With the previously generated set of points, the virtual guide represented in Figure 5.4,i) is created.

(a) Cobot programming interface. Lower green button is used to record key points. Upper black button is used to end the recording and activate the virtual guide.

(b) Expert user recording a key point

Figure 5.3: Interaction with the 3-DOF ISybot collaborative robot

For *Case B - Virtual Guides Assistance Mode* there is an obstacle blocking the trajectory. As a consequence, the previous approach must be completed by a modification done by the expert. This modification of the guide is done using the approach explained in Section 4.4.2. When the guide is active, the expert user is able to move along the trajectory while being constrained to move in other directions. Once arrived near the obstacle, the user is able to escape the guide, as presented in Section 4.3, by using the lower green button placed on the 3rd axis of the cobot. Then, the partial modification is shown by demonstration to the cobot and recorded using the upper black button (see Figure 5.4,i). After recording is stopped, the upper button triggers the refining algorithm and a new guide is created (see Figure 5.4,ii).

Our approach allowed the expert user to modify the first guide while being assisted by it, in order to adapt to a change of environment. In this case of application, only the demonstration of two points and the demonstration of a portion of

trajectory were needed to obtain two guides. **Thanks to our method, the user just had to modify locally the guide to get around the obstacle, avoiding the entire guide demonstration with the cobot.**

For the *Virtual Guide Assistance Mode*, the controller gains were tunned as explain in Chapter 2: "Virtual Guides Definition via Virtual Mechanisms" and set as follows:

$$K = \begin{bmatrix} 10000 & 10000 & 10000 \end{bmatrix} N/m$$

$$B = \begin{bmatrix} 400 & 400 & 400 \end{bmatrix} N/m.s^{-1}$$

$$K_s = \begin{bmatrix} 5000 \end{bmatrix} N/m \text{ and } N/m.rad$$

$$B_s = \begin{bmatrix} 5 \end{bmatrix} N/m.rad.s^{-1}$$

Sampling time was set at $1ms$.
During the modification of the virtual guide, the expert user set:

$$d_{max} = 0.01m$$
$$\alpha_1 = 1$$
$$\alpha_2 = 2$$

These values were chosen through trial-and-error to meet the conditions described in 4.4.2. An example of the influence of the choice of α_1 et α_2 is shown in Figure 5.4,ii.

5.2.2 Protocol

We recruited 14 participants from different research laboratories (between 22 and 33 years old, 5 women). Nine participants stated they had prior experience with robots, ranging from robotic courses to hands-on experience with industrial robots.

All participants were asked to perform the tasks *Case A* and *Case B* (ie. cleaning task; cleaning task with an obstacle), in both Modes: *Gravity Compensation* and *Virtual Guides Assistance*, resulting to four test conditions. The four (2×2) test conditions were presented in a randomized order to avoid biasing the results towards the last tested *Mode* due to training effects. For each condition, the participants were asked to perform the task 3 times in a row (*Repetitions*). In total, the participants performed $12 = 2 \times 2 \times 3$ (*Case*Mode*Repetition*) cleaning tasks. At the beginning of each condition, the *Case* and the *Mode* were presented to show the participants how to use the cobot. Then, they were able to familiarize themselves with the system and try the tested condition on their own. When a condition was completed, it was asked to the participants to fill the same post-condition survey – Likert-scale

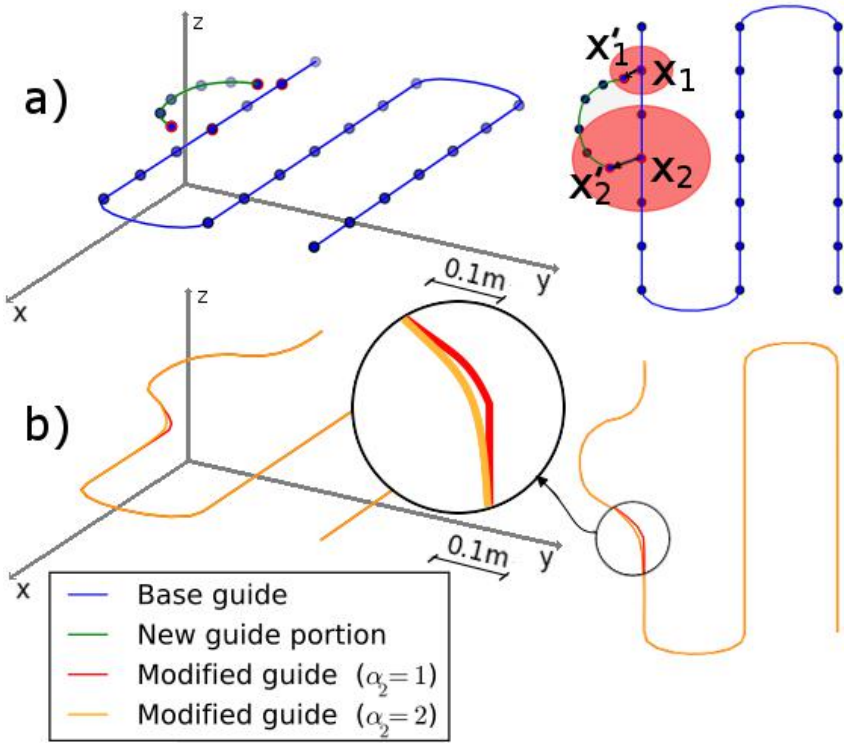


Figure 5.4: a) Base guide and new guide portion. The areas of influence are represented by the red disks which are centered at the constraint points X_1 and X_2, lying on the base guide. Each radius of influence is α times proportional to the magnitude of the corresponding displacement vector, as explained in Section IV. Here, $\alpha_1 = 1$ and $\alpha_2 = 2$. $X_1^{'}$ and $X_2^{'}$ are the initial and final points of the new guide portion. b) Resulting virtual guide after local refinement. The choice of each α has an impact on the final curve. In our case, we compare $\alpha_2 = 1$ and $\alpha_2 = 2$ (α_1 is fixed to 1). $\alpha_2 = 2$ allows to reach another key point of the base guide, thus improving smoothness of the resulting curve.

survey with a rating from 1 (strong disagreement) to 7 (strong agreement).

To validate our hypotheses, we recorded the following **measures**:

1. Time of execution of the task for the twelve conditions, to validate H1 and H3 – Virtual Guides Assistance reduces the execution time of the task and is perceived as helpful by the users.
2. Observed collisions in Case B, to validate H2 – Virtual Guides Assistance improves the user's task performance.
3. Survey results for both Cases and both Modes, to validate H2-H5 – Virtual Guides Assistance improves the user's task performance, is perceived as helpful by the users, is easy to use and is more useful when the task requires higher level of attention.

5.2.3 Results

We performed a Repeated-measures ANOVA for the time execution of the task as the Dependent Variable on three *within factors* (internal conditions of the experience):

1. Modes
2. Cases
3. Repetitions

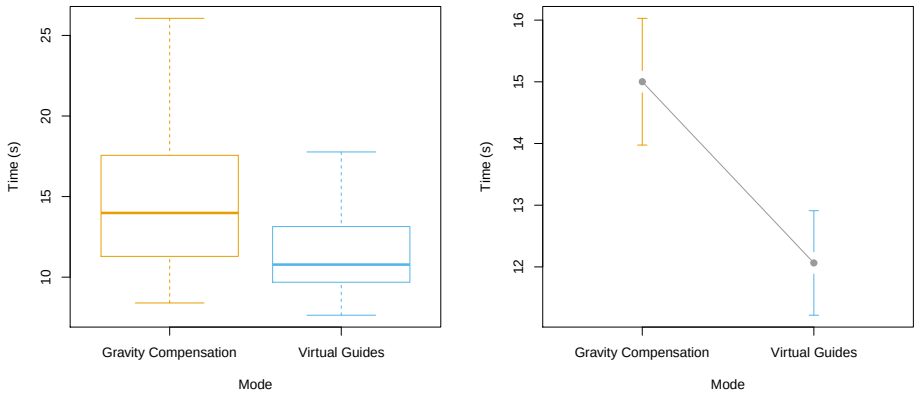
We performed independent repeated-measures ANOVA for each survey question on the first two *within factors* mentioned above. *Repetitions* were not taken into account on the post-condition surveys. For all ANOVA tests, participants were grouped by one *between factor* (external conditions of the experience): their previous *Experience* with robots. Other factors as *Age* and *Gender* had no influence on the time of execution.

We set our significance level at $\alpha = 0.05$, which means there is at least 5% chance of having a difference between means of the studied variables. We consider a result strongly significant when p-value < 0.01 and poorly significant when p-value < 0.1. We use poorly significant results to show trends and give further analysis of data, however we do not draw conclusions based on these results. Post hoc pairwise comparisons were computed using non-pooled error terms (i.e., by computing separate paired-samples t-tests; sequentially acceptive step-up Benjamini procedure [Benjamini 1995], with an alpha level of 0.05.)

We use box plots and group means plots to visualize both main effects and interaction effects on the dependent variables. Box plots are useful to represent the median value (less affected by outliers) and data dispersion, while group means plots indicate a more "easy to see" relation between the mean values of the studied variables. On the group means plot, the bars represent a confidence interval of 95%.

All the results and graphs were obtained using R language[4].

[4]https://www.r-project.org/



(a) Box plot

(b) Group means with confidence interval of 95%

Figure 5.5: Main effect of the *Assistance Mode* on time

5.2.3.1 Time and collision results

We found a significant main effect of the *Modes* on the execution time of the task ($p - value < .01$). As shown in Table 5.2 and Figure 5.5, participants were faster when they used the *Virtual Guides Mode*. This validates hypothesis H1 – *Virtual Guides Assistance* reduces the execution time of the task.

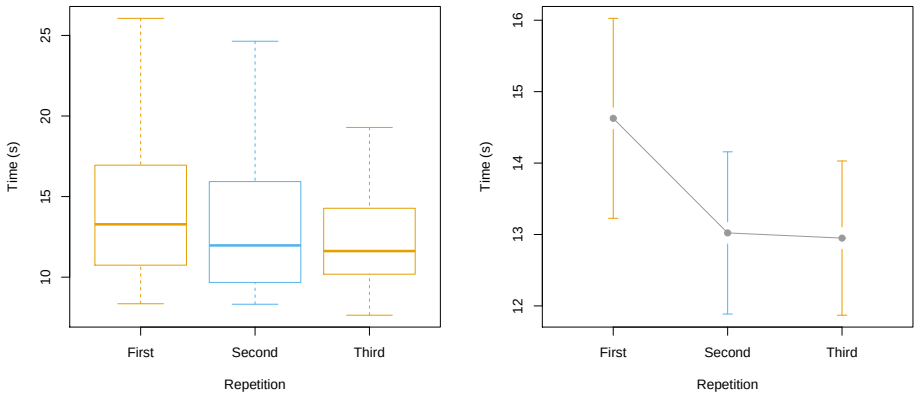
Table 5.2: Effect of the *Modes* on the execution time of the task

Mode	$mean_{Time}$ (s)	σ_{Time} (s)
Gravity Compensation	15	4.73
Virtual Guides	12.06	3.90

We also found a significant main effect of the *Repetitions* on the execution time of the task ($p - value < .001$). Post hoc analyses showed that the first *Repetition* was longer than the second ($p-value < .025$) and the third *Repetitions* ($p-value < .001$). These results are shown in Table 5.3 and Figure 5.6, and they confirm there is a training effect through *Repetitions*.

Table 5.3: Main effect of the *Repetitions* on the execution time of the task

Repetition	$mean_{Time}$ (s)	σ_{Time} (s)
1^{st}	14.63	5.23
2^{nd}	13.02	4.24
3^{rd}	12.95	4.03



(a) Box plot

(b) Group means with confidence interval of 95%

Figure 5.6: Effect of the *Repetitions* on the execution time of the task

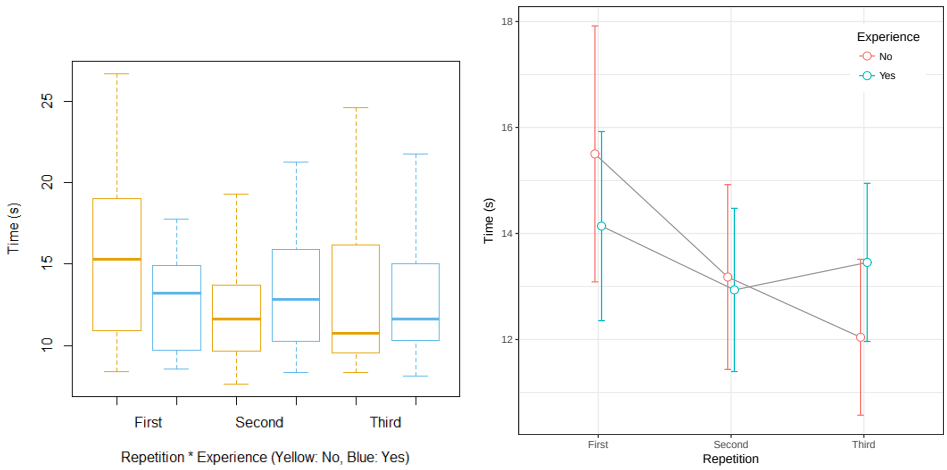
Moreover, there is a significant effect of interaction between *Repetitions* and participants' previous *Experience* with a robot ($p - value < .01$). Post hoc analyses showed that the first *Repetition* was longer than the second ($p - value < .025$) and the third *Repetitions* ($p - value < .025$), only in the group of participants without previous Experience with robots. These results are shown in Table 5.4 and Figure 5.7. This indicates us that the participants who did not have previous *Experience* with robots had a greater improvement in their execution time through *Repetitions* than those who had previous *Experience*. These results support the hypothesis H3 – *Virtual Guides Assistance* is perceived as helpful by the users.

Table 5.4: Interaction effect of *Repetitions* and *Experience* with robots on the time execution of the task

Repetition	**Experience**	$mean_{Time}$ (s)	σ_{Time} (s)
1^{st}	No	15.50	5.16
2^{nd}	No	13.18	3.72
3^{rd}	No	12.04	3.14
1^{st}	Yes	14.14	5.27
2^{nd}	Yes	12.94	4.55
3^{rd}	Yes	13.45	4.41

We found a low effect of the *Case* on the execution time of the task ($p - value = 0.064$). As shown in Table 5.5 and Figure 5.8, participants were slower when there was an obstacle obstructing the sweeping trajectory.

Moreover, there is a significant interaction effect of *Mode* and *Case* on the execution time of the task ($p - value < 0.05$) (see Table 5.6). Post hoc anal-



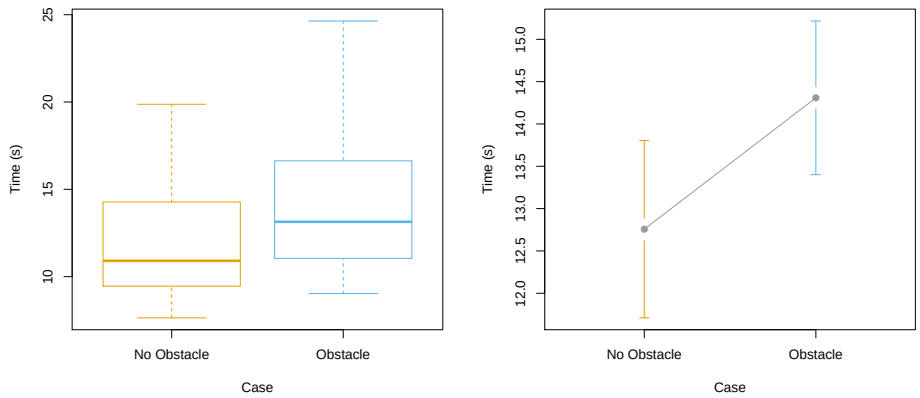
(a) Box plot

(b) Group means with confidence interval of 95%

Figure 5.7: Interaction effect of *Repetitions* and *Experience* with robots on the time execution of the task

Table 5.5: Effect of the *Case* on the execution time of the task.

Case	$mean_{Time}$ (s)	σ_{Time} (s)
A (Obstacle)	12.76	4.82
B (No Obstacle)	14.31	4.18



(a) Box plot

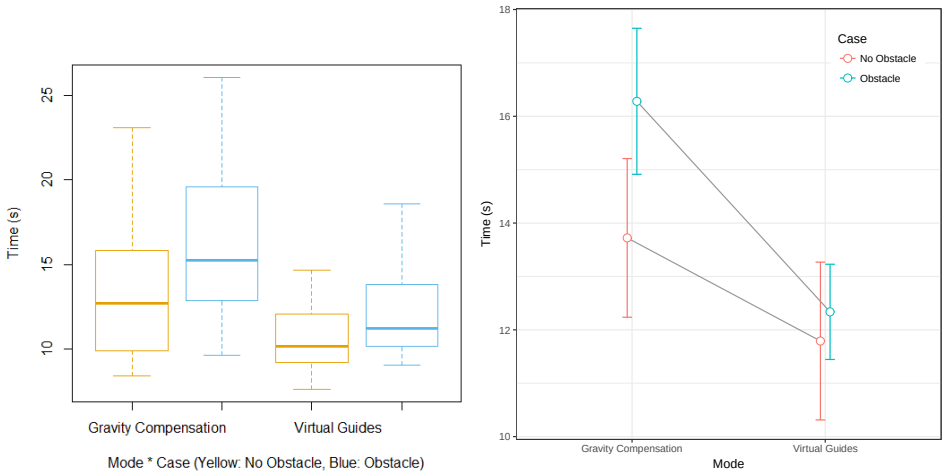
(b) Group means with confidence interval of 95%

Figure 5.8: Main effect of the Case on time

ysis showed that there was a significant influence of the *Mode* only for *Case A* ($p - value < 0.001$). In other words, the *Virtual Guides Assistance Mode* made a bigger difference on the execution time of the task where there was an object obstructing the sweeping trajectory (see Figure 5.9). This result validates H5 – *Virtual Guides Assistance* is more useful when the task requires higher level of attention.

Table 5.6: Interaction effect of *Modes* and *Cases* on the time execution of the task

Mode	**Case**	$mean_{Time}$ (s)	σ_{Time} (s)
Gravity Compensation	A	13.72	4.77
Virtual Guides	A	11.79	4.74
Gravity Compensation	B	16.28	4.39
Virtual Guides	B	12.34	2.86



(a) Box plot

(b) Group means with confidence interval of 95%

Figure 5.9: Interaction effect of *Modes* and *Cases* on the time execution of the task

Finally, for *Case B*, collisions only occurred when the users used the *Gravity Compensation Assistance Mode*. This result goes in the direction to validate H2 – *Virtual Guides Assistance* improves the users' task performance.

5.2.3.2 Survey results

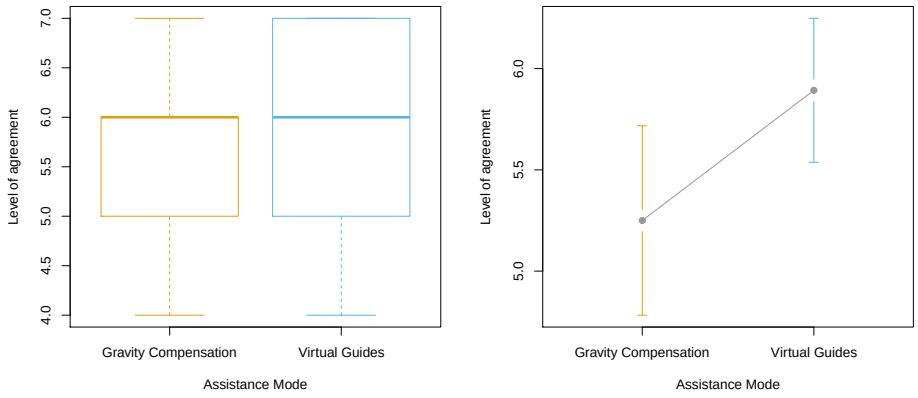
For clarity reasons, the information of the participants' answers to the questions of our user study for each *Mode* are summarized in Table 5.7. With this survey, we observed a significant main effect of the *Mode* on the execution time of the task.

In *Virtual Guides Assistance Mode*, participants found that:

Table 5.7: Survey results of the user study for the *Assistance Modes*

	Gravity Compensation		Virtual Guides		
Question	*mean*	σ	*mean*	σ	**p-value**
Q1: Do you think the task was easy to perform?	5.32	1.44	5.68	1.15	ns
Q2: Do you think that you performed the task well?	5.25	1.21	5.89	0.92	0.0135
Q3: Did you feel you performed the task precisely?	4.82	1.17	5.64	1.06	0.0031
Q4: Did you feel comfortable during the task execution?	5.18	1.16	5.36	1.39	ns
Q5: Did you feel stressed during the task execution?	1.93	1.09	1.96	1.23	ns
Q6: Do you think the robot was easy to work with?	5.11	1.26	5.39	1.37	ns
Q7: Do you think the robot was helpful during the task execution?	3.82	1.59	5.29	1.65	0.0234

1. They performed the task better (see Figure 5.10).
2. They felt more precise in the execution in the task (see Figure 5.11)
3. The cobot was more helpful (see Figure 5.12)



(a) Box plot

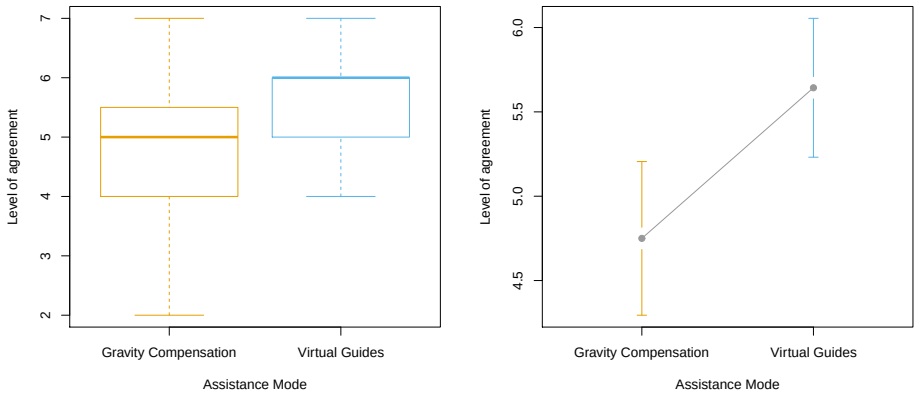
(b) Group means with confidence interval of 95%

Figure 5.10: Effect of *Modes* on the participants' perception of the task performance

Like suggested by the previous analyses, this questionnaire validates H2 and H3 – *Virtual Guides Assistance* improves the users' task performance and *Virtual Guides Assistance* is perceived as helpful by the users.

We also found a significant main effect of the *Experience* on the participants' perception of the task difficulty ($p - value = 0.0335$). It is shown in Table 5.8 and Figure 5.13, that participants who had previous *Experience* with robots found the task easier.

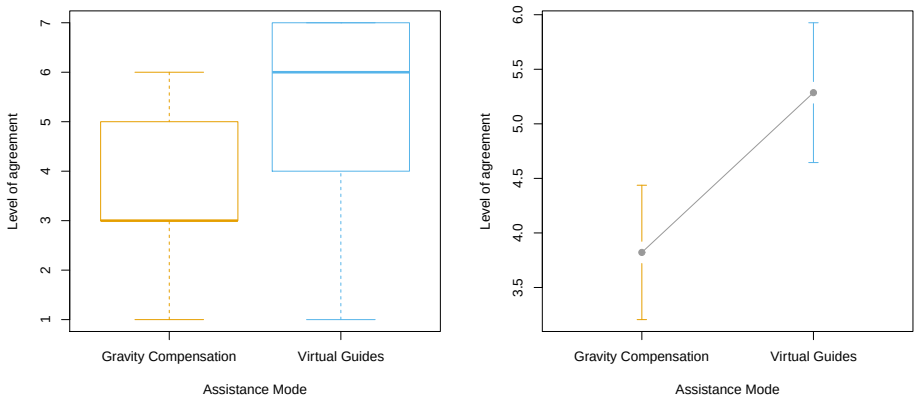
There were no significant interaction effects between *Modes* and participants' previous *Experience* with robots. This means the benefits of the *Virtual Guides Assistance Mode* highlighted above do not depend on the user's *Experience* with robots.



(a) Box plot

(b) Group means with confidence interval of 95%

Figure 5.11: Effect of *Modes* on the participants' perception of the accuracy of the task performance



(a) Box plot

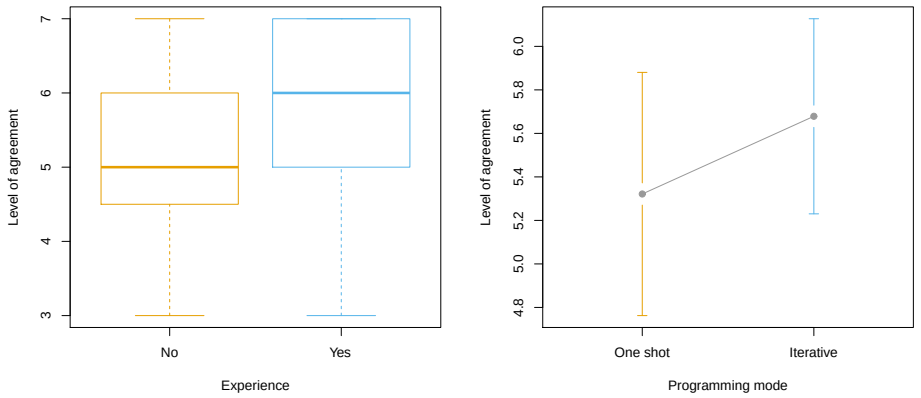
(b) Group means with confidence interval of 95%

Figure 5.12: Effect of *Modes* on the participants' perception of the helpfulness of the cobot

From Table 5.7, we can see that there is no significant main effect of the *Mode* for questions Q1, Q4 and Q6, so we cannot validate hypothesis H4 – Virtual Guides Assistance is easy to use. However, we can see that, in average, participants found the task easy to perform for both *Modes* and slightly easier when using the *Virtual Guides Assistance*. Also, they felt more comfortable performing the task and they found the robot easier to work with when they were assisted by the Virtual Guides.

Table 5.8: Effect of the *Experience* on the participants' perception of the task difficulty

Experience	$mean_{Time}$ (s)	σ_{Time} (s)
No	4.95	1.36
Yes	5.80	1.19



(a) Box plot

(b) Group means with confidence interval of 95%

Figure 5.13: Effect of *Experience* on the participants' perception of the difficulty of the task

This goes in the direction to validate H4. A possible reason for not getting a significant result for this hypothesis might be that the task was too easy, so it is harder for participants to perceive the difference of influence of the two *Assistance Modes*. Or, having more participants could also increase the chances of getting a significant result.

Finally, there was no significant main effect of the *Mode* on the participants' stress while executing the task. We can see in Table 5.7 that, in average, participants' did not feel stressed when using both *Assistance Modes*.

Comanipulation Participants were free to manipulate the cobot the way they felt more comfortable. In general, they used the tool handler with one hand as shown in Figure 5.14.

5.2.4 Conclusion of experiment 1

Using the previous results, we can make the following conclusions about the initial hypotheses of the experiment:

Figure 5.14: Participants executing the task with the cobot

H1 The main effect of the *Assistance Mode* on the time execution of the task validates the hypothesis H1 – *Virtual Guides Assistance* reduces the execution time of the task.

H2 Several results supported the hypothesis H2 – *Virtual Guides Assistance* improves the users' task performance – and though we validate it:

- There were no collisions while using the *Virtual Guides Assistance*.
- Participants found that they performed the task better (i.e. best compromise between time and accuracy) and more precisely with the *Virtual Guides Assistance*.

H3 Several results supported the hypothesis H3 – *Virtual Guides Assistance* is perceived as helpful by the users – and though we can also validate it:

- Participants found that the cobot was more helpful with the *Virtual Guides Assistance*.

- From the interaction effect of *Repetitions* and user's *Experience* with robots, we found out that the participants with no previous *Experience* had a better improvement through *Repetitions* than participants with *Experience* with robots.

H4 We couldn't validate hypothesis H4 – *Virtual Guides Assistance* is easy to use – since the task was generally perceived as easy for the users in both *Assistance Modes*. However we confirmed that the task remains easy while using the *Virtual Guides Assistance*.

H5 From the interaction effect of *Modes* and *Cases*, we found out the *Virtual Guides Assistance Mode* made a bigger difference on the execution time of the task when there was an object obstructing the sweeping trajectory. Thus, we validate hypothesis H5 – *Virtual Guides Assistance* is more useful when the task requires higher level of attention.

With this experiment we confirmed the advantages of using virtual guides for comanipulation tasks. On the next experience we will show an extended application using 6D virtual guides (translation and orientation) generated by demonstration by the users without using any automatic algorithms. We will also analyze the impact of our approach on users that not only execute a task but **program a cobot using our iterative programming method**.

5.3 Experiment 2: Iterative Programming of a comanipulation task

This second user study was designed in order to observe how novice users perceive the *Iterative Programming Mode* and to analyse the impact of our approach on a comanipulation task programming by comparing it to a classic programming mode used in the industry that we called *One Shot* and provides a gravity compensation assistance.

The following hypotheses were tested:

- H1: The *Iterative Programming Mode* reduces the programming time of the task.
- H2: The *Iterative Programming Mode* improves the accuracy of the results.
- H3: The *Iterative Programming Mode* is intuitive and comfortable to use.
- H4: The *Iterative Programming Mode* is perceived as helpful by the users.
- H5: The *Iterative Programming Mode* reduces the user's physical effort and cognitive overload to program the task.
- H6: The comanipulation task with the cobot is not perceived as stressful.

We will now verify these hypotheses with the following comanipulation task programming experience.

5.3.1 Task definition

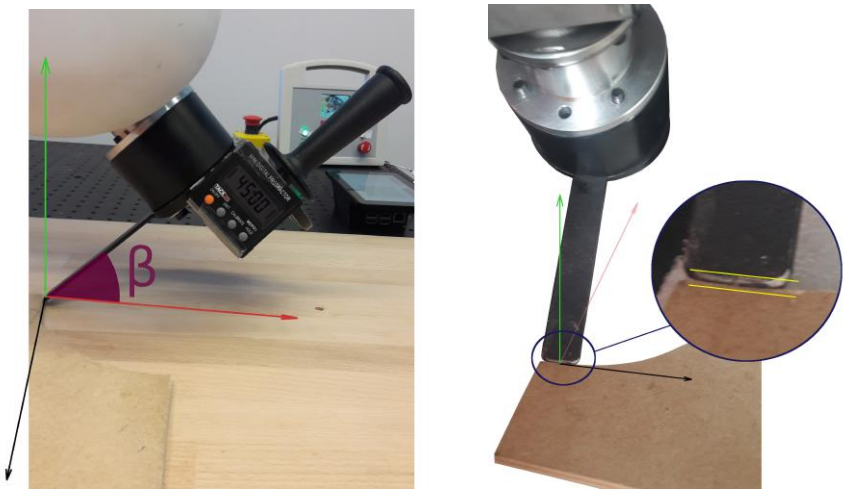
The following experience is conducted with a 6-DOF ISybot collaborative robot of Figure 5.1(b) and a scraper tool. The general task consists in programming the cobot to follow the contour of a 2cm thickness wood part (see Figure 5.15) respecting two conditions. As shown in Figure 5.16, participants have to:

1. Keep the tooltip orientation angle at 45° with the vertical axis (Figure 5.16(a)).
2. Stay parallel to the tangent of the curve at each point of the trajectory (Figure 5.16(b)).

The scraper tool (with a rectangular form) was designed for the experience in order to increase the difficulty of the task, by asking the participants to keep the tool parallel to the tangent of the curve.

Figure 5.15: Programming task setup. The path to follow is highlighted in red

An inclinometer is placed on the tool in order to help the users to maintain the 45° angle instruction. Participants have to follow a specific portion of the top contour designed with 4 straight lines, convex radius, two concave different radius and a right angle (270°). The path to be learnt is highlighted in red on Figure 5.15.



(a) (b)

Figure 5.16: . *a)* The scraper tool orientation angle β must be kept at 45° within the vertical plane. *b)* The scraper tool must be kept parallel to the tangent of the curve at each point of the path.

To program the cobot, participants must record discrete key points of the trajectory using the *rec button* – upper button placed on the 4^{th} axis of the cobot, indicated by the red arrow in Figure 5.15. These points are then interpolated via Akima splines and quaternion interpolations and used to create *6D Virtual Guides* as explained in Chapter 3. Each participant is free to choose the number of key points. Each key point is recorded with a short click on the *rec button*, but for the last one a long click is used to indicate to the cobot that the trajectory is over. Participants should program the task with "*a good accuracy/speed ratio*".

To analyse the impact of our *Iterative Programming Mode*, we compare it to a classic programming mode used in the industry that we call *One Shot*.

One Shot programming mode allows only one demonstration with gravity compensation assistance.

The *Iterative* programming mode allows programming in two steps:

1. demonstration of translation key points with gravity compensation assistance,
2. demonstration of orientation key points with virtual guiding assistance in translation using the previous demonstration, as explained in Chapter 4.

For the virtual guides assistance used in the *Iterative Programming Mode*, the controller gains were tunned as explain in Chapter 2: "Virtual Guides Definition via Virtual Mechanisms" and set as follows:

$$K = \begin{bmatrix} 10000 & 10000 & 10000 & 3000 & 3000 & 3000 \end{bmatrix} N/m \text{ and } N/m.rad$$

$B = \begin{bmatrix} 150 & 150 & 150 & 30 & 30 & 30 \end{bmatrix}$ $N/m.s^{-1}$ and $N/m.rad.s^{-1}$

$K_s = \begin{bmatrix} 5000 \end{bmatrix}$ N/m and $N/m.rad$

$B_s = \begin{bmatrix} 5 \end{bmatrix}$ $N/m.rad.s^{-1}$

Sampling time was set at $1ms$.

5.3.2 Protocol

We recruited 17 participants from our research laboratory (between 20 and 53 years old, 7 females). Ten participants stated they had prior experience with robots, ranging from robotic courses to hands-on experience with industrial robots. All participants were asked to program the cobot to follow the contour of the wood part, using both Programming Modes – One Shot and Iterative – resulting in two test conditions. The two test conditions were presented in a randomized order to avoid biasing the results towards the last Mode tested. For each condition, the participants were asked to program the cobot 2 times in a row (Repetitions). In total, the participants performed $4 = 2 \times 2$ (Mode $\times$ Repetition) cobot programming tasks.

At the beginning of each condition, the Programming Mode was presented to the participants and the tested case was demonstrated to show the participants how to use the cobot and the programming interface. Then, they were able to familiarize themselves with the system and try the tested case on their own. After each repetition, the result was shown to the participants (i.e., they could feel the virtual guide they created and compare the learned path with the real contour of the part thanks to the haptic feedback given by the cobot. Finally, when a condition was completed, it was asked to the participants to fill the same post-condition survey – Likert-scale survey with a rating from 1 (strong disagreement) to 7 (strong agreement).

5.3.3 Measures

To validate our hypotheses, we performed the following measures:

1. Programming time, to validate H1 – The *Iterative Programming Mode* reduces the programming time of the task.
2. *RMSE* of the angle and *RMSE* of the distance, to validate H2 – The *Iterative Programming Mode* improves the accuracy of the results.
3. Survey results, to validate H2, H3, H4, H5 and H6 – The *Iterative Programming Mode* improves the accuracy of the results, is intuitive and comfortable to use, is perceived as helpful by the users, reduces the user's physical effort and cognitive overload to program the task. *The comanipulation task with the cobot* is not perceived as stressful.

1 - Time: The programming time is measured in seconds and automatically recorded by the cobot for each *Programming Mode* and each *Repetition*. It was measured between the beginning of the programming task (one short click on upper button) and the last point saved (one long click on upper button). For the *Iterative Programming Mode*, the total programming time is the result of the addition of the two iterations.

2 - Accuracy: To measure the accuracy of the programming task, two quantitative variables were taken into account:

- The *RMSE* of the angle
- The *RMSE* of the distance between the participants' and the reference paths

Participants' paths were saved after each programming task execution using Matlab through a RPC communication protocol with the cobot. The interpolation algorithm we used gives us a 0.5 mm spatial resolution.

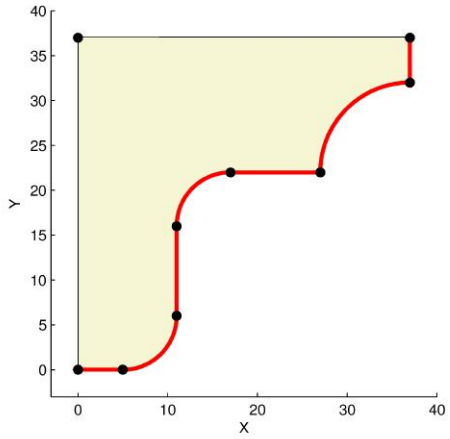


Figure 5.17: Calibration points. The initial wood part measured 37x37x2 cm. It was then cut to shape the contour to follow.

We defined a parametric model of the wood part based on its exact measures. Before the experience, we took 9 calibration points with the tooltip of the cobot as shown in Figure 5.17. Using these 9 recorded points, we computed the affine transformation to fit the cobot reference frame to the Matlab reference frame[5]. We applied this unique transformation to every participants' path for both *Programming Modes* and both *Repetitions*. We can now compare the participants' recorded re-aligned path to its corresponding model portion as shown in Figure 5.18.

[5]This calibration method has the drawback of being highly dependent on the calibration of the tooltip and the mechanic flexibility of the cobot. However, these measures were used to do a comparison between paths obtained under the same conditions and using the same calibration process. Besides we used 9 points for the calibration where 3 points could be enough.

For all interpolation points of the reference path, we defined normal vectors $\vec{n_j}$, tangent vectors $\vec{t_j}$ and the resulting vectors $\vec{s_j}$ of the cross product of the tangent and normal vectors (respectively in green, black and magenta in Figure 5.18), thus resulting in a direct orthonormal basis.

For every interpolation point of a participants' path, we defined the vector $\vec{v_j}$ representing the orientation of the tooltip (in dark yellow).

We define :

- $v_{tj} = \vec{t_j} \cdot \vec{v_j}$, the tangential component,
- $v_{nj} = \vec{n_j} \cdot \vec{v_j}$, the normal component.

2.1 - Angle: The required angle β_j was calculated as: $\beta_j = atan2d(v_{tj}, v_{nj})$. In other words, it corresponds to the angle between the scraper tool and the vertical vector of the cobot frame within the plane perpendicular to the trajectory. For this angle, we calculated the *RMSE* of the angle δ_{RMSE}, between the 45° instruction and β_j for each trajectory.

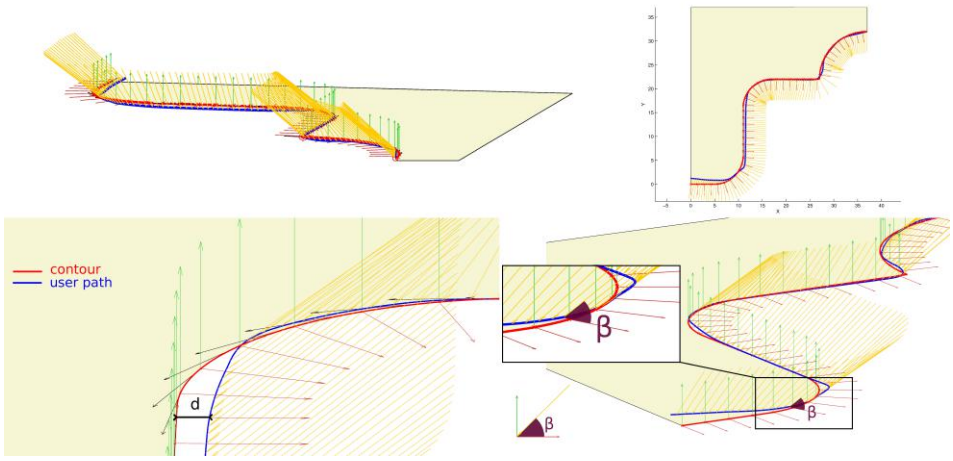


Figure 5.18: Accuracy measures: d – distance between each recorded interpolation point and its closest corresponding point on the reference path – and β – angle between the scraper tool and the vertical vector of the cobot frame within the plane perpendicular to the trajectory. Reference paths are plotted in red and participant's paths are plotted in blue. The yellow arrows represent the tool orientation vectors, green arrows represent the normal vectors to the path, and the red arrows represent the normal vectors to the tangent of the path at each interpolation point. *Top-left figure*: Perspective view of the wood part. *Top-right figure*: Top view (x-y axis) of the representation of the followed path by participants. Distance values are on cm. *Bottom-left figure*: Perspective view. The measure d is shown. *Bottom-right figure*: Perspective view. The measure β is shown.

2.2 - Distance: As previously defined, the participant had also to precisely stick to the contour of the part. Thus, we measured the distance d between each recorded interpolation point and its closest corresponding point on the reference path as shown in Figure 5.18. We computed the *RMSE* of the distance d_{RMSE} for each trajectory.

Finally, we recorded the answers to the two post-condition surveys after each condition was performed by the participant.

5.3.4 Results

We performed independent repeated-measures ANOVA tests for 3 dependent variables:

1. Time
2. *RMSE* of the angle (δ_{RMSE})
3. *RMSE* of the distance (d_{RMSE})

Each ANOVA test was performed on two *within factors* (internal conditions of the experience):

1. *Modes*
2. *Repetitions*

Participants were grouped by four *between factors* (external conditions of the experience):

1. *Age*
2. *Gender*
3. Previous *Experience* with robots
4. Participants' perception of *Performance*

We performed independent repeated-measures ANOVA for each survey question on one *within factor*: the *Mode*. Only for two questions, participants were grouped by the four *between factors* mentioned above. For the other questions this factors had no influence. We set our significance level at $\alpha = 0.05$, which means there is at least 5% chance of having a difference between means of the studied variables. We consider a result strongly significant when p-value < 0.01 and poorly significant when p-value < 0.1. We use poorly significant results to show trends and give further analysis of data, however we do not draw conclusions based on these results. Post hoc pairwise comparisons were computed using non-pooled error terms (i.e., by computing separate paired-samples t tests; sequentially acceptive step-up Benjamini [Benjamini 1995] procedure, with an alpha level of .05.

We use box plots and group means plots to visualize both main effects and interaction effects on the dependent variables. Box plots are useful to represent the median value (less affected by outliers) and data dispersion, while group means plots indicate a more "easy to see" relation/interaction between the mean values of the studied variables. On the group means plot the bars represent a confidence interval of 95%.

All the results and graphs were obtained using R language[6].

5.3.4.1 Time results

Participants were grouped by *Gender*, *Age*, their previous *Experience* with robots and their perception of *Performance* – precisely or both precisely and fast.

We found a significant main effect of the *Programming Modes* on the task execution time of the participants (p-value $< .01$). As we can see in Table 5.9, participants were faster with the *Iterative Programming Mode* than with the *One Shot Programming Mode*. This indicates that the *Iterative Programming Mode* using virtual guides, reduced the programming time and also allowed more "stable" performance since the standard deviation is 1.6 times smaller (see Fig. 5.19). This result validates hypothesis H1 –The *Iterative Programming Mode* reduces the programming time of the task.

Table 5.9: Effect of the *Programming Mode* on the programming time.

Mode	$mean_{Time}$ (s)	σ_{Time} (s)
One Shot	248.08	93.67
Iterative	210.99	58.31

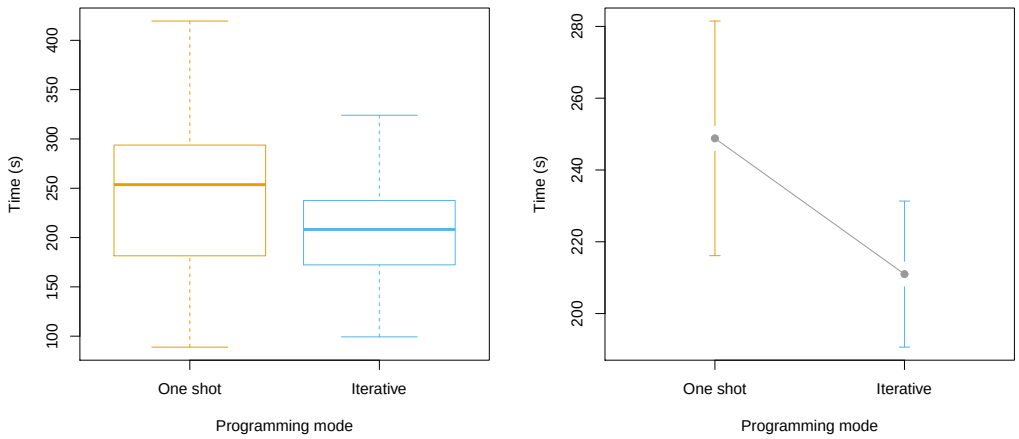
We also found a significant main effect of the *Repetitions* on the programming time of the participants (p-value $< .01$). We can see in Table 5.10 and Figure 5.20, participants were slower during the first repetition. This result confirms there is a training effect through *Repetitions*.

Table 5.10: Effect of the *Repetitions* on the programming time.

Repetitions	$mean_{Time}$ (s)	σ_{Time} (s)
1^{st}	242.89	85.07
2^{nd}	216.90	72.97

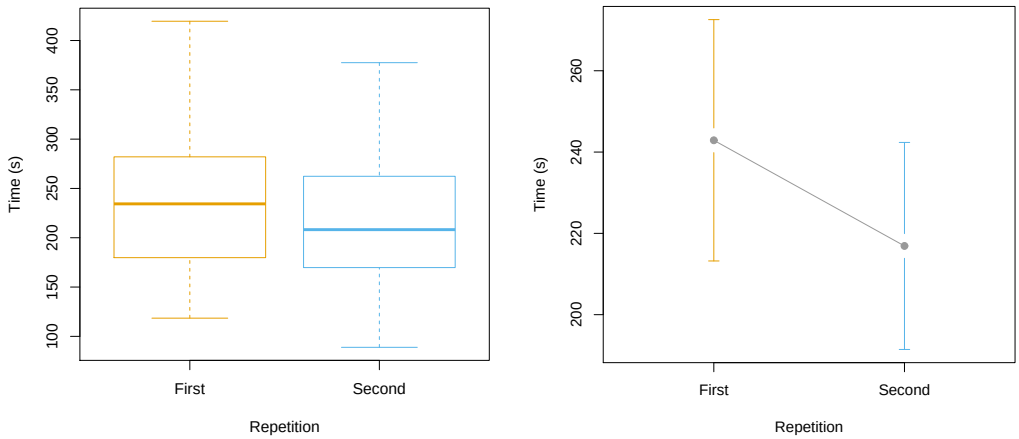
There is no significant effect of interaction between *Repetitions* and the *Programming Mode* (p-value $= 0.1204$). However, we can say from Table 5.11 and Figure 5.21 that participants were faster during both *Repetitions* for the *Iterative Programming Mode*.

[6]https://www.r-project.org/



(a) Box plot (b) Group means with confidence interval of 95%

Figure 5.19: Effect of the *Programming Mode* on the programming time

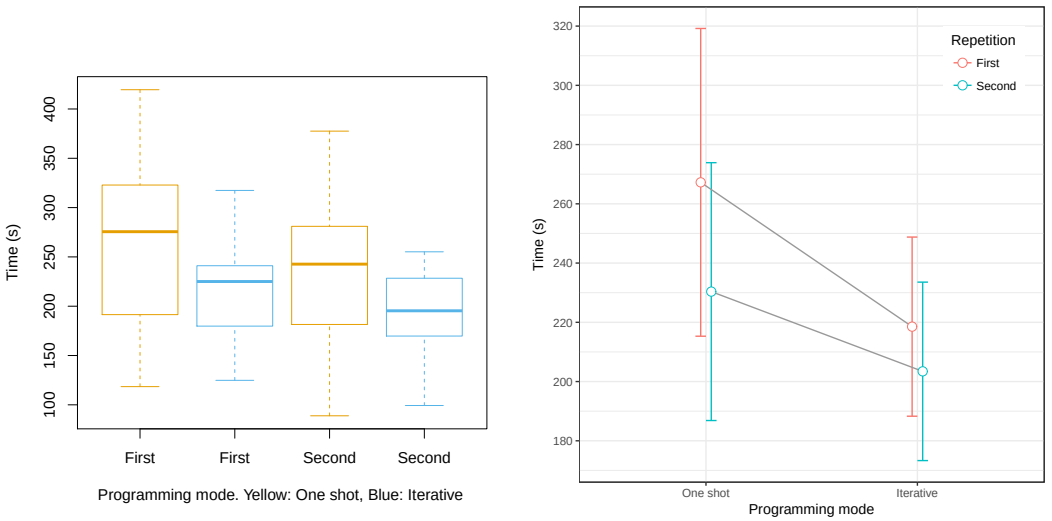


(a) Box plot (b) Group means with confidence interval of 95%

Figure 5.20: Effect of *Repetitions* on the programming time

Table 5.11: Interaction effect between *Repetitions* and *Programming Modes* on the programming time

Mode	**Repetition**	$mean_{Time}$ (s)	σ_{Time} (s)
One shot	1^{st}	267.26	101.03
Iterative	1^{st}	218.54	58.81
One shot	2^{nd}	230.37	84.65
Iterative	2^{nd}	203.44	58.59



(a) Box plot

(b) Group means with confidence interval of 95%

Figure 5.21: Interaction effect between Programming Mode and Repetitions on the programming time

In conclusion, the *Iterative Programming Mode* allowed the participants to execute the task faster during both *Repetitions*, which shows that the general training effect between *Repetitions* did not influence the effect of the *Programming Mode* on the time.

On the other hand, there is a small effect of interaction between the *Programming Mode*, *Repetitions* and *Gender* (p-value = 0.0523) (see Table 5.12 and Fig.5.22) that indicates the training effect and the influence of the *Programming Mode* on the programming time could depend on *Gender*. There is a bigger mean time difference between *Programming Modes* for men during the first *Repetition* and for women during the second *Repetition*. Also, the difference between *Programming Modes* is more constant through *Repetitions* for women. This could indicate that the training effect is more present in male participants. However, to give such a conclusion with higher confidence we would need more participants.

Table 5.12: Interaction effect between the *Programming Mode*, *Repetitions* and *Gender* on the programming time

Mode	**Repetition**	**Gender**	$mean_{Time}$ (s)	$\sigma_{Time}(s)$
One shot	1^{st}	Female	267.12	60.06
Iterative	1^{st}	Female	236.47	53.06
One shot	2^{nd}	Female	245.69	66.44
Iterative	2^{nd}	Female	205.39	55.65
One shot	1^{st}	Male	267.35	125.46
Iterative	1^{st}	Male	205.98	62.02
One shot	2^{nd}	Male	219.64	97.39
Iterative	2^{nd}	Male	202.08	63.50

There is a main effect of the participants' perception of *Performance* on the programming time (p-value = 0.064) (see Fig. 5.23). As shown in Table 5.13, when the participants thought they have done the task precisely they were indeed slower than when they thought they have done the task both precisely and fast. This indicates a relation between time and precision. However, there is no significant correlation between time and quantitative accuracy measures (mean angle error, mean distance).

Table 5.13: Effect of the *Performance* on the programming time

Performance	$mean_{Time}$ (s)	σ_{Time} (s)
Precisely	276.19	106.30
Precisely and fast	215.65	64.43

Finally, *Age* and previous *Experience* with robots did not have a significant influence on the programming time. There was not either any interaction effect between these factors and the *Programming Modes* or *Repetitions*.

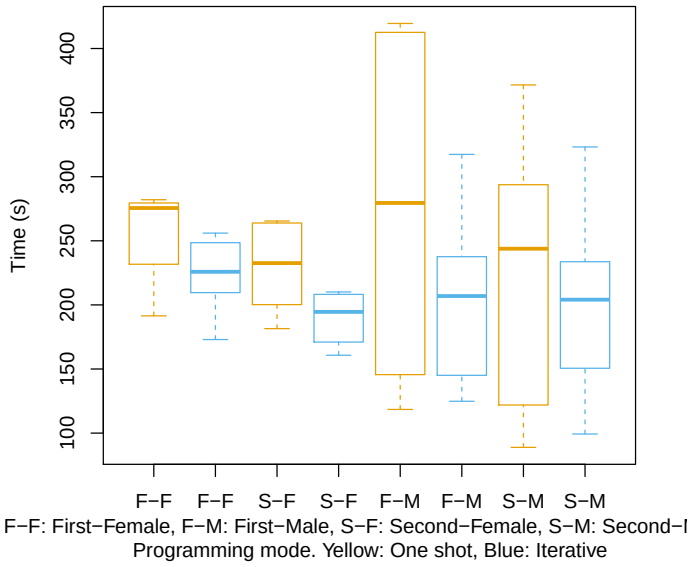
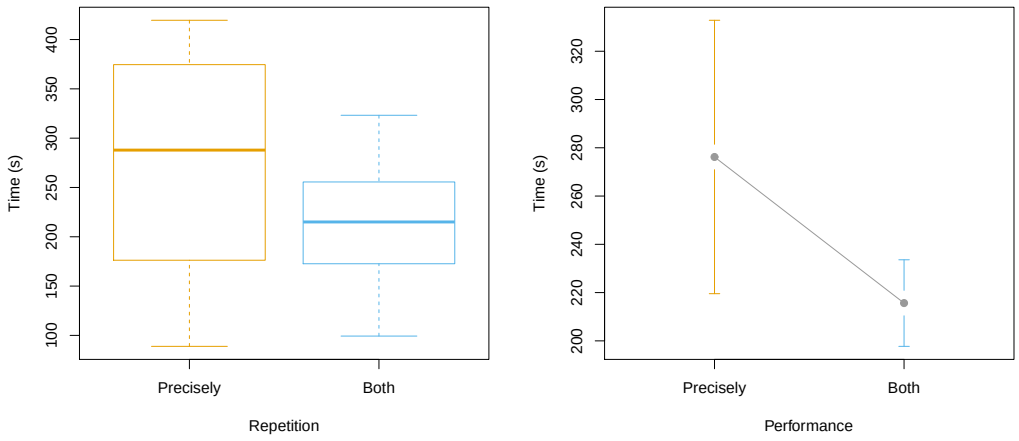


Figure 5.22: Interaction effect between the *Programming Mode*, *Repetitions* and *Gender* on the programming time



(a) Box plot

(b) Group means with confidence interval of 95%

Figure 5.23: Effect of the participants' perception of *Performance* on the programming time

5.3.4.2 Accuracy results

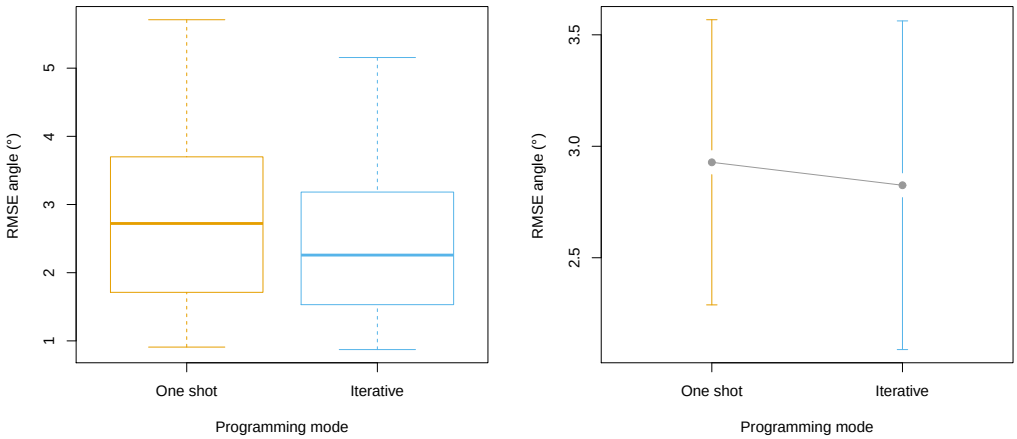
Participants were grouped by *Gender*, *Age*, their previous *Experience* with robots and their perception of *Performance* – precisely or both precisely and fast. Two quantitative variables were analyzed (cf. 5.3.3):

- the *RMSE* of the angle (δ_{RMSE}),
- the *RMSE* of the distance (d_{RMSE}).

There is no significant main effect of *Programming Modes* or *Repetitions* on the *RMSE* of the angle. However, we can see in Table 5.14 that the error was lower when participants used the *Iterative Programming Mode*. This effect is shown in Figure 5.24(a) and 5.24(b).

Table 5.14: Effect of the *Programming Modes* on the *RMSE* of the angle

Programming Mode	$mean_{\delta_{RMSE}}$ (°)	$\sigma_{\delta_{RMSE}}$ (°)
One Shot	2.93	1.83
Iterative	2.82	2.11



(a) Box plot (b) Group means with confidence interval of 95%

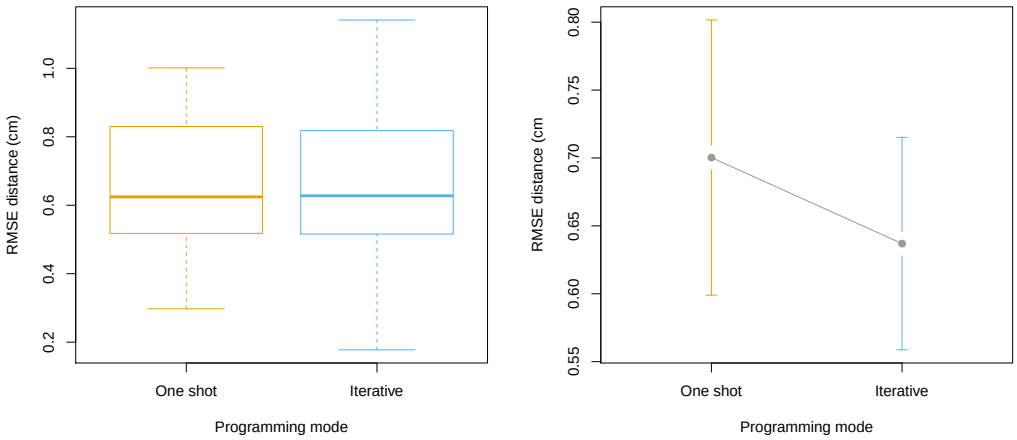
Figure 5.24: Effect of Programming Mode on the *RMSE* of the angle

There is no significant main effect of *Programming Modes* or *Repetitions* on the *RMSE* of the distance. However, Table 5.15 and Figure 5.25 show that the *RMSE* of the distance is smaller when the *Iterative Programming Mode* is used.

There is a significant main effect of *Gender* on the *RMSE* of the distance (p-value = 0.02). From Table 5.16 and Figure 5.26, we can conclude that male

Table 5.15: Effect of the *Programming Modes* on the *RMSE* of the distance

Programming Mode	$mean_{d_{RMSE}}$ (cm)	$\sigma_{d_{RMSE}}$ (cm)
One Shot	0.70	0.29
Iterative	0.64	0.22



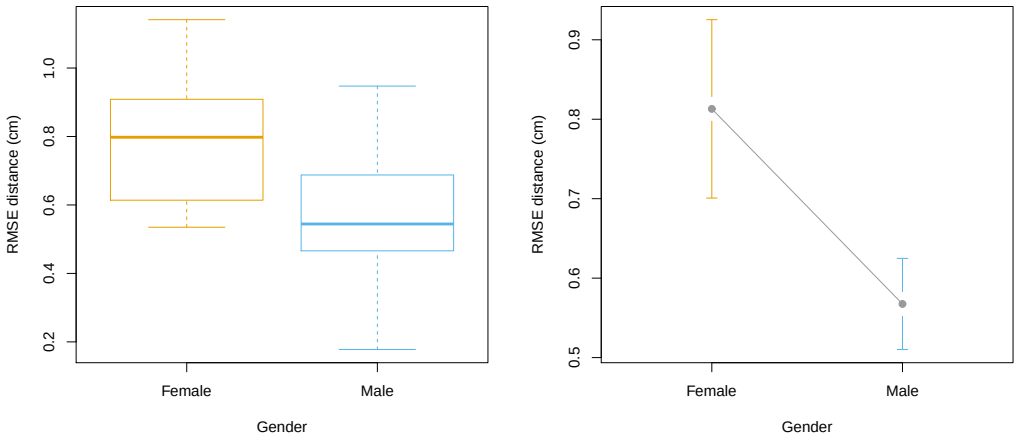
(a) Box plot (b) Group means with confidence interval of 95%

Figure 5.25: Effect of Programming Mode on the *RMSE* of the distance to reference path

participants were more accurate when programming the trajectory on the cobot. However, there is not a significant difference between male and female participants concerning the *RMSE* of the angle, so we cannot conclude that *Gender* has a main effect on the overall accuracy of the task.

Table 5.16: Effect of the *Gender* on the *RMSE* of the distance.

Gender	$mean_{d_{RMSE}}$ (cm)	$\sigma_{d_{RMSE}}$ (cm)
Female	0.81	0.29
Male	0.57	0.19



(a) Box plot (b) Group means with confidence interval of 95%

Figure 5.26: Effect of *Gender* on the mean distance to reference path

Moreover, there is no significant effect of the participants' previous *Experience* with robots on the *RMSE* of the distance or on the *RMSE* of the angle, so we cannot conclude that *Experience* with robots has a main effect on the overall accuracy of the task. Also, we recall that there was not either any influence of the *Experience* on the programming time.

On the other hand, there is no significant correlation between the *RMSE* of the angle and the *RMSE* of the distance. Thus, we cannot conclude that when participants were accurate in translation they were also accurate in orientation. This might be because the level of difficulty of both conditions (translation, orientation) is not the same and so even if a participant had the same level of concentration to perform both, other skills such as physical force prevented him/her to keep the same accuracy level.

In conclusion, there is no significant difference between the *Iterative* and the *One Shot Programming Modes* on the accuracy of the task, so we cannot validate H2 –

The *Iterative Programming Mode* improves the accuracy of the results. Some of the best participants' qualitative results using both *Programming Modes* are shown in Figures 5.27 and 5.28. We can see that it would be difficult to say which method produces better results only by looking at the curves, which can explain the no significant difference found statistically. However, quantitative results go in the direction of H2 and show that participants had better results when they used the *Iterative Programming Mode*.

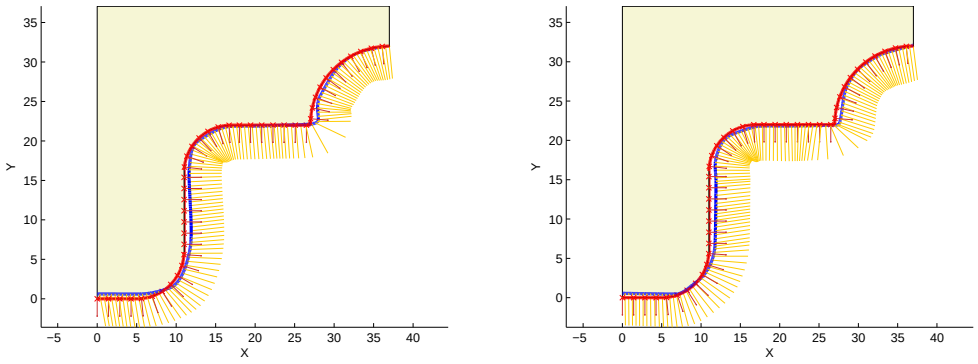


Figure 5.27: Best qualitative results from two different participants using *One Shot Programming Mode*. Top view (x-y axis) of the representation of the followed trajectory by participants. The reference path is plotted in red and participant's paths are plotted in blue. Distance values are on cm. The yellow arrows represent the tool orientation vectors and the red arrows represent the normal vectors to the tangent of the path at each interpolation point.

Age and participants' perception of *Performance* did not have a significant influence on the accuracy variables. There was not either any interaction effect between these factors and the *Programming Modes* or *Repetitions*.

5.3.4.3 Survey results

For clarity reasons, the answers to the 9 questions of our user study survey for each *Programming Mode* are summarized in Table 5.17. For questions 2 and 9, users where grouped by: *Age*, *Gender*, Previous *Experience* with robots and their perception of *Performance* (precisely *versus* both precisely and fast).

With this survey, we observed a significant main effect of the *Programming Mode* on several of the studied questions.

We found a low main effect of the *Programming Mode* on the participants' perception of their task programming performance. We can see in Table 5.17, that the p-value for this question (Q2) is not $< .05$, so we will consider this result as a trend. We can see in Figure 5.29 that, in average, **participants thought**

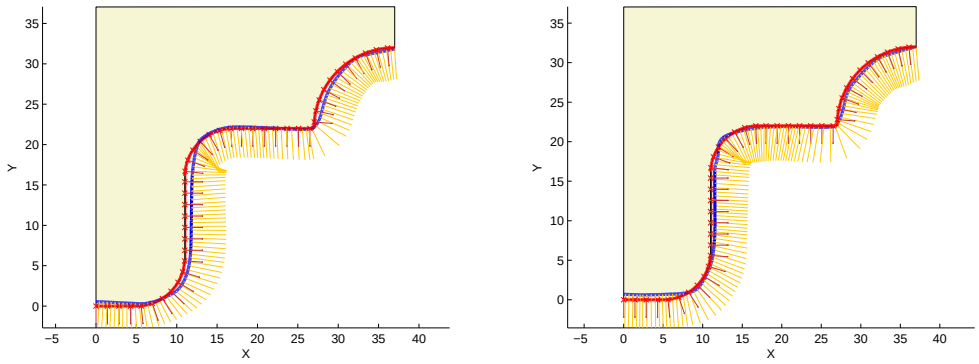


Figure 5.28: Best qualitative results from two different participants using *Iterative Programming Mode*. Top view (x-y axis) of the representation of the followed trajectory by participants. The reference path is plotted in red and participant's paths are plotted in blue. Distance values are on cm. The yellow arrows represent the tool orientation vectors and the red arrows represent the normal vectors to the tangent of the path at each interpolation point.

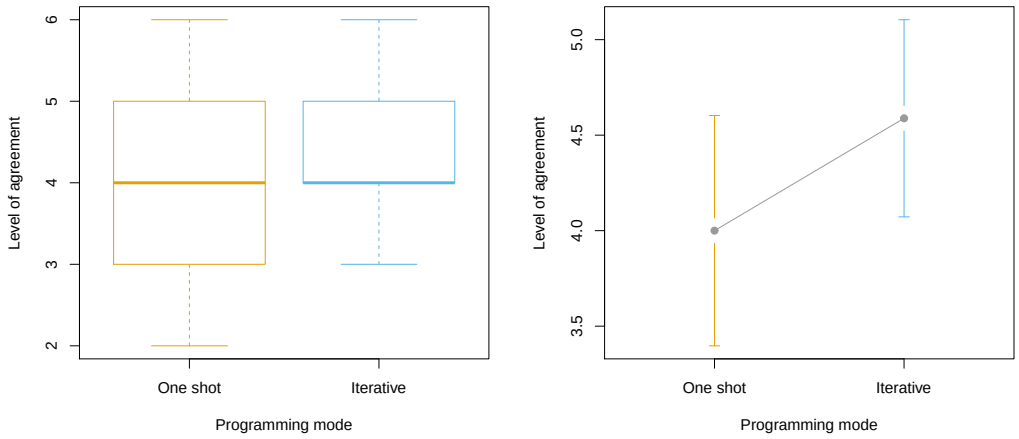
Table 5.17: Results of the user study survey for two *Programming Modes*

	One Shot		Iterative		
Question	**Mean**	σ	**Mean**	σ	**p-value**
Q1: Do you think that you performed well the task?	4.00	1.17	4.58	1	0.065
Q2: Do you think the task was easy to perform?	4.47	1.73	4.94	1.39	0.054
Q3: Do you think the robot was helpful during the task execution?	2.41	1.06	4.35	1.45	<0.0001
Q4: Did you feel comfortable with the robot while performing the task?	2.94	0.83	4.71	1.31	<0.0001
Q5: Do you think the programming interface was intuitive?	4.59	1.37	5.23	0.83	0.052
Q6: Do you think the robot was easy to manipulate?	2.65	1.17	4.12	1.41	<0.0001
Q7: Did you feel you had to put high physical effort to perform the task?	4.35	1.58	3.24	1.03	<0.01
Q8: Did you feel your level of concentration to perform the task was high?	5.88	1.26	3.88	0.93	<0.0001
Q9: Did you feel stressed using the robot while performing the task?	2.53	1.55	1.76	0.90	0.032

they performed better the programming task when using the *Iterative Programming Mode*. This result is coherent with the accuracy results of the previous Subsection, which shows a positive effect of our *Iterative Programming Mode* but not significant difference compared to the *One Shot Programming Mode*.

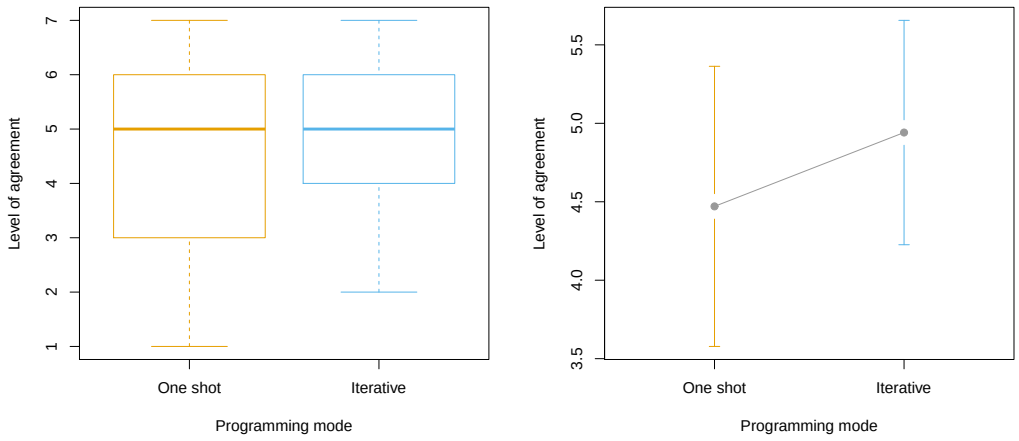
Participants found that **it was easier to program the cobot using the *Iterative Programming Mode***. We can see in Table 5.17, that the p-value for this question (Q1) is not $< .05$, so we will consider this result as a trend that we can also visualize in Figure 5.30. This result supports hypothesis H3 – The *Iterative Programming Mode* is intuitive and comfortable to use.

We also found a low main effect of the *Programming Mode* on the participants' perception of the intuitiveness of the programming interface. We can see in Table 5.17, that the p-value for this question (Q2) is not $< .05$, so we will consider this result as a trend. We can see in Figure 5.34 that, in average, **participants found the programming interface more intuitive while using the *Iterative Pro-***



(a) Box plot (b) Group means with confidence interval of 95%

Figure 5.29: Effect of *Programming Modes* on the participants' perception of their task programming performance

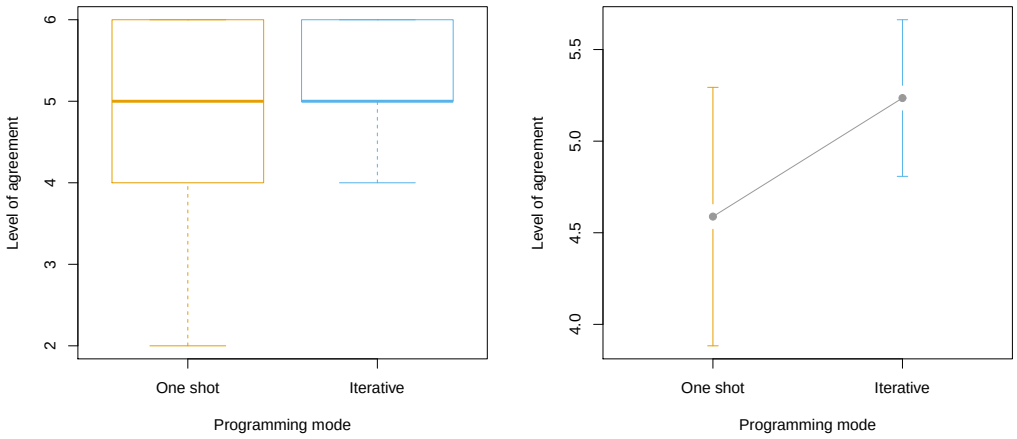


(a) Box plot (b) Group means with confidence interval of 95%

Figure 5.30: Effect of *Programming Modes* on the difficulty of the task

gramming Mode. This result supports hypothesis H3 – The *Iterative Programming Mode* is **intuitive** and comfortable to use.

Participants felt more **comfortable while programming the cobot with the *Iterative Programming Mode***. This significant result is shown in Table 5.17 and Figure 5.32 and supports hypothesis H3 – The *Iterative Programming Mode* is



(a) Box plot (b) Group means with confidence interval of 95%

Figure 5.31: Effect of *Programming Modes* on the intuitiveness of the programming interface

intuitive and **comfortable** to use.

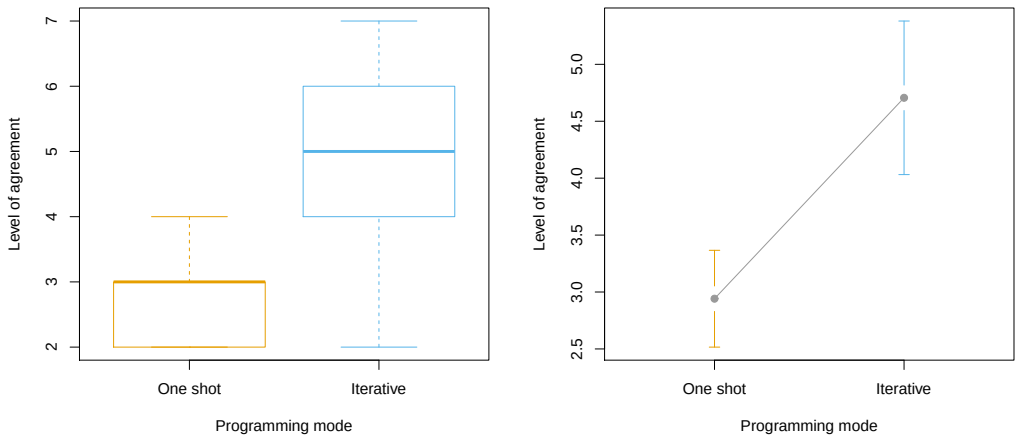
Participants found that **the cobot was more helpful when they used the *Iterative Programming Mode***. This significant result is shown in Table 5.17 and Figure 5.33 and validates hypothesis H4 – The *Iterative Programming Mode* is perceived as helpful by the users.

Participants found that **the cobot was easier to manipulate when they used the *Iterative Programming Mode***. This significant result is shown in Table 5.17 and Figure 5.34 and supports hypothesis H4 and H5 – The *Iterative Programming Mode* is perceived as **helpful** by the users and it reduces the user's **physical effort** and cognitive overload to program the task.

Participants found that **they had to used less physical effort and concentration to program the cobot with the *Iterative Programming Mode***. This significant results are shown in Table 5.17 and Figures 5.35 and 5.36. They validate hypothesis H5 – The *Iterative Programming Mode* reduces the user's **physical effort** and cognitive overload to program the task.

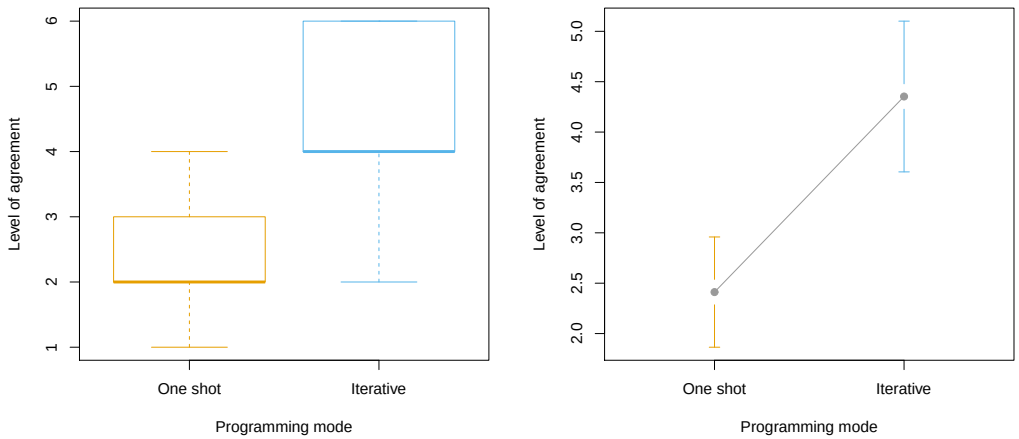
Participants felt less **stressed while programming the cobot with the *Iterative Programming Mode***. This significant result is shown in Table 5.17 and Figure 5.37. We can also see that, in average, participants did not feel stressed while using both *Programming Modes*, which validates hypothesis H6 – The comanipulation task with the cobot is not perceived as stressful.

Moreover, we found an interaction effect between the *Programming Mode* and the participants' previous *Experience* with robots on the task difficulty - Question 2 - (p-value = 0.0076). As we can see in Figure 5.38, participants who had not worked



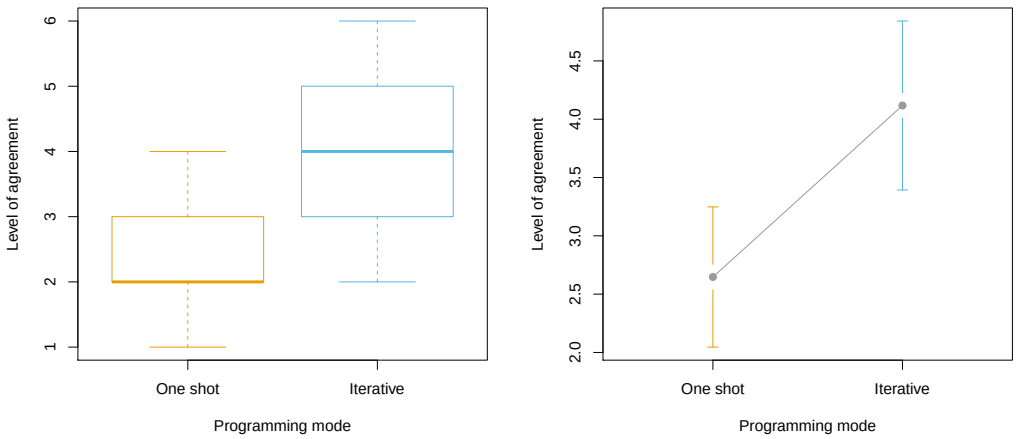
(a) Box plot (b) Group means with confidence interval of 95%

Figure 5.32: Effect of *Programming Modes* on the comfortability



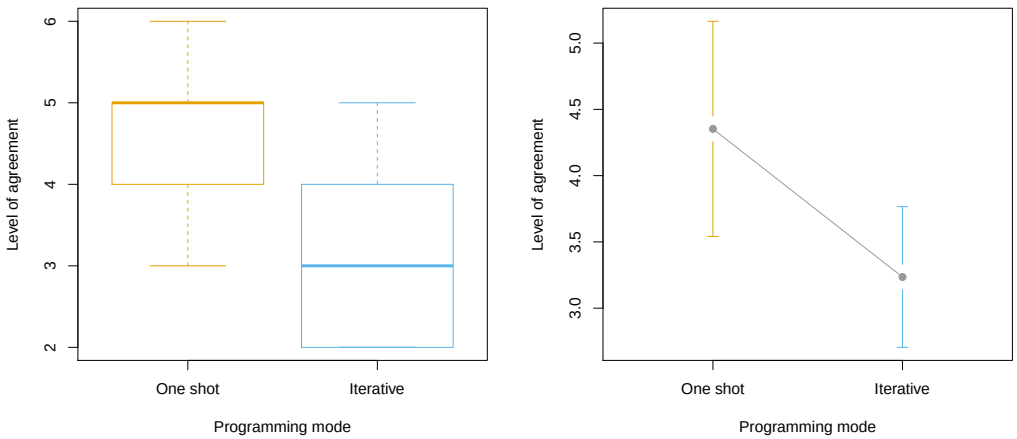
(a) Box plot (b) Group means with confidence interval of 95%

Figure 5.33: Effect of *Programming Modes* on the perception of the robot helpfulness



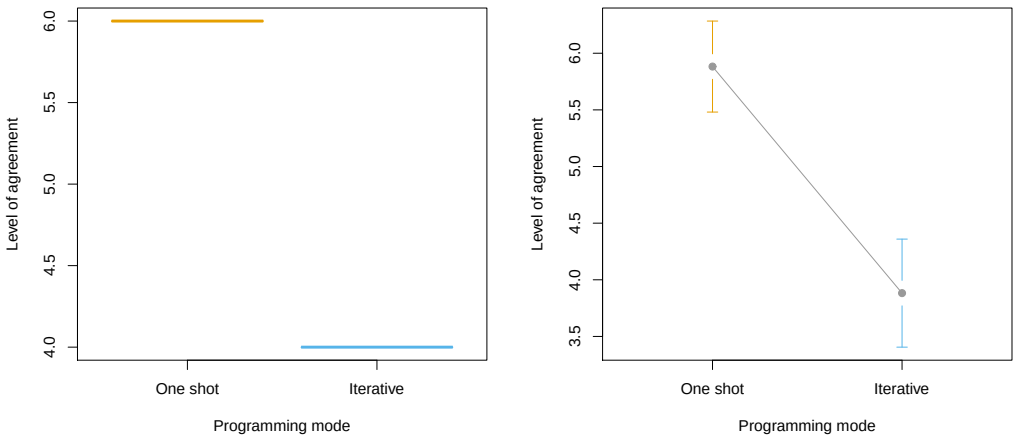
(a) Box plot (b) Group means with confidence interval of 95%

Figure 5.34: Effect of *Programming Modes* on the difficulty for manipulating the robot



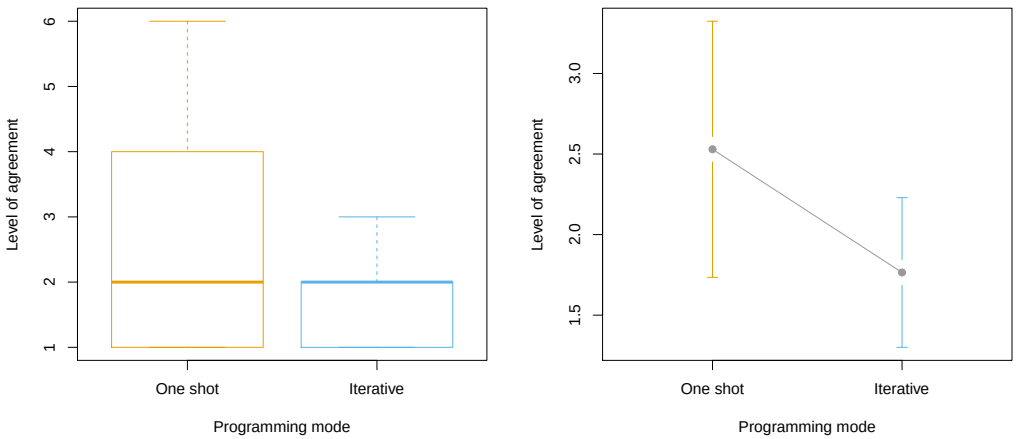
(a) Box plot (b) Group means with confidence interval of 95%

Figure 5.35: Effect of *Programming Modes* on the participants' physical effort



(a) Box plot (b) Group means with confidence interval of 95%

Figure 5.36: Effect of *Programming Modes* on the participants' cognitive load



(a) Box plot (b) Group means with confidence interval of 95%

Figure 5.37: Effect of *Programming Modes* on the participants' stress

with a robot before thought the task was easier to perform when they used the *Iterative Programming Mode*. For experimented participants, the difference between the *Programming Modes* was smaller. Post hoc tests showed no significant interaction effect between each *Mode* and *Experience* combinations. Thus, no conclusions can be done about the influence of *Experience* on the *Programming Mode* effect regarding the difficulty of the task.

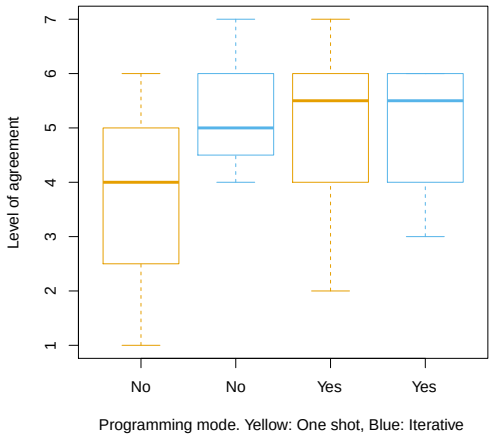


Figure 5.38: Interaction effect between the *Programming Mode* and *Experience* with robots on the difficulty of the task

There is also an interaction effect between the *Programming Mode* and the participants' previous *Experience* with robots on the participants' stress during the task - Question 9 - (p-value $= 0.0680$). We can see in Figure 5.39 that participants who had not worked with a robot before were much less stressed when using the *Iterative Programming Mode*. For experimented participants the difference between both *Modes* was smaller but they still felt less stressed using the *Iterative Programming Mode*. Post hoc tests showed no significant interaction effect between each *Mode* and *Experience* combinations. Thus, no conclusions can be done about the influence of *Experience* on the *Programming Mode* effect regarding the participants' stress. However, we recall that, in average, the level of stress was low for all participants, which is why we were able to validate hypothesis H6.

Comanipulation Participants were free to manipulate the cobot the way they felt more comfortable. In general, they used the tool handler and helped themselves by grabbing the 4^{th} axis of the cobot as shown in Figure 5.40. This could indicate that the cobot design was not totally apt for the task or that it was not transparent enough so the users found it difficult to place the tooltip exactly were they wanted to.

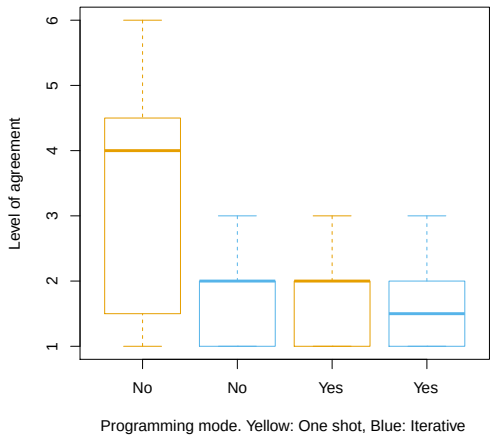
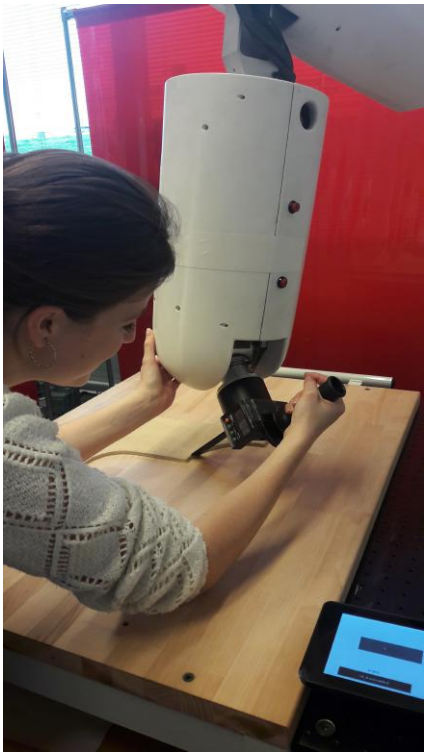


Figure 5.39: Interaction effect between the *Programming Mode* and *Experience* with robots on the participants' stress

(a) Manipulation by the handle of the tool



(b) Manipulation by the handle of the tool and the 4^{th} body of the arm

Figure 5.40: Regular comanipulation postures during the experiment

5.3.5 Conclusion of experiment 2

Using the previous results, we can make the following conclusions about the initial hypotheses of the experiment:

H1 The main effect of the *Programming Mode* on the time execution of the task validates the hypothesis H1 – *Iterative Programming Mode* reduces the execution time of the task.

H2 Several results supported the hypothesis H2 – *Iterative Programming Mode* improves the accuracy of the results :

- The *RMSE* of the angle was smaller when participants used the *Iterative Programming Mode*
- The *RMSE* of the distance was smaller when participants used the *Iterative Programming Mode*
- Participants found that they performed the task better when using the *Iterative Programming Mode*.

However, the tests ANOVA did not show a significant difference between both *Programming Modes* for both accuracy variables and for Question Q1 – Do you think you performed well the task. Thus, we cannot validate H2. This lack of significance can be explained by the number of participants on the experiment. We believe that our results show a trend and that a user study with more participants could lead to the validation of this hypothesis.

H3 Several results supported the hypothesis H3 – The *Iterative Programming Mode* is intuitive and comfortable to use – and though we validate it:

- Participants found that it was easier to program the robot when using the *Iterative Programming Mode*.
- Participants found that it was more comfortable to program the robot with the *Iterative Programming Mode*.
- Participants found that it was more intuitive to program the robot with the *Iterative Programming Mode*.

H4 The main effect of the *Programming Mode* on the survey questions Q3 and Q6 (Do you think the robot was helpful during the task execution? - Do you think the robot was easy to manipulate?) validates the hypothesis H4 – The *Iterative Programming Mode* is perceived as helpful by the users.

H5 The main effect of the *Programming Mode* on the survey questions Q6, Q7 and Q8, validates the hypothesis H5 – The *Iterative Programming Mode* reduces the user's physical effort and cognitive overload to program the task.:

- Q6: Do you think the robot was easy to manipulate?
- Q7: Did you feel you had to put high physical effort to perform the task?
- Q8: Did you feel your level of concentration to perform the task was high?

H6 For both *Programming Modes* the mean score on the survey question Q9 (Did you feel stressed using the robot while performing the task?) was low. This result validates the hypothesis H6 – The comanipulation task with the cobot is not perceived as stressful. Moreover, it was also shown that participants felt less stressed when using the *Iterative Programming Mode*.

With this experiment we confirmed the advantages of using 6D Virtual Guides for comanipulation tasks. We also showed the positive impact of our *Iterative Programming* approach – divide the programming process in two phases: first positions and then orientations. Participants found our method easier to use, more comfortable and more intuitive than the *One Shot* method (that uses Gravity Compensation). Participants also found that our approach reduces physical effort and cognitive overload. Finally, the comanipulation task with the cobot did not introduce any stress on participants.

5.3.6 Additional Remarks and Improvements

During the experience we noticed participants did not have the same posture while using both *Programming Modes*. Awkward and uncomfortable postures were mostly taken when participants used the *One Shot Programming Mode* as shown in Figure 5.41. More ergonomic postures were taken when they used the *Iterative Programming Mode* 5.42. An improvement of the experience will include measures of participants' forces in different body parts to analyse ergonomics and take the results into account to improve our algorithm, as suggested in [Maurice 2017].

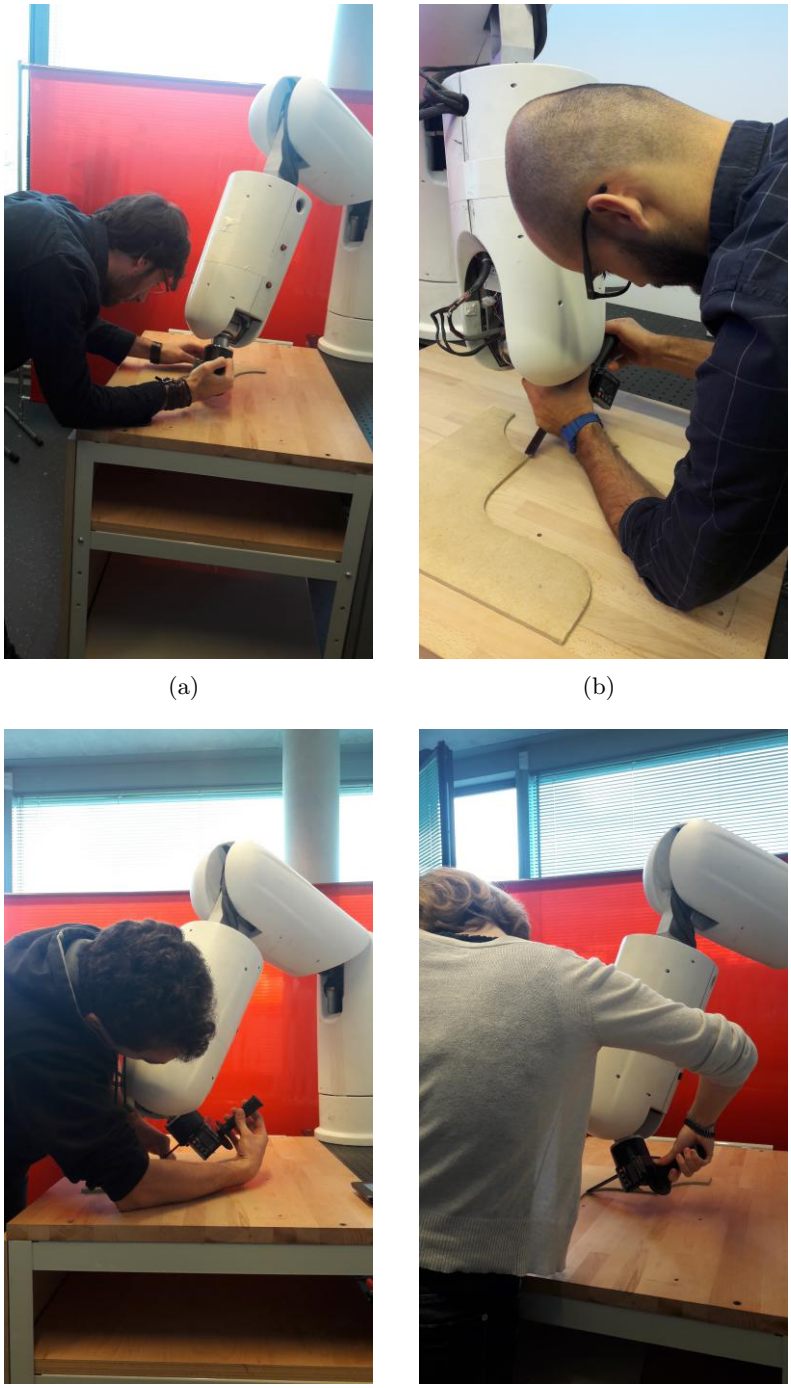
At the end of each survey we asked the participants to give extra feedback about the experiences and make some comments.

Some user's comments when using the *One Shot Programming Mode*:

- "It is very hard to manipulate the robot precisely."
- "It's difficult to keep the right orientation of the tool."
- "I cannot do what I want with the robot."

Some user's comments when using the *Iterative Programming Mode*:

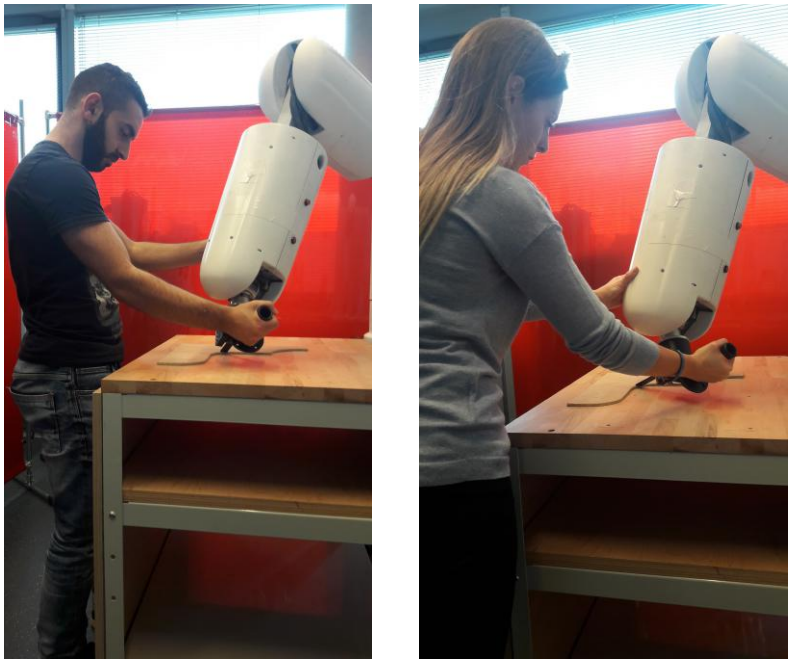
- "The programming process is really fast. To be guided in translation allows to better deal with the rotations."



(a) (b)

(c) (d)

Figure 5.41: Postures when using *One Shot Programming Mode*



(a) (b)

(c) (d)

Figure 5.42: Postures when using *Iterative Programming Mode*

- "Rather natural and easy to use. Programming is better with the iterative mode."
- "It was much better like that. A problem to solve at once!"

Some user's general comments:

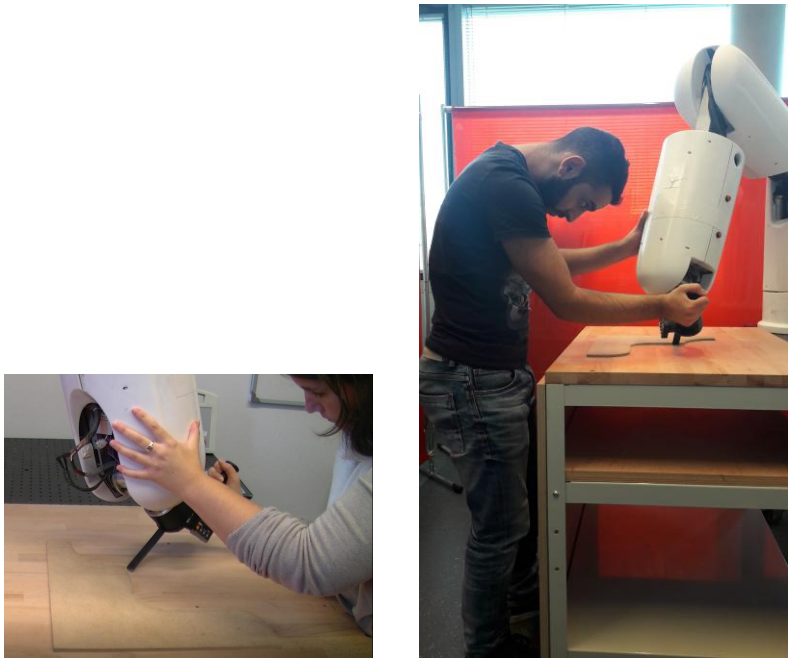
- "At the end it is more difficult to perform the task, depending on the angle of the last axis of the robot"
- "Translation is easier than rotation"
- "Trajectories are quite easy to teach, the interface is simple, but the robot itself is quite a pain to manipulate on the whole working area".

Between the improvements mentioned there was the need to have a recording (interaction) button on the handler and not on the 4^{th} axis so it could be more comfortable to program the cobot. Figure 5.43 shows two cases where a button on the tool handler would be more useful.

(a) (b)

Figure 5.43: Recording button should be placed on the handler

Another improvement would be to add two handles on the 4^{th} axis since it would be very useful to help manipulate the cobot. This is because in this particular version of the 6-DOF ISybot, axis 5 and 6 are too hard to manipulate. This could be taken into account by ISybot on the design of their future cobots. Figure 5.44 shows two cases where a handle on the 4^{th} axis would be useful.



(a) (b)

Figure 5.44: Another handling system should be placed on axis 4

We also noticed the height of the table was not adaptable to each person, so for taller people it was harder to find a comfortable posture for both *Programming Modes*.

5.4 Conclusion

Both experiments showed the interest of using Virtual Guides Assistance in comanipulation tasks. Experiment 2 showed also that these Guides are useful to assist the human operator during the iterative programming of a cobot. We demonstrated that our programming framework is intuitive, comfortable and helpful for users. We also showed that our approach has an influence in reducing the physical effort and cognitive overload of the human operator while accelerating the programming process. It is important to notice that these advantages do not depend on the user's previous experience with robots, gender or age, which make it suitable to be used by human operators that have high knowledge of the task and the gestures to execute it, but have little or no experience with robots. Finally, we have reasons to believe that our framework could allow the improvement of the user's ergonomy, which is a major issue in collaborative robotics applications.

Chapter 6

Conclusion

> *C'est le commencement qui est le pire, puis le milieu puis la fin ; à la fin, c'est la fin qui est le pire.*
>
> Samuel Beckett, L'Innommable, 1953

The main contributions presented in this thesis are summarized hereafter. Then, perspectives for improvements and extension of the developed Virtual Guides programming framework are proposed.

6.1 Contributions

We presented in Chapter 1: "Introduction" the problematics of the context of this research and the current limitations of the literature in both our areas of interest: Virtual Guides Assistance and Programming by Demonstration (*PbD*). Therefore, the work presented in this thesis focuses on the **development of new tools to enhance the flexibility of the use and programming of Virtual Guides Assistance in a human-robot comanipulation context**. The proposed approach relies on *PbD*, and particularly on *kinesthetic teaching*. Indeed, **the use of this method allows non-robotics experts to intuitively program robots**, because users can program a task by directly manipulating the end-effector of the robot.

In Chapter 2: "Virtual Guides Definition via Virtual Mechanisms" and Chapter 3: "Virtual Guides Construction", **we presented an extension of virtual mechanisms approach to 6D Virtual Guides for comanipulation applications**. Virtual Guides work as an impedance controller for the robot, allowing movement along the preferred directions and prohibiting movements along the restricted ones. In Chapter 2: "Virtual Guides Definition via Virtual Mechanisms", the passivity of the system is proven by studying the dissipated energy of the system which proves

also its stability. We also presented the problem of singularities of the kinematic model of Virtual Guides when they are defined through users demonstrations (using kinesthetic teaching). **We proposed a solution using Jacobian normalization to overcome this issue.**

In Chapter 3: "Virtual Guides Construction", **we proposed how the geometric and kinematic models of Virtual Guides can be programmed through kinesthetic teaching and modeled through interpolation functions**. To construct Position Constraints we implemented multi-dimensional Akima Spline interpolations. To construct Orientation Constraints in $\mathbb{SO}(3)$ we used *SQUAD* interpolations. In both cases we proposed to separate the time and space components of the curves to parameterize them in a way that guarantees the Jacobian normality. **We also proposed a 6D Virtual Guides definition through *XSplines***, based on both position interpolations – $MDSpline \in \mathbb{R}^3$ – and orientation interpolations – $Squad \in \mathbb{SO}(3)$ – of poses obtained through kinesthetic teaching. The advantage of *XSplines* is that they are parameterized in a way that allows the **synchronization of the translation and orientation movements**.

Chapter 4: "Iterative Virtual Guides Programming" **presented a new intuitive, iterative and assisted framework for programming 6D Virtual Guides**. We explained how Virtual Guides can be locally refined or modified by the user in both Cartesian position and orientation. During this iterative process, the user benefits from the Virtual Guides Assistance. Also, three interaction modes with the cobot are presented: *Soft*, *Hard* and *Null* Virtual Guides. Finally, we presented an iterative approach to program 6D Virtual Guides in two phases. **This method aims at reducing the cognitive load of the user and enhance the programming experience by separating the translation and the rotation programming**. The allocation of roles on the human-cobot interaction were also defined: the human operator masters the action plan and the cobot assists the human passively and under explicit demand.

Two experiments showing the advantages of using Virtual Guides Assistance and our programming framework are presented in Chapter 5: "Experimental Evaluation". Both experiments showed the interest of using Virtual Guides Assistance in comanipulation tasks. Experiment 2 showed also that these Guides are useful to assist the human operator during the iterative programming of a cobot. **We demonstrated that our programming framework is intuitive, comfortable and helpful for users. We also showed that our approach has an influence in reducing the physical effort and cognitive overload of the human operator while accelerating the programming process**. It is important to notice that these advantages do not depend on the user's previous experience with robots, gender or age, which make it suitable to be used by human operators that have high knowledge of the task and the gestures to execute it, but have little or no experience with robots. Finally, we have reasons to believe that our framework could allow the improvement of the user's ergonomy, which is a major issue in collaborative robotics applications.

Beyond the specific application of the Virtual Guides programming framework on industrial tasks, **this approach could be used and evaluated in other areas** such as robotic assisted surgery or rehabilitation. **This thesis provides some tools to construct, enforce and modify 6D Virtual Guides** that can be applied to the iterative programming framework but also used separately in other task programming or execution scenarios.

6.2 Perspectives

In the future, some interesting points should be investigated to improve and extend the 6D Virtual Guides programming framework proposed in this doctoral work.

6.2.1 Virtual Guides Construction

We presented in Chapter 3: "Virtual Guides Construction" an approach for constructing 6D Virtual Guides based on interpolation functions. It could be interesting to explore the use of other interpolations and the differences with the choice made in this thesis. Also, as mentioned in Chapter 4: "Iterative Virtual Guides Programming", simple guides such as straight lines, arcs or circles could be used to initialize the programming process. The choice of an interpolation function is usually made according to the application. The advantages of each method could be taken into account in order to use several interpolations rather than just the one which gives the best global compromise. For example, monotonic cubic interpolations are more suitable for generating straight lines and corners, and natural cubic splines should be used to generate circle paths. Quintic interpolation [Erkorkmaz 2005] could be also of interest in further research about Virtual Guides construction.

6.2.2 Virtual Guides Programming Framework

As presented in Chapter 4: "Iterative Virtual Guides Programming", the Virtual Guides iterative programming approach can be also applied for:

- Programming the force the robot should apply during a particular task execution: to this aim, a force sensor attached to the tool handle would be needed in order to record the forces during the kinesthetic teaching. Teaching forces by demonstration has been addressed in [Rozo 2014] for human-robot cooperative transportation tasks, this could be a good base to continue the research on this topic.

- Programming the velocity to reproduce a task automatically: in human-robot comanipulation, there are some parts during the execution of a task where the robot can be used in autonomous mode. During the Virtual Guides construction by kinesthetic teaching, the time of the demonstration is encoded. Then the guide is parameterized on space and a transformation function (using a monotonic interpolation) from the space parameter to time is done. Using

this function we can always access the parameterization time of the curve and then encode the velocity of the demonstration. This could also be used within the iterative approach, by first programming the trajectory, activating the 6D Virtual Guide, and finally demonstrating to the robot the desired velocity to reproduce the trajectory automatically.

- Programming the stiffness of the Virtual Guide: depending on the application, it could be useful to modify the stiffness of the Virtual Guide online, by modifying the gain K corresponding to the spring of the virtual mechanism. To this aim, we could use again kinesthetic teaching to show to the cobot the zones where the stiffness should vary. An idea to explore could be to perform wide going and coming movements to program low stiffness, and counterwise, small ones to program high stiffness. Again, this could be integrated to the iterative programming framework as a second or third phase of the process. With the 6D Virtual Guide activated in *Soft interaction mode*, the user is able to go out the guide and then go back. With this feature, he/she is able to demonstrate the desired stiffness as explained above.

Finally, it would be interesting to integrate the developments presented in our previous work Appendix A: "Co-manipulation with a Library of Virtual Guiding Fixtures" to the ones presented in this thesis, in order to extend the Virtual Guides iterative programming framework using a library of Virtual Guides, where multiple guides can be activated at the same time. The library of guides was conceived using a probabilistic approach through *GMM* and *GMR*, though the challenge resides on merging both Virtual Guides construction approaches in an optimal manner.

6.2.3 Interaction - Interface

6.2.3.1 Virtual Guides Visualization

One problem raised by some participants during the experiments presented in Chapter 5: "Experimental Evaluation" and particularly on the experiment presented in our previous work [Raiola 2017a], is the absence of a visualization interface for the Virtual Guides (voir Appendix A: "Co-manipulation with a Library of Virtual Guiding Fixtures"). This makes it harder for the user to find where the guides are placed in the robot's workspace, and the difficulty increases with the number of *DOF* of the path to follow. For the Pick-and-Place task in [Raiola 2017a], Virtual Guides were on $\mathbb{R}^3$ and several guides were active at the same time (see Figure 6.1). After programming the guides, the user did not remember exactly how he performed the movement on space or where the starting point was. It was even more difficult to remember the emplacement of the multiple guides.

In Experiment 2 – Section 5.3: "Experiment 2: Iterative Programming", Virtual guides were on $\mathbb{R}^3$ and on $\mathbb{SO}(3)$, so the user needed to remember not only the position of the tool but also the orientation. Furthermore, we proposed in Chapter 4: "Iterative Virtual Guides Programming" an iterative programming framework were

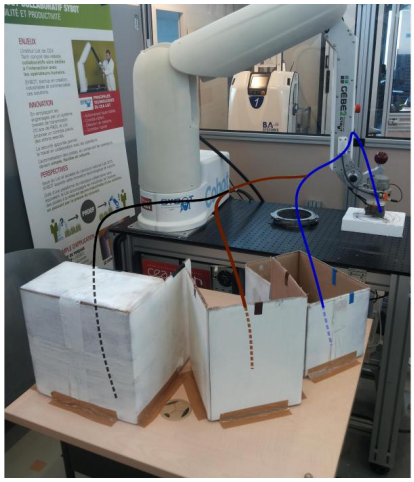


Figure 6.1: Multiple Virtual Guides created in the robot workspace for the Pick-and-Place Experiment presented in [Raiola 2017a]. The task consisted in taking 6 discs from the robot's workstation and insert them inside specific boxes identified with 3 different colors: blue, brown and black. For each box there were two discs with a piece of tape of the same color.

6D Virtual Guides can be refined and modified. An enhancement of our framework would be to add visualization assistance in order to make it more intuitive to the user to program the guides or make modifications. During this thesis, we developed a prototype for Virtual Guides visualization using an *AR* interface, however no measures or user studies were performed. This feature could also be useful to allow a new user (who did not program the task initially) to visualize what has been done on the workspace and go on with the task. Which will be also important in the context of flexible manufacturing introduced in Chapter 1: "Introduction", where robots should be collaborative and intuitive in order to add flexibility to the production system.

The visualization problem affects the interaction with the Virtual Guides in both *Soft* and *Hard* interaction modes. During the *Soft* interaction mode, the user can "escape" the guides, but since he/she can not see them, it is difficult to join them back again. This could also affect the iterative modification of the Virtual Guides. During the *Hard* interaction mode, the absence of visual feedback causes the user to involuntarily move against the guided directions generating a corrective force exerted by the spring-damper system which could cause the user to exert unnecessary efforts. In fact, when interacting with the Virtual Guides the user only benefits from haptic feedback.

The idea of visualizing Virtual Guides is not new in literature. The original definition of *virtual fixtures* proposed also a visualization interface in order to fully exploit their advantages [Rosenberg 1993]. Virtual Fixtures visualization was used by Rosenberg to enhance operator performance in the telerobotic control of Fitt's Law peg-board task (see Figure 6.2). However, the technology at this time made the

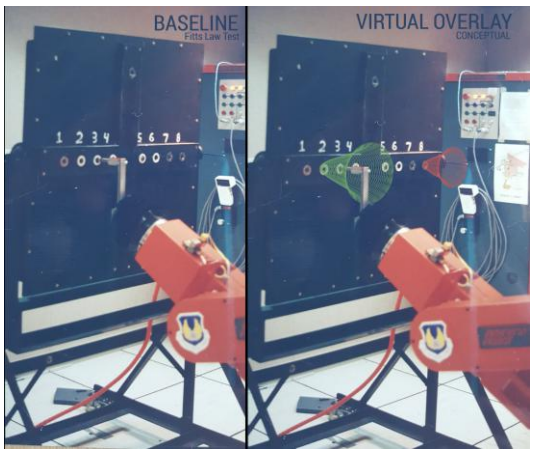


Figure 6.2: Augmented reality application on telerobotic control of Fitt's Law peg-board task [Rosenberg 1993]. Image by GardenM - Own work, CC BY-SA 4.0.

interface cumbersome and neither easy nor cost-effective to implement in industrial environments.

(a) *AR* interface proposed by [Rosenberg 1993], used in teleoperation tasks.

(b) Microsoft Hololens *AR* headset, used by our team for the comanipulation of an ISybot cobot.

Figure 6.3: Virtual Guides visualization interfaces.

Today, recent commercial solutions, like the Microsoft Hololens, allow to integrate

an *AR* interface to robotics applications (see Figure 6.3). We actually used the Hololens headset to create an *AR* interface in order to allow Virtual Guides visualization, as shown in Figure 6.4. As stated above, we did not run any experimental validation of this implementation but we have reasons to believe that this interface could enhance the Virtual Guides programming process. Future work includes the integration of our refinement and modification tools to the *AR* interface so the user is able to, for example, modify a point of the guide by manipulating it via the *AR* interface. We could imagine the user "drawing" Virtual Guides directly on the working space.

Some drawbacks of our implementation were related to the calibration process. For example, sometimes the visualized Virtual Guide did not match exactly the real contour of the object, and the rendering changed with the perspective of the point of view. Further analysis and developments are needed to overcome this problem. Also, the first version of the Hololens headset presented a limited field of view which disturbs the user interaction with the Virtual Guides. A solution to this problem should be proposed soon by Microsoft.

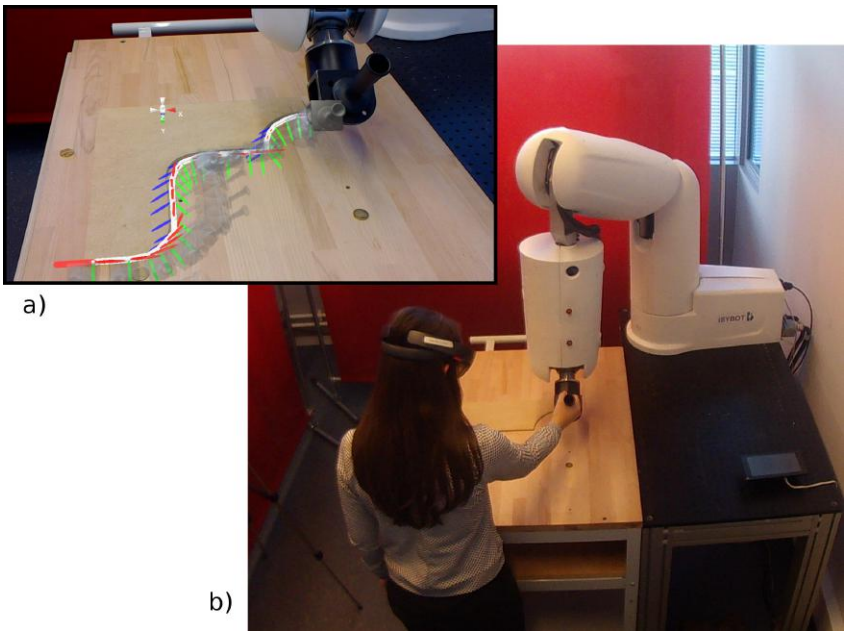


Figure 6.4: Virtual Guides visualization through an *AR* interface using an Hololens headset. Tested on the Experiment 2 (5.3) presented in Chapter 5: "Experimental Evaluation". *a)* Visualization of 6D Virtual Guides using an AR interface. 6D Virtual Guides are represented by the pose of the tool attached to the cobot. The translational component of the guide is represented by cartesian points (white points) and the rotational component of the guide is represented by trihedrons along the path (red-green-blue). We can also see the different orientations of the model of the tool through the path (transparent gray). *b)* User wearing the Hololens headset to visualize the Virtual Guides (shown in Figure *a)*

6.2.3.2 Ergonomy

The main objective of our programming framework is to assist the human operator. In Chapter 5: "Experimental Evaluation", we confirmed that the use of our proposed Virtual Guides Iterative programming approach reduced the physical effort and cognitive overload of the user. However, this was just validated by a user survey. It would be useful to add a force sensor on the cobot handle in order to measure the forces exerted by the user with and without the iterative programming framework. Also, a deep analysis of the user's postures and gestures while manipulating the cobot in real case scenarios, would allow to better understand and quantify the impact of the tools presented in this thesis and would also give some leads to further developments. An excellent base to continue this research is the work realized in [Maurice 2017], where a generic method for performing detailed ergonomic assessments of comanipulation activities is presented (see Figure 6.5).

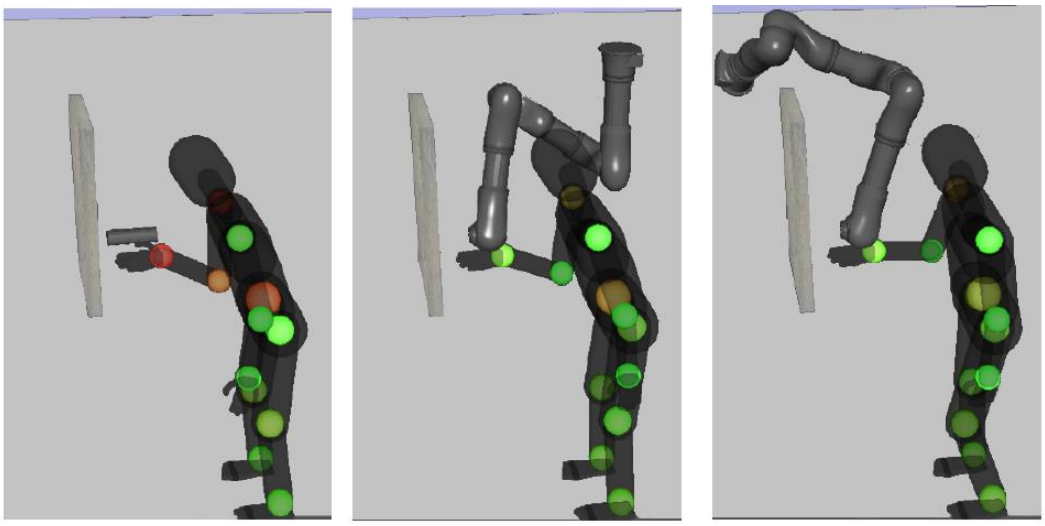


Figure 6.5: Snapshots of a digital human model performing a drilling activity without assistance (left) and with the assistance of two near-optimal collaborative robots (middle and right). The colored spheres represent the instantaneous level of joint effort: where maximum efforts are in red and minimum in green. [Maurice 2017].

Appendix A

Co-manipulation with a Library of Virtual Guiding Fixtures

The following document [Raiola 2017a] is a copy of the journal article we published in the Autonomous Robots Journal.

Noname manuscript No.
(will be inserted by the editor)

Co-manipulation with a Library of Virtual Guiding Fixtures

Gennaro Raiola · Susana Sanchez Restrepo · Pauline Chevalier · Pedro Rodriguez-Ayerbe · Xavier Lamy · Sami Tliba · Freek Stulp

the date of receipt and acceptance should be inserted later

Abstract Virtual guiding fixtures constrain the movements of a robot to task-relevant trajectories, and have been successfully applied to, for instance, surgical and manufacturing tasks. Whereas previous work has considered guiding fixtures for single tasks, in this paper we propose a library of guiding fixtures for multiple tasks, and propose methods for 1) Creating and adding guides based on machine learning; 2) Selecting guides on-line based on probabilistic implementation of guiding fixtures; 3) Refining existing guides based on an incremental learning method. We demonstrate in an industrial task that a library of guiding fixtures provides an intuitive haptic interface for joint human-robot completion of tasks, and improves performance in terms of task execution time, mental workload and errors.

G. Raiola[1,2,4]
E-mail: gennaro.raiola@ensta-paristech.fr

S. Sanchez Restrepo[2]
E-mail: susana.sanchezrestrepo@cea.fr

P. Chevalier[1]
E-mail: pauline.chevalier@ensta-paristech.fr

P. Rodriguez-Ayerbe[3]
E-mail: pedro.rodriguez@centralesupelec.fr

X. Lamy[2]
E-mail: xavier.lamy@cea.fr

S. Tliba[3]
E-mail: sami.tliba@lss.supelec.fr

F. Stulp[1,4,5]
E-mail: freek.stulp@dlr.de

[1] Robotics and Computer Vision, ENSTA-ParisTech, Palaiseau, France · [2] CEA-List, Gif-sur-Yvette, France · [3] Univ. Paris-Sud, CNRS, CentraleSupelec, Gif-sur-Yvette, France · [4] FLOWERS Team, INRIA, Bordeaux Sud-Ouest, France · [5] German Aerospace Center (DLR), Institute of Robotics and Mechatronics, Wessling, Germany

Keywords Human robot collaborative tasks in manufacturing · Learning from demonstration · Virtual fixture

1 Introduction

Recent improvements in the safety and (force) sensing capabilities of robots now enable humans to physically interact and solve tasks collaboratively with robots. The advantages of this collaboration is that it enables non-expert users to quickly teach robots new behaviours for new tasks. This is essential for modern assembly lines, where lot sizes are becoming ever smaller due to customization, and high degrees of flexibility are necessary to quickly adapt to changing markets (Hermann et al, 2016).

This flexibility and teach-in programming requires robots to be adaptive and to predict the intentions of humans, for which machine learning is a key enabler. In this paper, we apply machine learning and probabilistic methods to "virtual guiding fixtures" (Lin et al, 2006), so that non-expert users can teach new fixtures, and the robot is able to recognize on-line which fixture the human intends to select.

A virtual guiding fixture (Lin et al, 2006) constrains the motion of an end-effector to certain task-relevant trajectories. A well-known example of a guiding fixture from everyday life is the ruler, which allows us to draw very straight lines by constraining the movement of the pen tip along a 1-D trajectory on the 2-D paper. Robots are able to implement more complex virtual guiding fixtures, as illustrated in Fig. 1.

Whereas previous work has focussed on single virtual guides for single tasks, here we consider scenarios in which multiple tasks must be solved, and a library of virtual guides is thus necessary. Therefore, to maintain and evaluate such a library, we make the following contributions[1]:

- Apply incremental training of Gaussian Mixture Models (GMM), as previously used for Programming by Demonstration (PbD) in (Calinon, 2007), to create and refine virtual guides (Section 4 and 6).
- Define a controller to select among multiple virtual guides based on a probabilistic implementation of them (Section 5), which constitutes the main technical contribution of our work.
- Present an user study (Section 7), in which we evaluate the usability and impact of the library of virtual guides in the context of an industrial pick-and-place task.

The rest of this paper is structured as follows. In the next section, we discuss related work. In Section 3, we introduce

[1] Our previous work (Raiola et al, 2015a,b) focussed on the theoretical framework underlying multiple virtual guides, as well as an analysis of their stability. This paper focusses instead on the pragmatic implementation of a library of such guides, including a full user study.

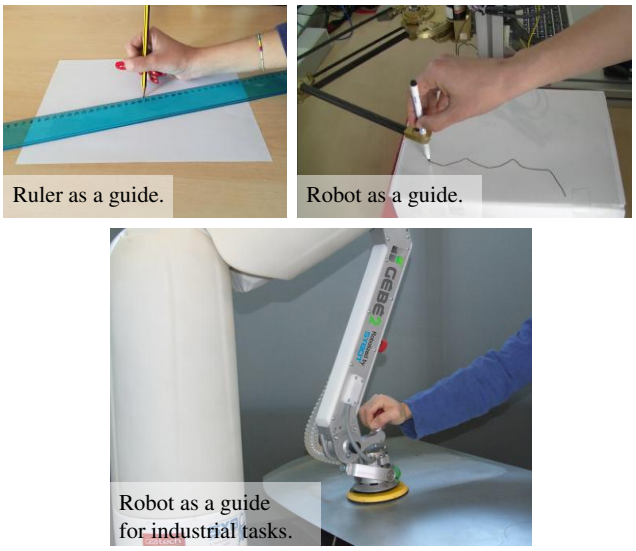


Fig. 1: Rulers simplify the drawing of lines, because they constrain (guide) the movement of the pencil tip (top left). Robots can similarly constrain human motions with virtual guides, but allow more flexibility on the shape of the guide (top right). Such virtual guides enable co-manipulation for industrial tasks (bottom).

a possible way to generate virtual guides by using virtual mechanisms as proposed in (Joly and Andriot, 1995), which forms the background of our work. In Section 4, we introduce how to create and add guides to the library. In Section 5 we present the controller which enables the on-line selection of multiple virtual guides. In Section 6 we discuss how to refine existing guides based on the incremental training of GMM. We present the user study in Section 7, and conclude with Section 8.

2 Related Work

Virtual guides are used to enforce virtual constraints on the movements of robots, in order to assist the user during a collaborative task. Virtual fixtures are especially useful in contexts where human decision making is still required to perform the overall task, but where constraints on the accuracy or required forces of the motion preclude humans from performing such tasks without robot assistance. Virtual fixtures were first introduced by (Rosenberg, 1993), where virtual fixtures are presented as an overlay of augmented sensory information on a workspace used to improve human performance in a teleoperated manipulation task. The fundamental concept is that virtual fixtures can reduce mental workload, task time, and errors during the collaborative task. After Rosenberg's initial work, the use of virtual fixtures has been extended to robotic surgery under the name of *active constraints* (Ho et al, 1995; Davies et al, 2006) and to industrial applications by (Colgate et al, 2003) in the context of *Intelligent Assist Devices*.

Nowadays, virtual fixtures has been featured in several different works, but unfortunately "there is currently no definitive concept which unifies the field" (Bowyer et al, 2014) because of the different definitions, applications and implementation methods. Generally, virtual fixtures has been used in teleoperation or comanipulation contexts. In teleoperation, the user controls a slave robot via a separate master device (Joly and Andriot, 1995; Aarno et al, 2005; Abbott, 2005; Bowyer and y Baena, 2013), this offers benefits such as motion scaling and the possibility to operate in restricted and unsafe environments, for example (Ryden et al, 2013) use virtual guides to teleoperate an underwater robot, while (David et al, 2014) proposed a supervisory control system to speed up a disk-cutter insertion process.

In a comanipulation context, the user directly interacts with the robot through physical contact (Raiola et al, 2015a; Becker et al, 2013; Dumora, 2014; Pezzementi et al, 2007). This allows a direct interaction between the robot and the user, and a more intuitive ability to perform the task since the user is better integrated in the procedure compared to the case where the user interacts with the environment through a teleoperated robot. The type of assistance offered by the virtual fixtures can vary among different definitions, but in general they are either used to guide the user along a task-specific pathway or to limit the user to move the robot within a safe region.

The particular implementation of virtual guides we use is based on the work presented by (Joly and Andriot, 1995), where a passive virtual mechanism is connected to the robot end-effector by a spring-damper system in a teleoperation context. Instead, we use the virtual mechanisms in a co-manipulation framework, i.e. the user is directly in contact with the robot. Virtual mechanisms have also been used by (Pezzementi et al, 2007), where they are called "proxies". Virtual guides may also be implemented by using anisotropic admittances to attenuate the non-preferred user force component (Marayong et al, 2003; Bettini et al, 2004). These methods require sensing external inputs, such as the force or the velocity applied by the user on the robot end-effector. This is not required with our control scheme.

Regarding the way virtual fixtures can be created, there are many possible solutions since there are different possible applications where they can be useful, usually the way to create them is strictly related to the goals of the application. In general, virtual fixtures have often been limited to predefined geometric shapes (Marayong et al, 2003) or combinations of shapes (Aarno et al, 2005; Kuang et al, 2004) or defined through well-defined geometric models (Joly and Andriot, 1995; Dumora, 2014). On the other hand, programming by demonstration (PbD) appears as a promising solution to program robots in a fast and simple way when the

task is known by the user. In PbD, teaching a path usually involves demonstrating the set of trajectories and retrieving a generalized representation of the data set suitable for reproduction by a robot. Generating guides from demonstrations has been explored by (Aarno et al, 2005), who model demonstrations in a segmented sequence of straight lines. Another interesting work about virtual fixtures and programming by demonstrations has been conducted by (Yoon et al, 2014). In this work the authors *personalize* the virtual fixture based on a set of demonstrations provided by the users in order to match their preferences about the guidance.

In our work, we use the demonstrations of the user to train Gaussian Mixture Models (GMM) as in (Calinon et al, 2007), which ensures smooth movements and explicitly models the variance in user demonstrations. Moreover, this allows us to define one of the novel aspect of our work, i.e. the *probabilistic* virtual guides. A first advantage of the probabilistic approach is that it enables a guide to be activated/deactivated based on the probability of belonging to it, which leads to smooth transitions. This is preferable to switching the guide on/off as in (Li and Okamura, 2003; Aarno et al, 2005; Yu et al, 2005), and does not require the manual design of distance thresholds for activation, as in (Nolin et al, 2003).

A second advantage is that the probabilistic approach allows us to simultaneously activate and recognize several guides, by assigning probabilities to each guide based on user behavior. Thus, our method enables the use of a *library of guides*, with one guide for each distinct task. Multiple guides have been previously used, but these (sub)guides are activated sequentially for one unique task, rather than in parallel for several tasks. For instance (Kuang et al, 2004) combine different shape primitives to facilitate maze navigation. (Aarno et al, 2005) use HMM to probabilistically choose a guide in a sequence of linear guides to accomplish a pick and place task.

Finally, we consider a co-manipulation instead of teleoperation framework, as is customary with PbD. In this respect, our work can be compared to (Amor et al, 2014; Medina et al, 2012; Rozo et al, 2016; Wrede et al, 2013) where the user and the robot have to execute a learned task together. Regarding the definition of the virtual guides through PbD, our work can be compared to the work done by (Vakanski et al, 2012; Mollard et al, 2015; Boy et al, 2007; Ewerton et al, 2016; Lee and Ott, 2011; Sanchez Restrepo et al, 2017) where the concept of *Task refinement* is exploited, as we will see in Section 6 this is possible due to the incremental training of GMM (Calinon, 2007).

3 Background: Virtual Mechanisms as Virtual Guides

We implement a virtual guide as a connection between the end-effector of the robot and a simulated virtual robot called "virtual mechanism" (Joly and Andriot, 1995). The equations and control schemes in (Joly and Andriot, 1995) are important background knowledge to understand our contributions, so we provide an overview of them in this section.

In general the virtual robot has fewer degrees of freedom than the real one, and thus the movements of the real robot are constrained by the possible movements of the virtual robot, see Fig. 2.

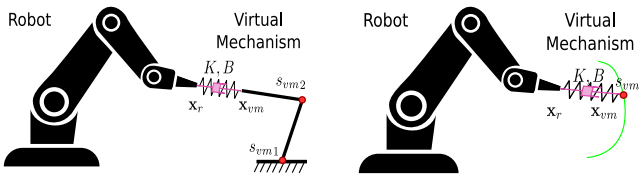


Fig. 2: Left: A virtual mechanism is a virtual (spring-damper) connection between the robot and the virtual robot with fewer degrees of freedom. $\mathbf{x}_{\text{vm}}$ and $\mathbf{x}_{\text{r}}$ represent respectively the end-effector position in Cartesian space of the virtual mechanism and the robot. s_{vmi} represents the degree of freedom of the virtual mechanism. Right: In our work, the virtual mechanism has only one degree of freedom represented by s_{vm}, which represents the movement along a trajectory, the virtual guide. The virtual mechanism can be thought as a cart moving along a rail, with the rail acting as the constraint.

The robot end-effector and the virtual "cart" mechanism are coupled by a spring-damper system. In this way if the robot end-effector moves, the cart is pulled along the rail in the direction of the movement, on the other hand, the cart also pulls the robot towards the rail, because the connection pulls in both directions. The overall effect is that the robot end-effector can be moved easily along the virtual rail, but not away from the rail. The position of the cart on the rail in Cartesian space is described by $\mathbf{x}_{\text{vm}}$. The distance it has traveled along the rail is function of the phase s_{vm}, with $s_{\text{vm}} = 0$ at the beginning and $s_{\text{vm}} = 1$ at the end of the rail, as illustrated in Fig. 3. The kinematics of the virtual mechanism is described by:

$$\mathbf{x}_{\text{vm}} = f(s_{\text{vm}}), \tag{1}$$

$$\dot{\mathbf{x}}_{\text{vm}} = \mathbf{J}_{\text{vm}}(s_{\text{vm}})\dot{s}_{\text{vm}}. \tag{2}$$

In Section 5.1 we will describe how to implement the functions $f(s_{\text{vm}})$ and $\mathbf{J}_{\text{vm}}(s_{\text{vm}})$ from user demonstrations.

3.1 Force on the virtual mechanism

The virtual mechanism is connected to the robot end-effector with a virtual spring-damper system. The force ap-

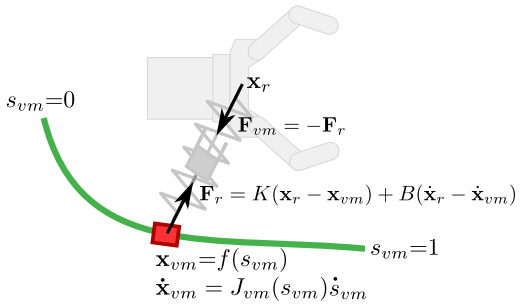


Fig. 3: The main variables and equations of the virtual mechanism.

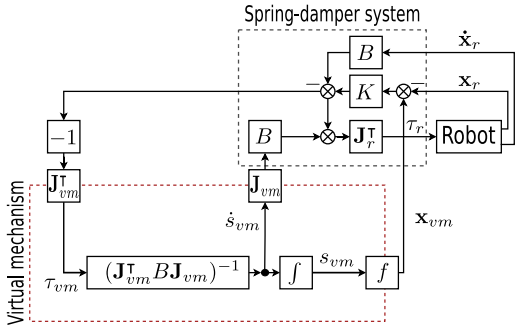


Fig. 4: Control scheme for the virtual mechanism.

plied to the virtual mechanism by the robot is:

$$\mathbf{F}_\mathrm{r} = K(\mathbf{x}_\mathrm{r} - \mathbf{x}_\mathrm{vm}) + B(\dot{\mathbf{x}}_\mathrm{r} - \dot{\mathbf{x}}_\mathrm{vm}). \tag{3}$$

The virtual mechanism is ideal, so the efforts applied on it are null

$$\mathbf{J}_\mathrm{vm}{}^\intercal \mathbf{F}_\mathrm{r} = 0, \tag{4}$$

which leads to

$$\mathbf{J}_\mathrm{vm}{}^\intercal (K(\mathbf{x}_\mathrm{r} - \mathbf{x}_\mathrm{vm}) + B(\dot{\mathbf{x}}_\mathrm{r} - \mathbf{J}_\mathrm{vm}\dot{s}_\mathrm{vm})) = 0. \tag{5}$$

By solving (5) with respect to $\dot{s}_\mathrm{vm}$, we obtain a first order dynamical system that expresses the evolution of the virtual cart along the virtual rail:

$$\dot{s}_\mathrm{vm} = (\mathbf{J}_\mathrm{vm}{}^\intercal B \mathbf{J}_\mathrm{vm})^{-1} \mathbf{J}_\mathrm{vm}{}^\intercal (K(\mathbf{x}_\mathrm{r} - \mathbf{x}_\mathrm{vm}) + B\dot{\mathbf{x}}_\mathrm{r}). \tag{6}$$

Moving the robot end-effector away from the virtual cart ($\mathbf{x}_\mathrm{r} \neq \mathbf{x}_\mathrm{vm}$) will thus make it slide along the rail, with a velocity described by (6)[2].

[2] Eq. (6) contains the inverse of the matrix $(\mathbf{J}_\mathrm{vm}{}^\intercal B \mathbf{J}_\mathrm{vm})$, which may lead to singularities. This problem and possible solutions are presented in Section 2.3 of (Raiola, 2017).

3.2 Force on the robot end-effector

Because the virtual mechanism and the robot end-effector are connected to *each other*, the virtual mechanism also applies a force on the robot end-effector, i.e.

$$\mathbf{F}_\mathrm{vm} = -\mathbf{F}_\mathrm{r} = K(\mathbf{x}_\mathrm{vm} - \mathbf{x}_\mathrm{r}) + B(\dot{\mathbf{x}}_\mathrm{vm} - \dot{\mathbf{x}}_\mathrm{r}). \tag{7}$$

This virtual force can be transformed into actual control commands for the robot, for instance with a compliance controller. In our implementation, we used the robot's Jacobian transposed $\mathbf{J}_\mathrm{r}^\intercal$ to convert the forces into torque references for the motor controllers. Fig. 4 illustrates the signals connections between the robot and the virtual mechanism.

4 Creating and Adding Guides

In the previous section, we explained how a virtual guide is implemented as a virtual mechanism. In this paper, the mechanism may be considered as a virtual cart on a rail (a 3D trajectory), which is connected to the robot end-effector with a spring-damper system. In this section, we present a method for adding a new guide (rail) to a library of guides through demonstrations and machine learning.

4.1 Gaussian Mixture Model

In our approach, virtual guides are extracted from (multiple) user demonstrations by training a Gaussian Mixture Model with Expectation Maximization, as in (Calinon et al, 2007).

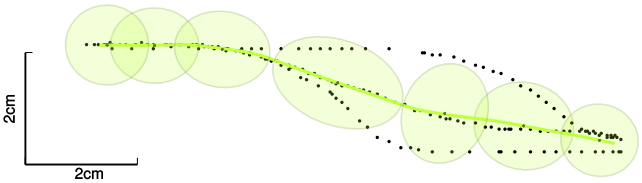


Fig. 5: Example of a Gaussian Mixture Model, trained on three demonstrated trajectories

In a GMM, the demonstrated data is modelled by a mixture of K components defined by a probability density function:

$$p(\zeta_m) = \sum_{k=1}^{K} p(k)p(\zeta_m|k), \tag{8}$$

where $\{\zeta_m\}_{m=1}^{M}$ represents the demonstrated set of Cartesian points of dimension D, $p(k)$ is the prior probability and $p(\zeta_m|k)$ is the conditional probability. A Gaussian Mixture Model can be fully described by its parameters θ which are

$\theta = \{\pi_k, \mu_k, \Sigma_k, M\}_{k=1}^{K}$, respectively the priors, the means, the covariance matrices and the number of samples[3] in ζ.

For a mixture of K components of dimensionality D the parameters in (8) are defined as:

$$\begin{aligned} p(k) &= \pi_k, \\ p(\zeta_m|k) &= \mathcal{N}(\zeta_m; \mu_k, \Sigma_k), \qquad (9) \\ &= \frac{e^{\left(-\frac{1}{2}(\zeta_m-\mu_k)^\intercal {\Sigma_k}^{-1}(\zeta_m-\mu_k)\right)}}{\sqrt{(2\pi)^D|\Sigma_k|}}. \qquad (10) \end{aligned}$$

The log-likelihood of the model described by θ, given a set of M datapoints $\{\zeta_m\}_{m=1}^{M}$ is:

$$\mathcal{L}(\theta) = \frac{1}{M}\sum_{m=1}^{M} \ln(p(\zeta_m)). \qquad (11)$$

Where $p(\zeta_m)$ is the probability that ζ_m has been generated by the model, which is computed using (8).

4.1.1 Training the Gaussian Mixture Model

The GMM is initially trained from user demonstrations, using the approach described in (Calinon et al, 2007; Raiola et al, 2015a), and briefly repeated here. The demonstrated trajectories consist of samples $\{\zeta_m = [x(t_m), y(t_m), z(t_m)]\}_{m=1}^{M}$. Multiple trajectories are first aligned using Dynamic Time Warping. Then, each sample in a trajectory is associated with a phase value defined as $s(t_m) = (t_m - t_1)/(t_M - t_1)$, i.e. $s(t_1) = 0$ at the beginning of the demonstration, and $s(t_M) = 1$ at the end. The resulting samples in one trajectory then have the format $\{[x_m, y_m, z_m, s_m]\}_{m=1}^{M}$.

Fitting the GMM to this data is done with the Expectation-Maximization algorithm (EM). EM incrementally adjusts the priors π_k and the parameters μ_k and Σ_k of the Gaussian functions to fit the data until a stop criterion is met. This algorithm guarantees monotone increase of the likelihood of the training set during optimization.

4.2 Gaussian Mixture Regression

Virtual mechanisms require implementations of the kinematics equations $\mathbf{x}_{\text{vm}} = f(s_{\text{vm}})$ (1) and $\dot{\mathbf{x}}_{\text{vm}} = \mathbf{J}_{\text{vm}}(s_{\text{vm}})\dot{s}_{\text{vm}}$ (2). These are extracted from the GMM through Gaussian Mixture Regression (GMR). In the context of a virtual mechanism, the input space is S, and the output space is X, respectively the phase s_{vm} and virtual mechanism's position $\mathbf{x}_{\text{vm}}$. Given this partition, the mean and covariance matrix[4] are decomposed as

$$\mu_k = [\mu_{k,S}^\intercal, \mu_{k,X}^\intercal]^\intercal \text{ and } \Sigma_k = \begin{bmatrix} \Sigma_{k,S} & \Sigma_{k,SX} \\ \Sigma_{k,XS} & \Sigma_{k,X} \end{bmatrix}, \qquad (12)$$

The implementation of $\mathbf{x}_{\text{vm}} = f(s_{\text{vm}})$ in (1) corresponds to computing $\overline{\mathbf{x}}_{\text{vm}} = E(\mathbf{x}_{\text{vm}}|s_{\text{vm}})$, i.e. the expectation of $\mathbf{x}_{\text{vm}}$ given the input s_{vm}:

$$\overline{\mathbf{x}}_{\text{vm}} = \sum_{k=1}^{K} \beta_k(s_{\text{vm}})(\mu_{k,X} + \Sigma_{k,XS}\Sigma_{k,S}^{-1}(s_{\text{vm}} - \mu_{k,S})), \qquad (13)$$

with:

$$\beta_k(s_{\text{vm}}) = \frac{\pi_k g(\mathbf{x}; \mu_{k,S}, \Sigma_{k,S})}{\sum_{l=1}^{K} \pi_l g(\mathbf{x}; \mu_{l,S}, \Sigma_{l,S})} = \frac{\pi_k g(\mathbf{x}; s_{\text{vm}}^k)}{\sum_{l=1}^{K} \pi_l g(\mathbf{x}; s_{\text{vm}}^l)}. \qquad (14)$$

The function g represents a Gaussian distribution defined as:

$$g(\mathbf{x}; \mu, \Sigma) = \frac{e^{\left(-\frac{1}{2}(\mathbf{x}-\mu)^\intercal \Sigma^{-1}(\mathbf{x}-\mu)\right)}}{\sqrt{(2\pi)^D|\Sigma|}}. \qquad (15)$$

The function $\mathbf{J}_{\text{vm}}(s_{\text{vm}})$ in (2) is implemented with the analytical derivative of (13) in respect of s_{vm}.

In summary, the kinematics of the virtual mechanism $\mathbf{x}_{\text{vm}} = f(s_{\text{vm}})$ is computed with Gaussian Mixture Regression (13), based on a Gaussian Mixture Model (8), whose parameters are trained by applying the Expectation Maximization algorithm to a set of demonstrated trajectories.

5 Selecting Guides

In a library of virtual guides, different guides exist to solve different tasks, see Fig. 6. Our aim is to enable the robot to recognize on-line which task the user intends to solve. To avoid abrupt switches, we implement a control scheme in which all mechanisms are simultaneously active, but scaled with the probability that the task with which the mechanism is associated is being solved. Thus, the final force $\mathbf{F}_{\text{res}}$ applied to the end-effector is a weighted sum of the forces from each guide $\mathbf{F}_{\text{vm}}^{n=1\ldots N}$:

$$\mathbf{F}_{\text{res}} = \sum_{n=1}^{N} p_n \mathbf{F}_{\text{vm}}^n. \qquad (16)$$

Our approach requires the computation of the probabilities $p_{n=1\ldots N}$, which represent the probability that the n^{th} guide is responsible for the current task. To do so, we first propose "probabilistic virtual mechanisms", show how they enable $p_{n=1\ldots N}$ to be computed, and propose different interaction modes based on the exact scaling in (16).

[3] Note that M is not strictly necessary to describe the model but it will be useful for the incremental training.

[4] The covariance matrix $\Sigma_{e,S}$ is actually a scalar, because the phase is always 1-dimensional. For consistency, we nevertheless use the bold symbol Σ rather than σ^2.

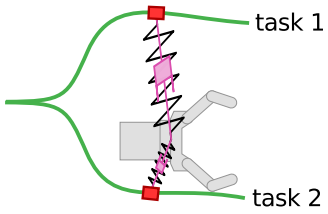


Fig. 6: Multiple virtual mechanisms – one for each task – simultaneously connected to the robot end-effector.

5.1 Probabilistic Virtual Mechanism

We define a probabilistic virtual mechanism as a virtual mechanism in which there is uncertainty about the position of the virtual end-effector, i.e. the position of the cart on the rail. This uncertainty is represented by a Gaussian distribution, as visualized in Fig. 7.

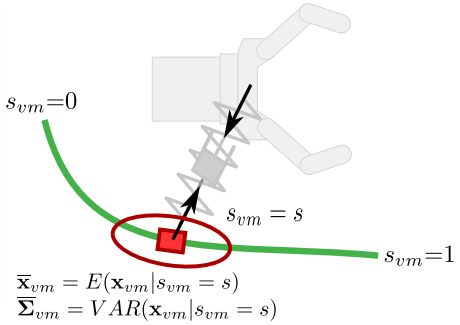


Fig. 7: The current state of the virtual mechanism is modelled as a multi-variate Gaussian distribution $\mathcal{N}(\overline{\mathbf{x}}_{\text{vm}}, \Sigma_{\text{vm}})$.

The covariance matrix of the distribution is readily computed from the Gaussian Mixture Model through Gaussian Mixture Regression by computing the conditional variance $VAR(\mathbf{x}_{\text{vm}}|s_{\text{vm}})$:

$$\Sigma_{\text{vm}} = \sum_{k=1}^{K} \beta_k(s_{\text{vm}})^2 \left(\Sigma_{k,X} - \Sigma_{k,XS}\Sigma_{k,S}^{-1}\Sigma_{k,XS}^{\mathsf{T}}\right). \tag{17}$$

Thus, the (uncertain) position of the virtual mechanism is represented by the Gaussian distribution:

$$\overline{\mathbf{x}}_{\text{vm}} = E(\mathbf{x}_{\text{vm}}|s_{\text{vm}} = s), \tag{18}$$

$$\Sigma_{\text{vm}} = VAR(\mathbf{x}_{\text{vm}}|s_{\text{vm}} = s). \tag{19}$$

We now show how the probabilistic virtual mechanism is used to compute the probabilities $p_{n=1\ldots N}$ for each of the N guides in the library.

5.2 Probabilistic Weighting

Fig. 8 illustrates the association problem when using multiple guides. The two Gaussian Mixture Models represent the two virtual guides, which are associated with two different tasks. The inset to the right shows how the robot end-effector $\mathbf{x}_r$ is connected to both of the virtual mechanism (the "carts"). Because $\mathbf{x}_r$ is closer to the lower cart 2, it is more likely that the user intends to execute task 2, and the force exerted by the cart 2 should be higher than that exerted by cart 1. This intuition is implemented with the probabilistic weighting scheme.

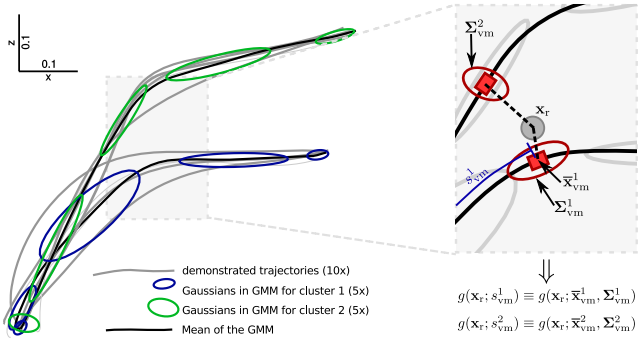


Fig. 8: Left: demonstrated trajectories (light gray) and the two GMMs. Right: Relevant variables for computing $g(\mathbf{x}_{\text{r}}; s_{\text{vm}})$.

If we have N virtual mechanisms, there are N cart positions $\mathbf{x}_{\text{vm}}^{n=1:N}$, and N probabilities. The probability p_n that the n^{th} cart is responsible for guiding the end-effector at position $\mathbf{x}_{\text{r}}$ is

$$p(n; \mathbf{x}_{\text{r}}, s_{\text{vm}}^n) = \frac{g(\mathbf{x}_{\text{r}}; \mu_{\text{vm}}^n, \Sigma_{\text{vm}}^n)}{\sum_{i=1}^{N} g(\mathbf{x}_{\text{r}}; \mu_{\text{vm}}^i, \Sigma_{\text{vm}}^i)} = \frac{g(\mathbf{x}_{\text{r}}; s_{\text{vm}}^n)}{\sum_{i=1}^{N} g(\mathbf{x}_{\text{r}}; s_{\text{vm}}^i)}, \tag{20}$$

where the means and covariance matrices of the cart position are determined from the cart phase s_{vm} with (13) and (17) respectively.

Each of the N virtual mechanisms applies a force $\mathbf{F}_{\text{vm}}^n$ to the end-effector. The relative influence of each VM is scaled with the probability $p(n; \mathbf{x}_{\text{r}}, s_{\text{vm}}^n)$, so that the resultant force on the end-effector is[5]:

$$\mathbf{F}_{\text{res}} = \sum_{n=1}^{N} p(n; \mathbf{x}_{\text{r}}, s_{\text{vm}}^n)\mathbf{F}_{\text{vm}}^n. \tag{21}$$

As described in (Raiola et al, 2015a), the underlying assumption in using (21) is that $\mathbf{x}_{\text{r}}$ *must* belong to one of the VMs[6]. Another approach is to assume that if $\mathbf{x}_{\text{r}}$ is too far from the VMs, it does not belong to any of the VMs. To do so, we use a Gaussian function $h(\mathbf{x}_{\text{r}}; \mu_{\text{vm}}, \Sigma_{\text{vm}})$ combined with (21), i.e. a probability density function as in (10), but

[5] The stability of such probabilistically weighted virtual guides is analyzed in (Raiola et al, 2015b).

[6] For this reason, we call the resulting guides "Hard Guides"

without the normalization factor $\sqrt{(2\pi)^k|\Sigma_{\text{vm}}|}$, as the Gaussian function has a known maximum of 1.

$$h(\mathbf{x},\mathbf{x}_{\text{vm}}) = e^{\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}_{\text{vm}})^{\mathsf{T}}\Sigma_{\text{vm}}{}^{-1}(\mathbf{x}-\mathbf{x}_{\text{vm}})\right)}. \tag{22}$$

By using these weights (to determine if an *individual* virtual guide is active in the first place), as well as the probability $p(n;\mathbf{x}_{\text{r}},s^n_{\text{vm}})$ (to determine the relative weighting *between all the guides*), the resultant force becomes

$$\mathbf{F}_{\text{res}} = \sum_{n=1}^{N} h(\mathbf{x}_{\text{r}};s^n_{\text{vm}})p(n;\mathbf{x}_{\text{r}},s^n_{\text{vm}})\mathbf{F}^n_{\text{vm}}. \tag{23}$$

The overall effect of this weighting scheme is the following: if the robot end-effector is not close to any guide, it is simply in zero-gravity mode, as none of the guides exerts a force to pull the end-effector towards the guide[7]. As soon as the robot end-effector approaches one of the guides, it starts exerting a force, and the end-effector is pulled toward the guide. The relative scaling between the guides is determined by the relative probability that the guide is responsible.

6 Refining Guides

To *modify* a guide we can exploit the incremental EM presented in (Calinon, 2007). After the user provides a new demonstration, the incremental clustering detects if the demonstration belongs to an existing guide, see Section 6.2. In this case, the demonstration is used to incrementally train the GMM which gets updated with the new data. Otherwise the demonstration is used to create a new guide, see Fig. 9.

6.1 Incremental GMM Estimation

The idea is to adapt the classic EM algorithm by splitting the part related to the old data from the part dedicated to the newly demonstrated data (Calinon, 2007). The update of the model is done under the assumption that the set of posterior probabilities $\{p(k|\zeta_m)\}_{j=1}^{M}$ remains the same when the new data $\{\tilde{\zeta}\}_{m=1}^{\tilde{M}}$ is used to update the model, this is called *data coherency constraint*. This assumption is true only if the new data is close to the trained model. This means that it is necessary to determine if the new data belongs or not to an already trained GMM (as anticipated, we will address this problem in the next section). Thus, the model is first created using the classic EM algorithm. Starting from an initial estimation[8] of the GMM with parameters $\{\pi^0_k,\mu^0_k,\Sigma^0_k\}_{k=1}^{K}$, at each step the following two steps are performed until a stop criterion is met:

E-step:

$$p^{t+1}_{k,m} = \pi^t_k \mathcal{N}(\zeta_m;\mu^t_k,\Sigma^t_k), \tag{24}$$

$$E^{t+1}_k = \sum_{m=1}^{M} p^{t+1}_{k,m}. \tag{25}$$

M-step:

$$\pi^{t+1}_k = \frac{E^{t+1}_k}{M}, \tag{26}$$

$$\mu^{t+1}_k = \frac{\sum_{m=1}^{M} p^{t+1}_{k,m}\zeta_m}{E^{t+1}_k}, \tag{27}$$

$$\Sigma^{t+1}_k = \frac{\sum_{m=1}^{M} p^{t+1}_{k,m}(\zeta_m-\mu^{t+1}_k)(\zeta_m-\mu^{t+1}_k)^{\mathsf{T}}}{E^{t+1}_k}. \tag{28}$$

The EM algorithm stops after a certain number of iterations T when

$$\frac{\mathscr{L}^{t+1}}{\mathscr{L}^t} - 1 < \mathscr{C}, \tag{29}$$

with the $\mathscr{L}$ defined in (11)[9]. The resulting GMM is completely defined by the set of parameters $\theta = \{\pi^T_k,\mu^T_k,\Sigma^T_k,M\}_{k=1}^{K}$.

When a new demonstration is provided by the user, $\tilde{T}$ steps are performed to update the model with the new data $\tilde{\zeta}$ with initial condition given by the previous model $\{\tilde{\pi}^0_k,\tilde{\mu}^0_k,\tilde{\Sigma}^0_k,\tilde{E}^0_k\}_{k=1}^{K} = \{\pi^T_k,\mu^T_k,\Sigma^T_k,E^T_k\}_{k=1}^{K}$ with $\tilde{E}^0_k = \tilde{\pi}^0_k M$.

The EM algorithm can be rewritten as:

E-step:

$$\tilde{p}^{t+1}_{k,m} = \tilde{\pi}^t_k \mathcal{N}(\tilde{\zeta}_m;\tilde{\mu}^t_k,\tilde{\Sigma}^t_k), \tag{30}$$

$$\tilde{E}^{t+1}_k = \sum_{m=1}^{\tilde{M}} \tilde{p}^{t+1}_{k,m}. \tag{31}$$

M-step:

$$\tilde{\pi}^{t+1}_k = \frac{\tilde{E}^0_k + \tilde{E}^{t+1}_k}{M+\tilde{M}}, \tag{32}$$

$$\tilde{\mu}^{t+1}_k = \frac{\tilde{E}^0_k\tilde{\mu}^0_k + \sum_{m=1}^{\tilde{M}} \tilde{p}^{t+1}_{k,m}\tilde{\zeta}_m}{\tilde{E}^0_k + \tilde{E}^{t+1}_k}, \tag{33}$$

$$\tilde{\Sigma}^{t+1}_k = \frac{\tilde{E}^0_k(\tilde{\Sigma}^0_k + (\tilde{\mu}^0_k-\tilde{\mu}^{t+1}_k)(\tilde{\mu}^0_k-\tilde{\mu}^{t+1}_k)^{\mathsf{T}})}{\tilde{E}^0_k + \tilde{E}^{t+1}_k}, \tag{34}$$

$$+ \frac{\sum_{m=1}^{\tilde{M}} \tilde{p}^{t+1}_{k,m}(\tilde{\zeta}_m-\tilde{\mu}^{t+1}_k)(\tilde{\zeta}_m-\tilde{\mu}^{t+1}_k)^{\mathsf{T}}}{\tilde{E}^0_k + \tilde{E}^{t+1}_k}. \tag{35}$$

Also in this case, the number of iterations $\tilde{T}$ is determined by the stop criterion defined in (29).

[7] We call the resulting guides "Soft Guides"

[8] In practice, the initial estimation is frequently performed using the K-means clustering algorithm which defines the initial values for the priors, means and covariance matrices.

[9] The threshold $\mathscr{C} = 0.01$ is used in our case.

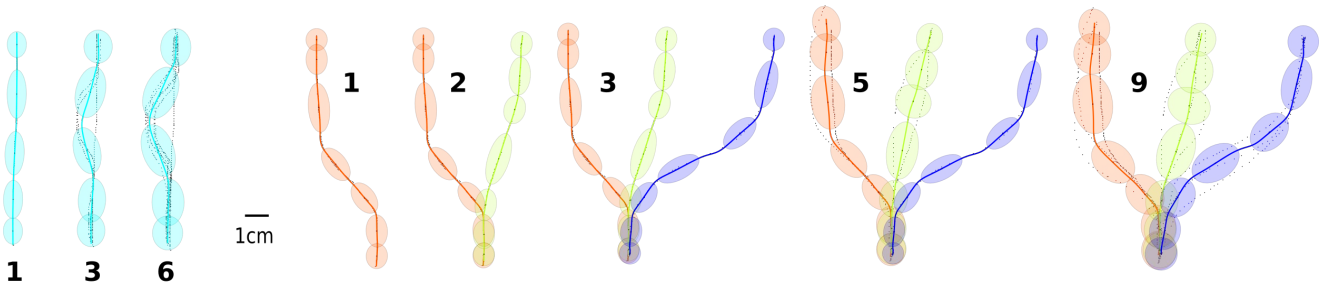


Fig. 9: Left: With the incremental training of GMM it is possible to iteratively modify an existing virtual guide. The number indicates the number of demonstrated trajectories used to incrementally train the guide. Right: Incremental training including incremental clustering, as explained in Section 6.2.

6.2 Incremental clustering

When the user demonstrates a new trajectory, we have to automatically detect if the new data belongs to one of the guide that has been previously created or if can be used to create a new one. This is necessary given the constraints on the data coherency of the proposed incremental EM. When the user demonstrates a new trajectory, the new data $\tilde{\zeta}$ is used to create a new GMM with the following parameters $\theta_{new} = \{\pi_k, \mu_k, \Sigma_k, M\}$. Since these parameters are the result of a set of EM steps, the associated likelihood represents the maximum likelihood, i.e. $L(\theta_{new}|\tilde{\zeta}) = L(\theta_{ML}|\tilde{\zeta})$. We can use the maximum likelihood $L(\theta_{ML}|\tilde{\zeta})$ as a baseline to select which GMM best fits the data $\tilde{\zeta}$. To perform the comparison we use the relative likelihood (Held and Bov, 2013) expressed as:

$$\hat{L}(\theta_n|\tilde{\zeta}) = \frac{L(\theta_n|\tilde{\zeta})}{L(\theta_{ML}|\tilde{\zeta})}, \forall\, n = 1..N, \tag{36}$$

where $L(\theta_n|\tilde{\zeta})$ represents the likelihood of an existing model n given the new demonstrated data $\tilde{\zeta}$. In particular, we have $1 \geq \hat{L}(\theta_n) \geq 0$ and $\hat{L}(\theta_{ML}) = 1$; because of this property, the relative likelihood is also called the normalized likelihood. The same expression can be computed using the log-likelihood, i.e. $\hat{\mathscr{L}}(\theta_n) = \log(\hat{L}(\theta_n)) = \mathscr{L}(\theta_n) - \mathscr{L}(\theta_{ML})$ where for the log-likelihood we have $0 \geq \hat{\mathscr{L}}(\theta_n) > -\inf$ with $\hat{\mathscr{L}}(\theta_{ML}) = 0$. For simplicity, we omitted the data set $\tilde{\zeta}$ since is the same in each comparison. We can compute the relative likelihood $\hat{L}(\theta_n)$ for each existing GMM. As proposed in (Held and Bov, 2013), we can select the model to update by using the following categorization based on the relative likelihood and a threshold c:

$$1 \geq \hat{L}(\theta_n) > c. \tag{37}$$

The threshold c can be selected arbitrarily. For example, we could categorize the likelihood as:

$$1 \geq \hat{L}(\theta_n) > \frac{1}{3} \quad \theta_n \text{ very plausible}, \tag{38}$$

$$\frac{1}{3} \geq \hat{L}(\theta_n) > \frac{1}{10} \quad \theta_n \text{ plausible}, \tag{39}$$

$$\frac{1}{10} \geq \hat{L}(\theta_n) \geq 0 \quad \theta_n \text{ not plausible}. \tag{40}$$

However, such a pure likelihood approach to inference has the disadvantage that the threshold c is somewhat arbitrarily chosen. The candidate model to be updated with the new incoming data is chosen in the interval given by (38). Between all the models that satisfy this inequality, we select the model with the maximum $\hat{L}(\theta_n)$. If none of the available models satisfy (38), the model θ_{ML} is used to create a new guide. This method requires the creation of a new GMM each time new data is provided. By creating a new model, we can keep track of the updates, meaning that the user can, at any time, revert a guide to its original shape. The advantages of this method are: it is easy to implement, fast and configurable due to the parameter c, does not require storing the previous data (only the GMM parameters are stored). The main drawbacks are: the selection and the significance of c and the necessity to create a new GMM that could not be used.

7 Experimental Evaluation

The following experiment was conducted on a 3-DOF ISYBOT[10] comanipulation robot with a gripper, see Fig. 11. The general task was to use the robot and the library of virtual guides[11] to simulate pick and place operations. For the virtual guide assistance, the stiffness was set as $K = kI$ with $k = 10000\,N/m$ and the damping as $B = bI$ with $b = 400\,N/ms^{-1}$.

[10] http://www.isybot.com

[11] The code used to generate and interact with the library of virtual guides is available at https://github.com/graiola/virtual-fixtures

To create the virtual guides we used the incremental training presented in Section 6 with a fixed number of 10 Gaussians per model. Each demonstration was composed by $M = 2000$ samples $\{\zeta_m = [x(t_m), y(t_m), z(t_m)]\}_{m=1}^{M}$ recorded at 100 Hz. The phase s for each time step was computed with dynamic time warping and the mapping $s(t_m) = (t_m - t_1)/(t_M - t_1)$, as explained in Section 4.1.1.

7.1 User Study

We designed the study to observe: (1) how novice users perceived the virtual guide assistance with multiple guides, (2) to determine if creating new virtual guides with the library is intuitive and comfortable. We recruited 20 participants (with age between 22 and 33 years old, 7 females). Twelve participants stated they had prior experience with robots. We divided the user study in 4 sessions:

1 The user performs a pick and place task without the guides. (pp_1)
2 The user performs a pick and place task with multiple *default* guides active. (pp_2)
3 The user is *trained* on how to use the library of guides, afterwards the user is able to create his *personal* set of guides. (tr)
4 The user performs the pick and place task with the guides created in the previous session. (pp_3)

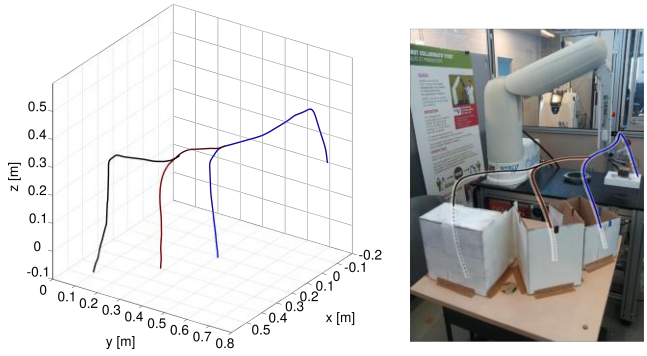


Fig. 10: Default virtual guides created by the expert user. Left: The colored lines represent the mean of the GMMs. Right: Virtual guides in the robot workspace.

The *default* guides for the session pp_2 were generated by an expert user with the library of virtual guides, see Fig. 10.

Four hypotheses were tested:

- H1: Virtual guides assistance improves task's performances, in terms of time and collisions occurrences.
- H2: Virtual guides assistance is more helpful when the task requires higher level of attention.
- H3: Virtual guides assistance is perceived as useful by the users.
- H4: It is intuitive and comfortable for novice users to create new virtual guides.

All participants were asked to perform the four sessions. The sessions $pp_{1,2}$ were presented in a randomized order to avoid training effects, while pp_3 was always presented after the session tr. At the beginning of $pp_{1,2}$, the participants were able to familiarize with the system.

7.2 Task explanation

The task in $pp_{1,2,3}$ consisted in taking 6 discs from the robot's workstation and insert them inside specific boxes identified with 3 different colors: blue, brown and black. For each box there were two discs with a piece of tape of the same color, see Fig. 11. The objective of the task was to place the discs in the associated box trying to minimize the time in respect of two constraints:

- The participant had to avoid collisions between the robot and the boxes.
- The discs had to be placed gently inside the boxes (it was not possible to drop the discs in the boxes to save time).

The boxes were disposed to obtain an increasing difficulty in terms of distance and accessibility ranging from the easiest (blue) to the hardest (black), see Fig. 11.

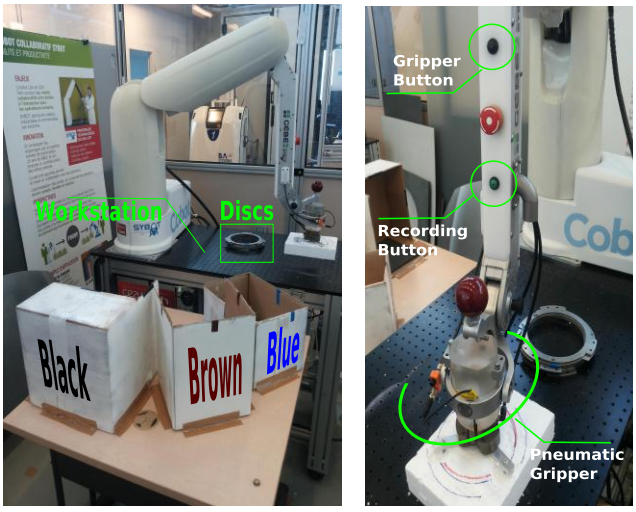


Fig. 11: Setup for the user study (left) and buttons used for the experiment (right). The upper button was used to hold and release the discs with the pneumatic gripper, while the lower button was used to start and stop the recording of the demonstrations.

We measured the total task time (T_j) necessary to complete the single session pp_j with $j = 1,2,3$ (the time T_j was

taken starting from the pick of the first disc and ending when the last disc was placed) and the pick and place time for each disc and session ($t_{i,j}$) with $i = 1,..,6$ (which leads to $18 = 6x3$ measures for each participant). The total time T_j differs from the sum over the single times $\sum_{i=1}^{6} t_{i,j}$ because it includes the time necessary for the user to pick the disc with the robot and to bring the robot back to the workstation after each placing.

In session tr the experimenter explained to the participants how to interact with the system, see Fig. 11. The participants were able to create their own guides in order to execute the task in session pp_3. During this session, the participants were allowed to ask for help from the experimenter. No time was recorded in tr.

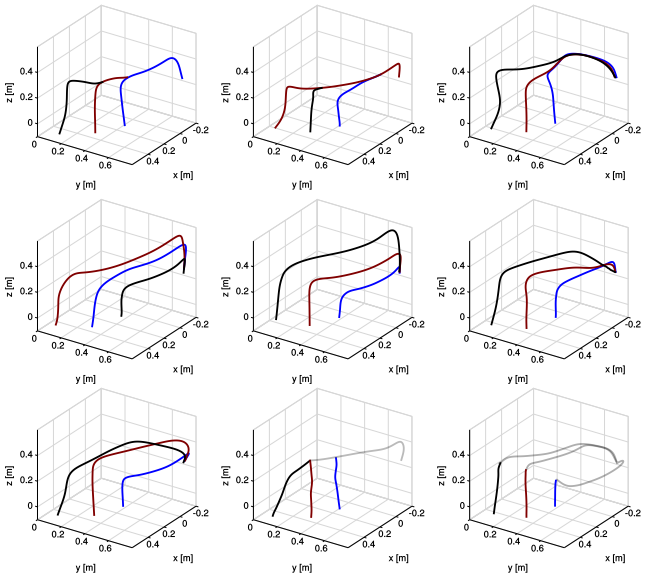


Fig. 12: Different approaches to the guides creation done by eight of our participants. Blue, brown and black curves are the guides created to place the disc inside the respective box. Gray curves are extra guides created to help connecting the guides. For comparison in the upper-left corner there are the guides created by the expert user.

Resulting guides from eight different users are shown in Fig. 12. At the end of $pp_{1,2,3}$ the participants answered a post-condition survey focusing on the usage experience with the virtual guides on the form of a Likert-scale survey with a rating from 1 to 7, with 1 as strong disagreement and 7 as strong agreement, see Table 1. For session tr the participants answered to a different survey focusing on the creation of the virtual guides, see Table 2.

7.3 Results

To validate our hypothesis we measured:

- Total task time T_j for each session $pp_{1,2,3}$ and pick and place time $t_{i,j}$ for each disc and session. Both are used to validate H1 and H2.
- Observed collisions, to validate H1 and H2.
- Survey results for sessions $pp_{1,2,3}$ to validate H3.
- Survey results for session tr to validate H4.

We performed a repeated-measure ANOVA (Girden, 1992) on T_j, $t_{i,j}$ and on the survey, on three factors: (1) the sessions pp_j with $j = 1,2,3$, (2) the difficulty, represented by the three boxes (blue; brown; black) and (3) the repetitions for each box $(r_1; r_2)$. Posthoc analyses were performed with Tuckey's HSD test (Abdi and Williams, 2010). For the collisions we observed that the participants collided with the boxes only during $pp_{1,3}$. For this reason, we performed Fisher-exact test between $pp_{1,3}$ on the number of participants that did at least one collision during the task and those that did not collide during the task. The significance threshold was set to $p < 0.05$.

7.3.1 Time Analysis

Effects of sessions on time:
We found a statistically significant difference $(p = .0023)$ between the sessions $pp_{1,2,3}$ on the total time (T) (Fig. 13, Left). Posthoc analysis shows that $pp_{1,2}$ and $pp_{1,3}$ are statistically different $(p = .005, p = .005)$. In pp_1 (No Guides) the participants were slower than in $pp_{2,3}$ (Default and Personal Guides). In addition, we found a statistically significant difference $(p = .00076)$ between the sessions on the pick and place time (t) (Fig. 13, Right). Also in this case, the posthoc analysis shows that $pp_{1,2}$ and $pp_{1,3}$ are statistically different, with $(p = .004, p = .001)$ respectively. We found again that in pp_1 the participants were slower than in $pp_{2,3}$. These two results enlighten that the virtual guides reduced the time to complete the task (both total time and pick and place time for each disc). This validates H1. Moreover we found that there is not statistical difference between the execution time with default and personal guides. This indicates us that the users were able to create guides that were as efficient as the default guides created by the expert user. This is an indication that H4 may be true, which we further discuss in Section 7.3.3.

Effects of the difficulty on time:
As pointed out in 7.2, the difficulty related to the disc insertion is different between the boxes $(p < .001)$. This can be seen in Fig. 14. Even if not statistically relevant, we reported also the t related to each box and each session. We can observe that the disc insertion for the black box requires more time without guides, this can be explained with the distance of the box from the workstation and with its disposition that does not facilitate the disc insertion. Instead, with the guides the time seems to increase linearly, meaning that

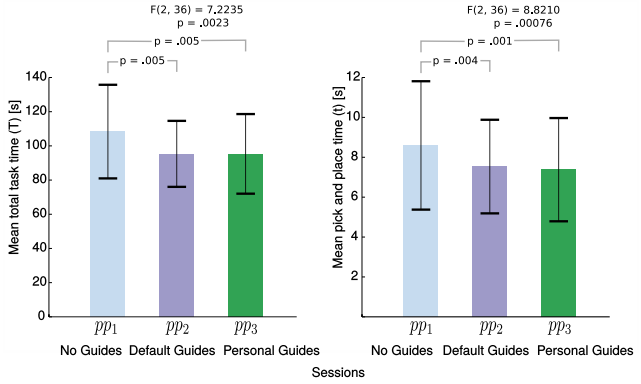


Fig. 13: Left: Mean of the total task time (T). Right: Mean of pick and place time (t).

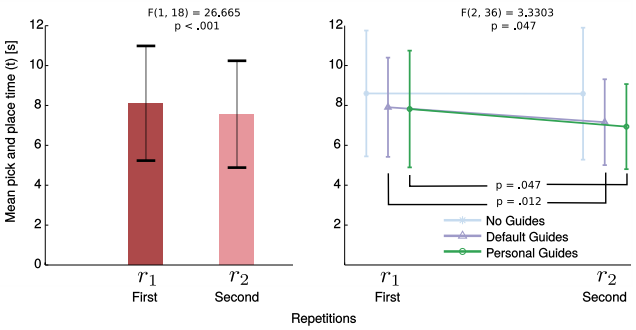


Fig. 15: Left: Mean of t for the two repetitions. Right: Mean of t for the two repetitions and each session.

the box disposition does not affect the insertion but only the distance does. These results support H2.

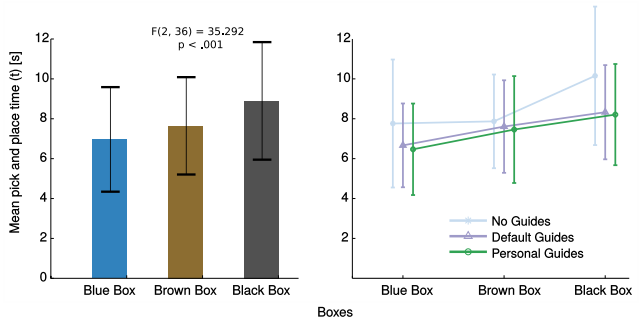


Fig. 14: Left: Mean of t for each box. Right: Mean of t for each box and each session.

Effects of repeated disc insertions on time:
We found a statistically significant effect $(p < .001)$ of the repeated insertions on t, see Fig. 15. The second repetition is shorter than the first. This represents a training effect on repeated disc insertions. However, we find a statistically significant interaction effect $(p = .047)$ between the sessions and the repetition. Posthoc analysis shows that in pp_1 there is no statistical difference between the two repetitions but in $pp_{2,3}$ the second repetition is shorter than the first one, with $(p = .047, p = .012)$ respectively. This informs us that, with guides, there is a training effect: repetitive use of virtual guides can improve the user performances. This is not necessary to prove H1 but it is a factor to take into account when using virtual guides.

7.3.2 Collisions Analysis

For the collisions, we measured that the participants collided with the boxes only during $pp_{1,3}$. In pp_2, the collisions were not possible due to the guides created by the expert user (Fig. 10). In pp_3, collisions occurred because some participants did not proof-test their guides. For the collision, we found a statistical difference $(p = 0.0001)$ between pp_1 and pp_3: in pp_1, 18 of the 20 participants had at least one collision during the task, while in pp_3 only 6 of the 20 participants did. This indicates that using virtual guides leads to a safer task execution (Fig. 16), which goes in the direction of H1. Another observation could be done on the number of collisions for each box. As shown in Fig. 16, the majority of collisions occurred with the black box when the guides were not available. When the guides are used, the number of collisions with the black box reduces drastically (respectively 0 collisions with default guides and 1 collision with personal guides). This last result goes in direction of H2. The higher number of collisions with the blue box when the personal guides are used can be explained with the fact that the participants often started to create a guide for the blue box; this lead to a higher number of mistakes since it was their first training trial with the system.

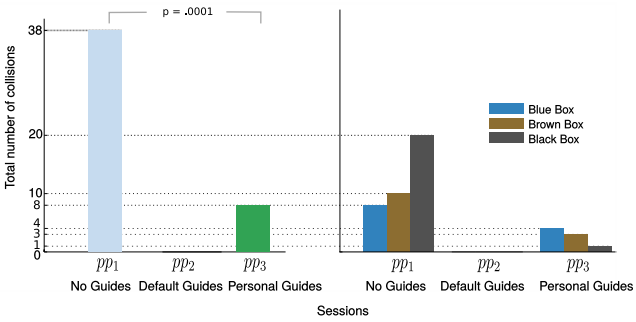


Fig. 16: Left: Total number of collisions by session. Right: Total number of collisions for each box and session.

	$pp1$		$pp2$		$pp3$			
Question	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**	**F(2, 38)**	**P-value**
1) Do you think the task was easy to perform?	5.0	1.45	**5.75**	1.21	**5.85**	1.0	3.6157	**0.03652**
2) Do you think that you performed well the task?	4.6	1.57	**5.8**	0.83	**5.45**	1.1	7.4660	**0.00184**
3) Do you think the robot was helpful during the task execution?	4.3	1.52	**5.45**	1.54	**5.75**	1.02	10.298	**0.00027**
4) You felt comfortable with the robot while performing the task:	5.25	1.55	5.5	1.1	5.65	1.2	0.59262	0.55791
5) You felt stressed to use the robot while performing the task:	2.35	1.35	1.75	0.85	2.3	1.56	1.9334	0.15862
6) Do you think the robot is easy to work with:	4.85	1.46	**5.6**	1.35	**5.55**	1.0	2.6567	**0.08319**
7) Did you feel you had to put physical effort to perform the task:	2.75	1.65	3.25	1.8	2.8	1.36	1.4790	0.24068
8) Did you feel you performed the task precisely?	4.25	1.52	**5.8**	0.89	**5.25**	0.96	13.048	**0.00005**
9) Did you feel constrained by the robot during the experience?	2.95	1.64	4.25	1.77	3.3	1.69	4.4915	0.01774

Table 1: Survey results for the three pick and place sessions.

7.3.3 Survey on Pick and Place

Table 1 shows the results of the survey. From it we can observe the following:

1 The task was perceived as easier to perform when using guides (both with default and personal guides).
2 Users thought that they performed better the task when using guides. Particularly better using the default guides. This can be verified with the collisions (no collisions using default guides, few collisions using personal guides). Moreover, the default guides were more precise since they were created by an expert user, see Fig. 10, while the personal guides were created in a little time by novice users.
3 Participants felt the robot was more helpful to perform the task when using guides. No relevant difference between default and personal guides.
4 Participants felt more comfortable with their own guides. In this case the result is not statistically relevant.
5 Participants felt less stressed when using the default guides, but more stressed when using their own guides. Again, this could be explained with the number collisions occurred during $pp3$. In this case we have a weak relevance.
6 Participants felt that was easier to work with the robot when the guides were active.
7 Participants perceived that they had to put more physical efforts to perform the task with the default guides. This could be explained by the fact that the controller generates a correction when the user tries to move away from the guide. This effect, reduces the naturalness of the switching when multiple guides are used. To solve this, it could be useful to measure or estimate user's external inputs, such as the force or the velocity applied on the end-effector, to facilitate the switch. Another possible reason for this result, is related to the fact that during the experiments some participants did not have a clear vision of where the guides were placed. During the task execution, some participants, instead of moving the robot along the guide, tried to move the robot where they wanted. Even if not statistically relevant, this could be interpreted as a clear evidence that some sort of visualization for the guides is needed.
8 Participants felt that they performed the task more precisely when using guides.
9 The participants felt more constrained when using the default guides. This can be explained by the fact that is easier to feel one's own guides than guides created by another person.

Question	**Mean**	**SD**
1) I believe that creating the new guides was:		
- Intuitive	5.35	1.31
- Comfortable	4.85	1.04
- Physically demanding	3.1	2.02
- Cognitively demanding	3.9	1.71
2) I believe that to perform the task I should use:		
- NO Guides at all	3.15	1.56
- ONE Guide	3.1	1.74
- MULTIPLE Guides	5.75	1.5
3) I believe that the guide(s) I created reflected what I demonstrated	5.7	1.17
4) I believe that the guide(s) I created was(were) precise	5.0	1.25

Table 2: Survey results for the training session.

By looking at the results highlighted in Table 1 for the questions $1, 2, 3, 6, 8$ we can confirm that virtual guide assistance is perceived as useful by the users, which validates H3.

7.3.4 Survey Training

From (Table 2) we can see that participants felt that creating the virtual guides was quite intuitive and comfortable (question 1). For the selected task multiple guides were felt as necessary (question 2). Moreover, participants felt that the guides they created effectively reflected what they demonstrated (question 3) and were enough precise (question 4). With these results we can validate H4.

8 Conclusions

The development of robotics tools such as virtual guides can be very useful to improve human performances in industrial tasks that can not be completely automatized. Robots possess characteristics such as precision, strength and accuracy that can be exploited in co-manipulation tasks by using the virtual guiding assistance. In this paper, we presented a novel way to create virtual guides; we developed an intuitive and easy way to program them through kinesthetic teaching by using Gaussian Mixture Models. Furthermore, the incremental training of GMM enables the user to refine the guides iteratively with the possibility to be assisted by the virtual guide during the refining process. We also defined a controller that allows the user to use multiple virtual guides in parallel, and selects which guide is responsible for the task execution, based on the variance estimated by the GMM. Together, this constitutes a *library* of virtual guides that enables the user to create, modify and use multiple guides. Finally, we studied the utility of virtual guides with an industrial task and concluded that virtual guides improve the human performances in terms of time and collisions, and they can relieve the workload from the user. In future work we will explore the possibility to exploit the uncertainty given by the probabilistic model to adapt the stiffness of the guide (Medina et al, 2012; Calinon et al, 2014) and add a way to visualize the virtual guides in order to increase the user's immersion in the robot's workspace (Rosenberg, 1993).

Acknowledgements This project has received funding from DIGITEO digiteo (www.digiteo.fr)

References

Aarno D, Ekvall S, Kragic D (2005) Adaptive virtual fixtures for machine-assisted teleoperation tasks. In: ICRA, pp 897–903

Abbott JJ (2005) Virtual fixtures for bilateral telemanipulation. PhD thesis, Johns Hopkins University

Abdi H, Williams LJ (2010) Tukeys honestly significant difference (hsd) test. Encyclopedia of Research Design Thousand Oaks, CA: Sage pp 1–5

Amor HB, Neumann G, Kamthe S, Kroemer O, Peters J (2014) Interaction primitives for human-robot cooperation tasks. In: Robotics and Automation (ICRA), 2014 IEEE International Conference on, IEEE, pp 2831–2837

Becker BC, Maclachlan RA, Lobes LA, Hager GD, Riviere CN (2013) Vision-based control of a handheld surgical micromanipulator with virtual fixtures. IEEE Trans Robot 29(3):674–683

Bettini A, Marayong P, Member S, Lang S, Okamura AM, Hager GD (2004) Vision assisted control for manipulation using virtual fixtures. In: International Conference on Intelligent Robots and Systems (IROS), pp 1171–1176

Bowyer SA, y Baena FR (2013) Dynamic frictional constraints for robot assisted surgery. In: World Haptics Conference (WHC), 2013, pp 319–324, DOI 10.1109/WHC.2013.6548428

Bowyer SA, Davies BL, y Baena FR (2014) Active constraints/virtual fixtures: A survey. IEEE Transactions on Robotics 30(1):138–157, DOI 10.1109/TRO.2013.2283410

Boy ES, Burdet E, Teo CL, Colgate JE (2007) Investigation of motion guidance with scooter cobot and collaborative learning. IEEE transactions on robotics 23(2):245–255

Calinon S (2007) Incremental learning of gestures by imitation in a humanoid robot. In: In Proceedings of the 2007 ACM/IEEE International Conference on Human-Robot Interaction, pp 255–262

Calinon S, Guenter F, Billard A (2007) On learning, representing and generalizing a task in a humanoid robot. IEEE Transactions on Systems, Man and Cybernetics, Special issue on robot learning by observation, demonstration and imitation 37(2):286–298

Calinon S, Bruno D, Caldwell DG (2014) A task-parameterized probabilistic model with minimal intervention control. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), Hong Kong, China, pp 3339–3344

Colgate JE, Peshkin MA, Klostermeyer SH (2003) Intelligent assist devices in industrial applications: a review. In: IROS, pp 2516–2521

David O, Russotto FX, Simoes MDS, Measson Y (2014) Collision avoidance, virtual guides and advanced supervisory control teleoperation techniques for high-tech construction: Framework design. Automation in Construction 44:63–72

Davies B, Jakopec M, Harris SJ, Baena FRY, Barrett A, Evangelidis A, Gomes P, Henckel J, Cobb J (2006) Active-constraint robotics for surgery. Proceedings of the IEEE 94(9):1696–1704, DOI 10.1109/JPROC.2006.880680

Dumora J (2014) Contribution à linteraction physique homme-robot: application à la comanipulation dobjets de grandes dimensions. PhD thesis, Montpellier 2

Ewerton M, Maeda G, Kollegger G, Wiemeyer J, Peters J (2016) Incremental imitation learning of context-dependent motor skills. In: Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on, IEEE, pp 351–358

Girden ER (1992) ANOVA: Repeated measures. 84, Sage

Held L, Bov DS (2013) Applied Statistical Inference: Likelihood and Bayes. Springer Publishing Company, Incorporated

Hermann M, Pentek T, Otto B (2016) Design principles for industrie 4.0 scenarios. In: 2016 49th Hawaii Inter-

national Conference on System Sciences (HICSS), IEEE, pp 3928–3937

Ho SC, Hibberd RD, Davies BL (1995) Robot assisted knee surgery. IEEE Engineering in Medicine and Biology Magazine 14(3):292–300, DOI 10.1109/51.391774

Joly L, Andriot C (1995) Imposing motion constraints to a force reflecting telerobot through real-time simulation of a virtual mechanism. In: Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on, vol 1, pp 357–362 vol.1, DOI 10.1109/ROBOT.1995.525310

Kuang A, Payandeh S, Zheng B, Henigman F, MacKenzie C (2004) Assembling virtual fixtures for guidance in training environments. In: Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2004. HAPTICS '04. Proceedings. 12th International Symposium on, pp 367–374, DOI 10.1109/HAPTIC.2004.1287223

Lee D, Ott C (2011) Incremental kinesthetic teaching of motion primitives using the motion refinement tube. Autonomous Robots 31(2-3):115–131

Li M, Okamura AM (2003) Recognition of operator motions for real-time assistance using virtual fixtures. In: In Proc. 11th Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems, pp 125–131

Lin HC, Marayong P, Mills K, Karam R, Kazanzides P, Okamura AM, Hager GD (2006) Portability and applicability of virtual fixtures across medical and manufacturing tasks. In: IEEE International Conference on Robotics and Automation, pp 225–340

Marayong P, Li M, Okamura AM, Hager GD (2003) Spatial motion constraints: theory and demonstrations for robot guidance using virtual fixtures. In: ICRA, IEEE, pp 1954–1959

Medina JR, Lee D, Hirche S (2012) Risk-sensitive optimal feedback control for haptic assistance. In: Robotics and Automation (ICRA), 2012 IEEE International Conference on, IEEE, pp 1025–1031

Mollard Y, Munzer T, Baisero A, Toussaint M, Lopes M (2015) Robot programming from demonstration, feedback and transfer. In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pp 1825–1831, DOI 10.1109/IROS.2015.7353615

Nolin JT, Stemniski PM, Okamura AM (2003) Activation cues and force scaling methods for virtual fixtures. In: in Proc. 11th Int. Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems, pp 404–409

Pezzementi Z, Hager GD, Okamura AM (2007) Dynamic guidance with pseudoadmittance virtual fixtures. In: IEEE International Conference on Robotics and Automation, pp 1761–1767

Raiola G (2017) Co-manipulation with a library of virtual guides. PhD thesis, Université Paris-Saclay

Raiola G, Lamy X, Stulp F (2015a) Co-manipulation with multiple probabilistic virtual guides. In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, IEEE, pp 7–13

Raiola G, Rodriguez-Ayerbe P, Lamy X, Tliba S, Stulp F (2015b) Parallel guiding virtual fixtures: Control and stability. In: Intelligent Control (ISIC), 2015 IEEE International Symposium on, IEEE, pp 53–58

Rosenberg L (1993) Virtual fixtures: perceptual tools for telerobotic manipulation. In: Proc. IEEE Virtual Reality International Sympsoium

Rozo L, Calinon S, Caldwell DG, Jimnez P, Torras C (2016) Learning physical collaborative robot behaviors from human demonstrations. IEEE Transactions on Robotics PP(99):1–15, DOI 10.1109/TRO.2016.2540623

Ryden F, Stewart A, Chizeck H (2013) Advanced telerobotic underwater manipulation using virtual fixtures and haptic rendering. In: Oceans - San Diego, 2013, pp 1–8

Sanchez Restrepo S, Raiola G, Chevalier P, Lamy X, Sidobre D (2017) Iterative virtual guides programming for human-robot comanipulation. In: IEEE International Conference on Advanced Intelligent Mechatronics (AIM)

Vakanski A, Mantegh I, Irish A, Janabi-Sharifi F (2012) Trajectory learning for robot programming by demonstration using hidden markov model and dynamic time warping. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 42(4):1039–1052, DOI 10.1109/TSMCB.2012.2185694

Wrede S, Emmerich C, Grünberg R, Nordmann A, Swadzba A, Steil J (2013) A user study on kinesthetic teaching of redundant robots in task and configuration space. Journal of Human-Robot Interaction 2(1):56–81

Yoon H, Wang R, Hutchinson S (2014) Modeling user's driving-characteristics in a steering task to customize a virtual fixture based on task-performance. In: Robotics and Automation (ICRA), 2014 IEEE International Conference on, pp 625–630, DOI 10.1109/ICRA.2014.6906920

Yu W, Alqasemi R, Dubey R, Pernalete N (2005) Telemanipulation assistance based on motion intention recognition. In: Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, pp 1121–1126, DOI 10.1109/ROBOT.2005.1570266

Bibliography

[Aarno 2005] D. Aarno, S. Ekvall and D. Kragic. *Adaptive Virtual Fixtures for Machine-Assisted Teleoperation Tasks.* In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005, April 2005. (Cited in pages 24, 29, 31, and 35.)

[Abbott 2003] Jake J. Abbott and Allison M. Okamura. *Virtual Fixture Architectures for Telemanipulation*, 2003. (Cited in pages 26, 27, and 33.)

[Abbott 2005] Jake J. Abbott. *Virtual Fixtures for Bilateral Telemanipulation.* PhD thesis, Johns Hopkins University, 2005. (Cited in page 24.)

[Abbott 2006] Jake J Abbott and Allison M Okamura. *Stable forbidden-region virtual fixtures for bilateral telemanipulation.* Journal of dynamic systems, measurement, and control, vol. 128, no. 1, pages 53–64, 2006. (Cited in page 32.)

[Abbott 2007] Jake J Abbott and Allison M Okamura. *Pseudo-admittance bilateral telemanipulation with guidance virtual fixtures.* The International Journal of Robotics Research, vol. 26, no. 8, pages 865–884, 2007. (Cited in page 32.)

[Akima 1970] Hiroshi Akima. *A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures.* J. ACM, October 1970. (Cited in pages 57, 60, 61, and 62.)

[Alissandrakis 2007] Aris Alissandrakis, Chrystopher L Nehaniv and Kerstin Dautenhahn. *Correspondence mapping induced state and action metrics for robotic imitation.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 37, no. 2, pages 299–307, 2007. (Cited in page 18.)

[Ammi 2007] Mehdi Ammi and Antoine Ferreira. *Robotic assisted micromanipulation system using virtual fixtures and metaphors.* In Robotics and Automation, 2007 IEEE International Conference on, pages 454–460. IEEE, 2007. (Cited in pages 32 and 34.)

[Becker 2013] B. C. Becker, R. A. MacLachlan, L. A. Lobes, G. D. Hager and C. N. Riviere. *Vision-Based Control of a Handheld Surgical Micromanipulator*

With Virtual Fixtures. IEEE Transactions on Robotics, June 2013. (Cited in pages 21 and 25.)

[Benjamini 1995] Yoav Benjamini and Yosef Hochberg. *Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing*. Journal of the Royal Statistical Society. Series B (Methodological), vol. 57, no. 1, pages 289–300, 1995. (Cited in pages 106, 112, and 127.)

[Bettini 2001] A Bettini, S Lang, A Okamura and G Hager. *Vision assisted control for manipulation using virtual fixtures*. In Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on, volume 2, pages 1171–1176. IEEE, 2001. (Cited in pages 29, 34, and 65.)

[Bettini 2004] Alessandro Bettini, Panadda Marayong, Student Member, Samuel Lang, Allison M. Okamura and Gregory D. Hager. *Vision Assisted Control for Manipulation Using Virtual Fixtures*. In International Conference on Intelligent Robots and Systems (IROS), pages 1171–1176, 2004. (Cited in pages 27 and 29.)

[Biggs 2003] Geoffrey Biggs and Bruce MacDonald. *A survey of robot programming systems*. In Proceedings of the Australasian conference on robotics and automation, pages 1–3, 2003. (Cited in page 15.)

[Billard 1999] Aude Billard and Gillian Hayes. *Drama, a connectionist architecture for control and learning in autonomous robots*. Adaptive Behavior, vol. 7, no. 1, pages 35–63, 1999. (Cited in page 17.)

[Billard 2008] Aude Billard, Sylvain Calinon, Ruediger Dillmann and Stefan Schaal. *Robot programming by demonstration*. In Springer handbook of robotics, pages 1371–1394. Springer, 2008. (Cited in pages 16 and 19.)

[Billard 2016] Aude G Billard, Sylvain Calinon and Rüdiger Dillmann. *Learning from humans*. In Springer Handbook of Robotics, pages 1995–2014. Springer, 2016. (Cited in page 20.)

[Bogue 2009] Robert Bogue. *Exoskeletons and robotic prosthetics: a review of recent developments*. Industrial Robot: An International Journal, vol. 36, no. 5, pages 421–427, 2009. (Cited in page 15.)

[Book 1996] Wayne John Book, Robert Charles, Hurley T. Davis and Mario Waldorff Gomes. *The Concept and Implementation of a Passive Trajectory Enhancing Robot*, November 1996. (Cited in page 21.)

[Borrel 1994] Paul Borrel and Ari Rappoport. *Simple Constrained Deformations for Geometric Modeling and Interactive Design*. ACM Trans. Graph., April 1994. (Cited in page 96.)

[Bowyer 2013] S. A. Bowyer and F. Rodriguez y Baena. *Dynamic frictional constraints for robot assisted surgery.* In World Haptics Conference (WHC), 2013, pages 319–324, April 2013. (Cited in pages 24 and 65.)

[Bowyer 2014a] S. A. Bowyer, B. L. Davies and F. Rodriguez y Baena. *Active Constraints/Virtual Fixtures: A Survey.* IEEE Transactions on Robotics, vol. 30, no. 1, pages 138–157, Feb 2014. (Cited in pages 24 and 32.)

[Bowyer 2014b] Stuart A Bowyer and Ferdinando Rodriguez y Baena. *Dynamic frictional constraints in translation and rotation.* In Robotics and Automation (ICRA), 2014 IEEE International Conference on, pages 2685–2692. IEEE, 2014. (Cited in page 65.)

[Bowyer 2015] Stuart A Bowyer and Ferdinando Rodriguez y Baena. *Dissipative Control for Physical Human–Robot Interaction.* IEEE Transactions on Robotics, vol. 31, no. 6, pages 1281–1293, 2015. (Cited in page 66.)

[Boy 2007] Eng Seng Boy, E. Burdet, Chee Leong Teo and J.E. Colgate. *Investigation of Motion Guidance With Scooter Cobot and Collaborative Learning.* IEEE Transactions on Robotics, April 2007. (Cited in page 36.)

[Burghart 1999a] Catherina Burghart, Jochen Keitel, Stefan Hassfeld, Ulrich Rembold and Heinz Woern. *Robot Controlled Osteotomy in Craniofacial Surgery*, 1999. (Cited in page 27.)

[Burghart 1999b] Catherina Burghart, Jochen Keitel, Stefan Hassfeld, Ulrich Rembold and Heinz Woern. *Robot controlled osteotomy in craniofacial surgery.* In Proceedings of the 1st International Workshop on Haptic Devices in Medical Applications, Paris, 1999. (Cited in pages 29 and 34.)

[Burschka 2005] Darius Burschka, Jason J Corso, Maneesh Dewan, William Lau, Ming Li, Henry Lin, Panadda Marayong, Nicholas Ramey, Gregory D Hager, Brian Hoffman*et al. Navigating inner space: 3-d assistance for minimally invasive surgery.* Robotics and Autonomous Systems, vol. 52, no. 1, pages 5–26, 2005. (Cited in pages 27 and 29.)

[Cagneau 2008] Barthelemy Cagneau, Guillaume Morel, Delphine Bellot, Nabil Zemiti and Ginluca A d'Agostino. *A passive force amplifier.* In Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pages 2079–2084. IEEE, 2008. (Cited in page 15.)

[Calinon 2007a] S. Calinon, F. Guenter and A. Billard. *On Learning, Representing, and Generalizing a Task in a Humanoid Robot.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), April 2007. (Cited in pages 17, 18, and 54.)

[Calinon 2007b] Sylvain Calinon and Aude G Billard. *What is the teacher's role in robot programming by demonstration?: Toward benchmarks for improved learning.* Interaction Studies, vol. 8, no. 3, pages 441–464, 2007. (Cited in page 35.)

[Calinon 2008] S. Calinon and A. Billard. *A probabilistic Programming by Demonstration framework handling constraints in joint space and task space.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008, September 2008. (Cited in page 30.)

[Calinon 2010] S. Calinon, F. D'halluin, E.L. Sauser, D.G. Caldwell and A.G. Billard. *Learning and Reproduction of Gestures by Imitation.* IEEE Robotics Automation Magazine, June 2010. (Cited in page 30.)

[Calinon 2011] Sylvain Calinon, Antonio Pistillo and Darwin G Caldwell. *Encoding the time and space constraints of a task in explicit-duration hidden Markov model.* In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pages 3413–3418. IEEE, 2011. (Cited in page 17.)

[Calinon 2013] Sylvain Calinon, Tohid Alizadeh and Darwin G Caldwell. *On improving the extrapolation capability of task-parameterized movement models.* In Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pages 610–616. IEEE, 2013. (Cited in page 20.)

[Calinon 2014] Sylvain Calinon, Danilo Bruno and Darwin G Caldwell. *A task-parameterized probabilistic model with minimal intervention control.* In Robotics and Automation (ICRA), 2014 IEEE International Conference on, pages 3339–3344. IEEE, 2014. (Cited in page 20.)

[Calinon 2018] Sylvain Calinon. *Robot learning with task-parameterized generative models.* In Robotics Research, pages 111–126. Springer, 2018. (Cited in page 20.)

[Castillo-Cruces 2010] Raúl A Castillo-Cruces and Jürgen Wahrburg. *Virtual fixtures with autonomous error compensation for human–robot cooperative tasks.* Robotica, vol. 28, no. 2, pages 267–277, 2010. (Cited in page 65.)

[Colgate 1988] J. Edward (James Edward) Colgate. *The control of dynamically interacting systems.* Thesis, Massachusetts Institute of Technology, 1988. (Cited in page 50.)

[Colgate 1996] J Edward Colgate, J Edward, Michael A Peshkin and Witaya Wannasuphoprasit. *Cobots: Robots for collaboration with human operators.* Citeseer, 1996. (Cited in page 11.)

[Colgate 2003] J.E. Colgate, M. Peshkin and S.H. Klostermeyer. *Intelligent assist devices in industrial applications: a review.* In 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings, October 2003. (Cited in pages 21 and 24.)

[Cretu 2003] A-M Cretu, Emil M Petriu and Gilles G Patry. *Neural network architecture for 3D object representation.* In Haptic, Audio and Visual Environments and Their Applications, 2003. HAVE 2003. Proceedings. The 2nd IEEE Internatioal Workshop on, pages 31–36. IEEE, 2003. (Cited in page 30.)

[Dam 1998] Erik B Dam, Martin Koch and Martin Lillholm. Quaternions, interpolation and animation, volume 2. Datalogisk Institut, Københavns Universitet, 1998. (Cited in pages 45, 71, 72, 73, 75, and 78.)

[David 2014] Olivier David, François-Xavier Russotto, Max Da Silva Simoes and Yvan Measson. *Collision avoidance, virtual guides and advanced supervisory control teleoperation techniques for high-tech construction: framework design.* Automation in Construction, August 2014. (Cited in pages 24 and 25.)

[Davies 2006] B. Davies, M. Jakopec, S. J. Harris, F. Rodriguez Y Baena, A. Barrett, A. Evangelidis, P. Gomes, J. Henckel and J. Cobb. *Active-Constraint Robotics for Surgery.* Proceedings of the IEEE, vol. 94, no. 9, pages 1696–1704, Sept 2006. (Cited in page 24.)

[De Boor 1972] Carl De Boor. *On calculating with B-splines.* Journal of Approximation theory, vol. 6, no. 1, pages 50–62, 1972. (Cited in page 56.)

[Dumora 2014] Julie Dumora. *Contribution à l'interaction physique homme-robot : Application à la comanipulation d'objets de grandes dimensions.* PhD thesis, Université Montpellier 2, March 2014. (Cited in pages 9, 24, 25, 29, 86, and 88.)

[Eberly 2002] David Eberly. *Quaternion algebra and calculus.* Magic Software Inc, 2002. (Cited in pages 72, 73, 74, and 75.)

[Ekvall 2006] Staffan Ekvall and Danica Kragic. *Learning task models from multiple human demonstrations.* In Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on, pages 358–363. IEEE, 2006. (Cited in page 18.)

[Erden 2011] Mustafa Suphi Erden and Bobby Marić. *Assisting manual welding with robot.* Robotics and Computer-Integrated Manufacturing, vol. 27, no. 4, pages 818–828, 2011. (Cited in page 15.)

[Erkorkmaz 2005] Kaan Erkorkmaz and Yusuf Altintas. *Quintic spline interpolation with minimal feed fluctuation.* Transactions of the ASME-B-Journal of

Manufacturing Science and Engineering, vol. 127, no. 2, pages 339–349, 2005. (Cited in page 153.)

[Farin 2014] Gerald Farin. Curves and surfaces for computer-aided geometric design: a practical guide. Elsevier, 2014. (Cited in page 73.)

[Fitts 1951] Paul M Fitts. *Human engineering for an effective air-navigation and traffic-control system.* 1951. (Cited in page 86.)

[Françoise 2013a] Vincent Françoise. *Assistance au fraisage par comanipulation en chirurgie orthopédique.* PhD thesis, Paris 6, 2013. (Cited in page 7.)

[Françoise 2013b] Vincent Françoise. *Assistance au fraisage par comanipulation en chirurgie orthopédique.* PhD thesis, Paris 6, 2013. (Cited in page 27.)

[Fritsch 1980] Frederick N Fritsch and Ralph E Carlson. *Monotone piecewise cubic interpolation.* SIAM Journal on Numerical Analysis, vol. 17, no. 2, pages 238–246, 1980. (Cited in pages 57, 63, and 77.)

[Funda 1990] Janez Funda, Russell H Taylor and Richard P Paul. *On homogeneous transforms, quaternions, and computational efficiency.* IEEE Transactions on Robotics and Automation, vol. 6, no. 3, pages 382–388, 1990. (Cited in page 70.)

[Gain 2008] James Gain and Dominique Bechmann. *A survey of spatial deformation from a user-centered perspective.* ACM Transactions on Graphics (TOG), vol. 27, no. 4, page 107, 2008. (Cited in page 95.)

[Garrec 2010] Phillipe Garrec. *Screw and Cable Acutators (SCS) and Their Applications to Force Feedback Teleoperation, Exoskeleton and Anthropomorphic Robotics.* In Robotics 2010 Current and Future Challenges. InTech, 2010. (Cited in pages 10 and 104.)

[Ghanbari 2010] Ali Ghanbari, Hamid Abdi, Ben Horan, Saeid Nahavandi, Xiaoqi Chen and Wenhui Wang. *Haptic guidance for microrobotic intracellular injection.* In Biomedical Robotics and Biomechatronics (BioRob), 2010 3rd IEEE RAS and EMBS International Conference on, pages 162–167. IEEE, 2010. (Cited in page 34.)

[Gibo 2009] Tricia L Gibo, Lawton N Verner, David D Yuh and Allison M Okamura. *Design considerations and human-machine performance of moving virtual fixtures.* In Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, pages 671–676. IEEE, 2009. (Cited in page 32.)

[Gorecky 2014] Dominic Gorecky, Mathias Schmitt, Matthias Loskyll and Detlef Zühlke. *Human-machine-interaction in the industry 4.0 era.* In Industrial Informatics (INDIN), 2014 12th IEEE International Conference on, pages 289–294. IEEE, 2014. (Cited in page 8.)

[Groshaw 1969] PF Groshaw. *General Electric Co.: Hardiman I Arm Test, Hardiman I Prototype, General Electric Rep.* Technical Report, S-70-1019, Schenectady, 1969. (Cited in page 15.)

[Guerry 2017] Joris Guerry, Bertrand Le Saux and David Filliat. *"Look At This One" Detection sharing between modality-independent classifiers for robotic discovery of people.* In ECMR 2017-European Conference on Mobile Robotics, pages 1–6, 2017. (Cited in page 6.)

[Hanson 2005] Andrew J Hanson. *Visualizing quaternions.* In ACM SIGGRAPH 2005 Courses, page 1. ACM, 2005. (Cited in page 71.)

[Ho 1995] S. C. Ho, R. D. Hibberd and B. L. Davies. *Robot assisted knee surgery.* IEEE Engineering in Medicine and Biology Magazine, vol. 14, no. 3, pages 292–300, May 1995. (Cited in page 32.)

[Hogan 1988] N. Hogan. *On the stability of manipulators performing contact tasks.* IEEE Journal on Robotics and Automation, December 1988. (Cited in page 50.)

[Hogan 1989] N. Hogan. *Controlling impedance at the man/machine interface.* In IEEE International Conference on Robotics and Automation, 1989. (Cited in pages 26, 50, and 87.)

[Hsu 1992] William M Hsu, John F Hughes and Henry Kaufman. *Direct manipulation of free-form deformations.* In ACM Siggraph Computer Graphics, volume 26, pages 177–184. ACM, 1992. (Cited in page 96.)

[Ijspeert 2003] AJ Ijspeert, Jun Nakanishi and Stefan Schaal. *Learning control policies for movement imitation and movement recognition.* In Neural information processing system, volume 15, pages 1547–1554, 2003. (Cited in page 19.)

[Ikits 2003] Milan Ikits, J Dean Brederson, Charles D Hansen and Christopher R Johnson. *A constraint-based technique for haptic volume exploration.* In Visualization, 2003. VIS 2003. IEEE, pages 263–269. IEEE, 2003. (Cited in page 29.)

[Inamura 2006] Tetsunari Inamura, Naoki Kojo and Masayuki Inaba. *Situation recognition and behavior induction based on geometric symbol representation of multimodal sensorimotor patterns.* In Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, pages 5147–5152. IEEE, 2006. (Cited in page 16.)

[Joli 2007] Pierre Joli, Shahram Payandeh, M Chan and Benjamin Bayart. *A new approach to solve constraint forces of virtual fixtures in haptic rendering.* In Proc. 12th World Congr. Mech. Mach. Sci. Conj. IFToMM, 2007. (Cited in page 32.)

[Joly 1995] L.D. Joly and C. Andriot. *Imposing motion constraints to a force reflecting telerobot through real-time simulation of a virtual mechanism.* In , 1995 IEEE International Conference on Robotics and Automation, 1995. Proceedings, May 1995. (Cited in pages 24, 27, 29, 32, 33, 43, 50, 52, and 53.)

[Joly 1997] Luc Joly. *Commande hybride position/force pour la teleoperation: une approche basée sur des analogies mécaniques.* PhD thesis, Paris 6, 1997. (Cited in pages 22, 27, 34, 43, 51, and 58.)

[Juhász 2001] Imre Juhász and Miklós Hoffmann. *The effect of knot modifications on the shape of B-spline curves.* Journal for Geometry and Graphics, vol. 5, no. 2, pages 111–119, 2001. (Cited in page 95.)

[Kaiser 1996] Michael Kaiser and Rüdiger Dillmann. *Building elementary robot skills from human demonstration.* In Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on, volume 3, pages 2700–2705. IEEE, 1996. (Cited in page 17.)

[Kang 1995] Sing Bing Kang and Katsushi Ikeuchi. *A robot system that observes and replicates grasping tasks.* In Computer Vision, 1995. Proceedings., Fifth International Conference on, pages 1093–1099. IEEE, 1995. (Cited in page 16.)

[Kapoor 2006] Ankur Kapoor, Ming Li and Russell H Taylor. *Constrained Control for Surgical Assistant Robots.* In ICRA, pages 231–236, 2006. (Cited in page 29.)

[Kazerooni 1993] Homayoon Kazerooni. *Extender: a case study for human-robot interaction via transfer of power and information signals.* In Robot and Human Communication, 1993. Proceedings., 2nd IEEE International Workshop on, pages 10–20. IEEE, 1993. (Cited in page 15.)

[Kazerooni 2001] Homayoon Kazerooni. *Human power amplifier for vertical maneuvers*, October 9 2001. US Patent 6,299,139. (Cited in page 12.)

[Khalil 1996] Hassan K Khalil and JW Grizzle. Nonlinear systems, volume 3. Prentice hall New Jersey, 1996. (Cited in page 50.)

[Khatib 1986] Oussama Khatib. *Real-time obstacle avoidance for manipulators and mobile robots.* The international journal of robotics research, vol. 5, no. 1, pages 90–98, 1986. (Cited in page 33.)

[Kikuuwe 2006] Ryo Kikuuwe, Naoyuki Takesue, Akihito Sano, Hiromi Mochiyama and Hideo Fujimoto. *Admittance and impedance representations of friction based on implicit Euler integration.* IEEE Transactions on Robotics, vol. 22, no. 6, pages 1176–1188, 2006. (Cited in page 33.)

[Kikuuwe 2008] Ryo Kikuuwe, Naoyuki Takesue and Hideo Fujimoto. *A control framework to generate nonenergy-storing virtual fixtures: Use of simulated plasticity*. IEEE Transactions on Robotics, vol. 24, no. 4, pages 781–793, 2008. (Cited in page 33.)

[Kornely 1989] Michael Kornely. *Self balancing electric hoist*, February 28 1989. US Patent 4,807,767. (Cited in page 12.)

[Kosuge 1995] K. Kosuge, T. Itoh, T. Fukuda and M. Otsuka. *Tele-manipulation system based on task-oriented virtual tool*. In , 1995 IEEE International Conference on Robotics and Automation, 1995. Proceedings, May 1995. (Cited in pages 27 and 36.)

[Kuang 2004] A.B. Kuang, S. Payandeh, Bin Zheng, F. Henigman and C.L. MacKenzie. *Assembling virtual fixtures for guidance in training environments*. In Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2004. HAPTICS '04. Proceedings. 12th International Symposium on, pages 367–374, March 2004. (Cited in pages 29, 30, and 65.)

[Kumar 2000] Rajesh Kumar, Peter Berkelman, Puneet Gupta, Aaron Barnes, Patrick S Jensen, Louis L Whitcomb and Russell H Taylor. *Preliminary experiments in cooperative human/robot force control for robot assisted microsurgical manipulation*. In Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on, volume 1, pages 610–617. IEEE, 2000. (Cited in page 15.)

[Kumar 2003] Rajesh Kumar, Ankur Kapoor and Russell H Taylor. *Preliminary experiments in robot/human cooperative microinjection*. In Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, volume 4, pages 3186–3191. IEEE, 2003. (Cited in page 28.)

[Kwok 2013] Ka-Wai Kwok, Kuen Hung Tsoi, Valentina Vitiello, James Clark, Gary CT Chow, Wayne Luk and Guang-Zhong Yang. *Dimensionality reduction in controlling articulated snake robot for endoscopy under dynamic active constraints*. IEEE Transactions on Robotics, vol. 29, no. 1, pages 15–31, 2013. (Cited in page 29.)

[Lamy 2011] Xavier Lamy. *Conception d'une interface de pilotage d'un Cobot*. PhD thesis, Paris 6, January 2011. (Cited in pages 15 and 25.)

[Lee 1996] Christopher Lee and Yangsheng Xu. *Online, interactive learning of gestures for human/robot interfaces*. In Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on, volume 4, pages 2982–2987. IEEE, 1996. (Cited in page 17.)

[Lee 2006] Kye-Young Lee, Seung-Yeol Lee, Jong-Ho Choi, Sang-Heon Lee and Chang-Soo Han. *The application of the human-robot cooperative system for*

construction robot manipulating and installing heavy materials. In SICE-ICASE, 2006. International Joint Conference, pages 4798–4802. IEEE, 2006. (Cited in pages 9 and 15.)

[Lee 2007] Dongheui Lee and Yoshihiko Nakamura. *Mimesis scheme using a monocular vision system on a humanoid robot.* In Robotics and Automation, 2007 IEEE International Conference on, pages 2162–2168. IEEE, 2007. (Cited in page 17.)

[Li 2003] Ming Li and Russell H Taylor. *Optimum robot control for 3D virtual fixture in constrained ENT surgery.* In International Conference on Medical Image Computing and Computer-Assisted Intervention, pages 165–172. Springer, 2003. (Cited in page 28.)

[Li 2005a] Ming Li, Ankur Kapoor and Russell H Taylor. *A constrained optimization approach to virtual fixtures.* In Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on, pages 1408–1413. IEEE, 2005. (Cited in page 27.)

[Li 2005b] Ming Li and Russell H Taylor. *Performance of surgical robots with automatically generated spatial virtual fixtures.* In Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, pages 217–222. IEEE, 2005. (Cited in page 29.)

[Li 2007] Ming Li, Masaru Ishii and Russell H Taylor. *Spatial motion constraints using virtual fixtures generated by anatomy.* IEEE Transactions on Robotics, vol. 23, no. 1, pages 4–19, 2007. (Cited in pages 29, 65, and 66.)

[Lin 2006] H. C. Lin, K. Mills, P. Kazanzides, G. D. Hager, P. Marayong, A. M. Okamura and R. Karam. *Portability and applicability of virtual fixtures across medical and manufacturing tasks.* In Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., pages 225–230, May 2006. (Cited in page 24.)

[Lomax 2013] Richard G Lomax and Debbie L Hahs-Vaughn. Statistical concepts: A second course. Routledge, 2013. (Cited in page 106.)

[Lucke 2008] Dominik Lucke, Carmen Constantinescu and Engelbert Westkämper. *Smart factory-a step towards the next generation of manufacturing.* Manufacturing systems and technologies for the new frontier, pages 115–118, 2008. (Cited in page 5.)

[Makarov 2013] Maria Makarov. *Contribution à la modélisation et la commande robuste de robots manipulateurs à articulations flexibles. Applications à la robotique interactive.* PhD thesis, Supélec, 2013. (Cited in page 25.)

[Marayong 2003] Panadda Marayong, Ming Li, Allison M. Okamura and Gregory D. Hager. *Spatial motion constraints: theory and demonstrations for robot guidance using virtual fixtures.* In ICRA, pages 1954–1959. IEEE, 2003. (Cited in pages 27, 29, and 34.)

[Marayong 2004] Panadda Marayong and Allison M Okamura. *Speed-accuracy characteristics of human-machine cooperative manipulation using virtual fixtures with variable admittance.* Human Factors, vol. 46, no. 3, pages 518–532, 2004. (Cited in pages 27, 29, and 91.)

[Maurice 2017] Pauline Maurice, Vincent Padois, Yvan Measson and Philippe Bidaud. *Human-oriented design of collaborative robots.* International Journal of Industrial Ergonomics, vol. 57, pages 88–102, 2017. (Cited in pages 25, 146, and 158.)

[Micaelli 1998] Alain Micaelli, Catherine Bidard and Claude Andriot. *Decoupling control based on virtual mechanisms for telemanipulation.* In Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on, volume 3, pages 1924–1931. IEEE, 1998. (Cited in page 33.)

[Mitchell 2007] Ben Mitchell, John Koo, Iulian Iordachita, Peter Kazanzides, Ankur Kapoor, James Handa, Gregory Hager and Russell Taylor. *Development and application of a new steady-hand manipulator for retinal surgery.* In Robotics and Automation, 2007 IEEE International Conference on, pages 623–629. IEEE, 2007. (Cited in page 15.)

[Mollard 2015] Y. Mollard, T. Munzer, A. Baisero, M. Toussaint and M. Lopes. *Robot programming from demonstration, feedback and transfer.* In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), September 2015. (Cited in page 36.)

[Morel 2012] Guillaume Morel, J. Szewczyk and M.A. Vitrani. *Comanipulation.* J. Troccaz, editeur, Robotique Medicale, page 343–392, 2012. (Cited in page 11.)

[Muench 1994] S Muench, J Kreuziger, M Kaiser and R Dillman. *Robot programming by demonstration (rpd)-using machine learning and user interaction methods for the development of easy and comfortable robot programming systems.* In Proceedings of the International Symposium on Industrial Robots, volume 25, pages 685–685. INTERNATIONAL FEDERATION OF ROBOTICS, & ROBOTIC INDUSTRIES, 1994. (Cited in page 16.)

[Mussa-Ivaldi F. A. 1985] Bizzi E. Mussa-Ivaldi F. A. Hogan N. *Neural, mechanical, and geometric factors subserving arm posture in humans.* The Journal of neuroscience : the official journal of the Society for Neuroscience, 1985. (Cited in page 87.)

[Navkar 2012] Nikhil V Navkar, Zhigang Deng, Dipan J Shah, Kostas E Bekris and Nikolaos V Tsekos. *Visual and force-feedback guidance for robot-assisted interventions in the beating heart with real-time MRI.* In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 689–694. IEEE, 2012. (Cited in page 28.)

[Newman 1994] Wyatt S Newman and Yuandao Zhang. *Stable interaction control and coulomb friction compensation using natural admittance control.* Journal of Field Robotics, vol. 11, no. 1, pages 3–11, 1994. (Cited in page 10.)

[Nolin 2003] Jason T. Nolin, Paul M. Stemniski and Allison M. Okamura. *Activation Cues and Force Scaling Methods for Virtual Fixtures.* In in Proc. 11th Int. Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems, pages 404–409, 2003. (Cited in pages 65, 91, 92, and 93.)

[Ortmaier 2006] Tobias Ortmaier, Holger Weiß, Ulrich Hagn, Markus Grebenstein, Mathias Nickl, A Albu-Schaffer, Christian Ott, S Jorg, Rainer Konietschke, Luc Le-Tien*et al. A hands-on-robot for accurate placement of pedicle screws.* In Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, pages 4179–4186. IEEE, 2006. (Cited in page 32.)

[Pardowitz 2007] Michael Pardowitz, Steffen Knoop, Ruediger Dillmann and Raoul D Zollner. *Incremental learning of tasks from user demonstrations, past experiences, and vocal comments.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 37, no. 2, pages 322–332, 2007. (Cited in page 18.)

[Park 2001] Shinsuk Park, Robert D. Howe and David F. Torchiana. *Virtual Fixtures for Robotic Cardiac Surgery.* In Medical Image Computing and Computer-Assisted Intervention – MICCAI 2001. Springer Berlin Heidelberg, October 2001. (Cited in page 21.)

[Park 2011] Jun Woo Park, Jaesoon Choi, Yongdoo Park and Kyung Sun. *Haptic virtual fixture for robotic cardiac catheter navigation.* Artificial organs, vol. 35, no. 11, pages 1127–1131, 2011. (Cited in page 28.)

[Passenberg 2011] Carolina Passenberg, Raphaela Groten, Angelika Peer and Martin Buss. *Towards real-time haptic assistance adaptation optimizing task performance and human effort.* In World Haptics Conference (WHC), 2011 IEEE, pages 155–160. IEEE, 2011. (Cited in page 91.)

[Payandeh 2002] Shahram Payandeh and Zoran Stanisic. *On application of virtual fixtures as an aid for telemanipulation and training.* In Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002. HAPTICS 2002. Proceedings. 10th Symposium on, pages 18–23. IEEE, 2002. (Cited in page 27.)

[Peshkin 2001] M.A. Peshkin, J.E. Colgate, W. Wannasuphoprasit, C.A. Moore, R.B. Gillespie and P. Akella. *Cobot architecture*. IEEE Transactions on Robotics and Automation, August 2001. (Cited in pages 12, 13, and 21.)

[Pezzementi 2007a] Z. Pezzementi, G. D. Hager and A. M. Okamura. *Dynamic Guidance with Pseudoadmittance Virtual Fixtures*. In IEEE International Conference on Robotics and Automation, pages 1761–1767, 2007. (Cited in pages 25 and 27.)

[Pezzementi 2007b] Zachary Pezzementi, Allison M Okamura and Gregory D Hager. *Dynamic guidance with pseudoadmittance virtual fixtures*. In Robotics and Automation, 2007 IEEE International Conference on, pages 1761–1767. IEEE, 2007. (Cited in page 33.)

[Powell 1969] Edgar R Powell. *Tool balancer*, February 18 1969. US Patent 3,428,298. (Cited in page 12.)

[Prada 2005] Rodolfo Prada and Shahram Payandeh. *A study on design and analysis of virtual fixtures for cutting in training environments*. In Eurohaptics Conference, 2005 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2005. World Haptics 2005. First Joint, pages 375–380. IEEE, 2005. (Cited in pages 29 and 65.)

[Prada 2009] Rodolfo Prada and Shahram Payandeh. *On study of design and implementation of virtual fixtures*. Virtual reality, vol. 13, no. 2, pages 117–129, 2009. (Cited in pages 30, 32, and 34.)

[Pratt 1995] Gill A Pratt and Matthew M Williamson. *Series elastic actuators*. In Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on, volume 1, pages 399–406. IEEE, 1995. (Cited in page 49.)

[Raffin 2000] Romain Raffin, Marc Neveu and Frédéric Jaar. *Curvilinear displacement of free-form-based deformation*. The Visual Computer, vol. 16, no. 1, pages 38–46, 2000. (Cited in pages 96, 97, and 98.)

[Raibert 1981] Marc H Raibert and John J Craig. *Hybrid position/force control of manipulators*. Journal of Dynamic Systems, Measurement, and Control, vol. 102, no. 127, pages 126–133, 1981. (Cited in page 21.)

[Raiola 2015] G. Raiola, X. Lamy and F. Stulp. *Co-manipulation with multiple probabilistic virtual guides*. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), September 2015. (Cited in pages 17 and 92.)

[Raiola 2017a] G. Raiola, S. Sánchez Restrepo, P. Chevalier, P. Rodriguez-Ayerbe, X. Lamy, S. Tliba and F. Stulp. *Co-manipulation with a Library of Virtual*

Guiding Fixtures. Autonomous Robots, Special issue on Learning for Human-Robot Collaboration, 2017. (Cited in pages 26, 31, 36, 54, 65, 154, 155, and 159.)

[Raiola 2017b] Gennaro Raiola. *Co-manipulation with a library of virtual guides*. PhD thesis, Université Paris-Saclay, 2017. (Cited in page 92.)

[Reed 2008] Kyle B Reed and Michael A Peshkin. *Physical collaboration of human-human and human-robot teams*. IEEE Transactions on Haptics, vol. 1, no. 2, pages 108–120, 2008. (Cited in page 86.)

[Reinkensmeyer 2004] David J. Reinkensmeyer, Jeremy L. Emken and Steven C. Cramer. *Robotics, motor learning, and neurologic recovery*. Annual Review of Biomedical Engineering, 2004. (Cited in page 21.)

[Ren 2008] Jing Ren, Rajni V Patel, Kenneth A McIsaac, Gerard Guiraudon and Terry M Peters. *Dynamic 3-D virtual fixtures for minimally invasive beating heart procedures*. IEEE transactions on medical imaging, vol. 27, no. 8, pages 1061–1070, 2008. (Cited in pages 29 and 30.)

[Restrepo 2017] Susana Sánchez Restrepo, Gennaro Raiola, Pauline Chevalier, Xavier Lamy and Daniel Sidobre. *Iterative virtual guides programming for human-robot comanipulation*. In Advanced Intelligent Mechatronics (AIM), 2017 IEEE International Conference on, pages 219–226. IEEE, 2017. (Cited in pages 25, 26, and 29.)

[Rosenberg 1993] L.B. Rosenberg. *Virtual fixtures: Perceptual tools for telerobotic manipulation*. In , 1993 IEEE Virtual Reality Annual International Symposium, 1993, September 1993. (Cited in pages 21, 24, 27, 29, 155, and 156.)

[Rozo 2014] L. Rozo, S. Calinon and D.G. Caldwell. *Learning force and position constraints in human-robot cooperative transportation*. In 2014 RO-MAN: The 23rd IEEE International Symposium on Robot and Human Interactive Communication, August 2014. (Cited in pages 9, 30, 35, and 153.)

[Rydén 2012] Fredrik Rydén and Howard Jay Chizeck. *Forbidden-region virtual fixtures from streaming point clouds: Remotely touching and protecting a beating heart*. In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pages 3308–3313. IEEE, 2012. (Cited in page 29.)

[Ryden 2013] F. Ryden, A. Stewart and H. J. Chizeck. *Advanced telerobotic underwater manipulation using virtual fixtures and haptic rendering*. In Oceans - San Diego, 2013, September 2013. (Cited in pages 21 and 24.)

[Santos-Munne 2010] J.J. Santos-Munne, M. Peshkin, E.L. Faulring, J.E. Colgate, A. Makhlin, T. Moyer, T. Hauptman and W. Hoffmann. *An apparatus for use in breaking down an animal carcass*, March 11 2010. WO Patent App. PCT/AU2009/001,164. (Cited in page 15.)

[Saunders 2006] Joe Saunders, Chrystopher L Nehaniv and Kerstin Dautenhahn. *Teaching robots by moulding behavior and scaffolding the environment.* In Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction, pages 118–125. ACM, 2006. (Cited in page 35.)

[Sederberg 1986] Thomas W Sederberg and Scott R Parry. *Free-form deformation of solid geometric models.* ACM SIGGRAPH computer graphics, vol. 20, no. 4, pages 151–160, 1986. (Cited in page 96.)

[Segre 1985] Alberto Segre and Gerald DeJong. *Explanation-based manipulator learning: Acquisition of planning ability through observation.* In Robotics and Automation. Proceedings. 1985 IEEE International Conference on, volume 2, pages 555–560. IEEE, 1985. (Cited in page 16.)

[Seung 2010] Sungmin Seung, Byungjeon Kang, Wooyoung Kim, Jongoh Park and Sukho Park. *Development of image guided master-slave system for minimal invasive brain surgery.* In Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK), pages 1–6. VDE, 2010. (Cited in pages 29 and 32.)

[Shima 2009] Tal Shima and Steven Rasmussen. Uav cooperative decision and control: challenges and practical approaches. SIAM, 2009. (Cited in page 9.)

[Shoemake 1985] Ken Shoemake. *Animating rotation with quaternion curves.* In ACM SIGGRAPH computer graphics, volume 19, pages 245–254. ACM, 1985. (Cited in pages 68, 71, and 73.)

[Siciliano 2009] B. Siciliano, L. Sciavicco, L. Villani and G. Oriolo. Robotics: Modelling, planning and control. Advanced Textbooks in Control and Signal Processing. Springer, 2009. (Cited in page 49.)

[Srimathveeravalli 2007] Govindarajan Srimathveeravalli, Venkatraghavan Gourishankar and Thenkurussi Kesavadas. *Comparative study: Virtual fixtures and shared control for rehabilitation of fine motor skills.* In EuroHaptics Conference, 2007 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics 2007. Second Joint, pages 304–309. IEEE, 2007. (Cited in page 29.)

[Student 1908] Student. *The probable error of a mean.* Biometrika, pages 1–25, 1908. (Cited in page 106.)

[Stulp 2013] Freek Stulp, Gennaro Raiola, Antoine Hoarau, Serena Ivaldi and Olivier Sigaud. *Learning compact parameterized skills with a single regression.* In Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on, pages 417–422. IEEE, 2013. (Cited in pages 19 and 54.)

[Taylor 2016] Russell H Taylor, Arianna Menciassi, Gabor Fichtinger, Paolo Fiorini and Paolo Dario. *Medical robotics and computer-integrated surgery.* In Springer handbook of robotics, pages 1657–1684. Springer, 2016. (Cited in page 35.)

[Tonietti 2005] Giovanni Tonietti, Riccardo Schiavi and Antonio Bicchi. *Design and control of a variable stiffness actuator for safe and fast physical human/robot interaction.* In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pages 526–531. IEEE, 2005. (Cited in page 49.)

[Tukey 1949] John W Tukey. *Comparing individual means in the analysis of variance.* Biometrics, pages 99–114, 1949. (Cited in page 106.)

[Tung 1995] Chao-Ping Tung and Avinash C Kak. *Automatic learning of assembly tasks using a dataglove system.* In Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on, volume 1, pages 1–8. IEEE, 1995. (Cited in page 16.)

[Turro 2001a] N. Turro, O. Khatib and E. Coste-Maniere. *Haptically augmented teleoperation.* In IEEE International Conference on Robotics and Automation, 2001. Proceedings 2001 ICRA, 2001. (Cited in pages 30 and 34.)

[Turro 2001b] Nicolas Turro and Oussama Khatib. *Haptically augmented teleoperation.* In Experimental Robotics VII, pages 1–10. Springer, 2001. (Cited in page 27.)

[Tykal 2016] Martin Tykal, Alberto Montebelli and Ville Kyrki. *Incrementally assisted kinesthetic teaching for programming by demonstration.* In The Eleventh ACM/IEEE International Conference on Human Robot Interaction, pages 205–212. IEEE Press, 2016. (Cited in pages 36, 37, and 54.)

[Vakanski 2012] A. Vakanski, I. Mantegh, A. Irish and F. Janabi-Sharifi. *Trajectory Learning for Robot Programming by Demonstration Using Hidden Markov Model and Dynamic Time Warping.* IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, August 2012. (Cited in page 30.)

[Vakanski 2017] Aleksandar Vakanski and Farrokh Janabi-Sharifi. Robot learning by visual observation. John Wiley & Sons, 2017. (Cited in page 17.)

[Van Damme 2010] M Van Damme, P Beyl, B Vanderborght, R Van Ham, I Vanderniepen, A Matthys, P Cherelle and D Lefeber. *The role of compliance in robot safety.* In Proceedings of the Seventh IARP Workshop on Technical Challenges for Dependable Robots in Human Environments, pages 65–71, 2010. (Cited in page 49.)

[Vitrani 2017] Marie-Aude Vitrani, Cécile Poquet and Guillaume Morel. *Applying virtual fixtures to the distal end of a minimally invasive surgery instrument.* IEEE Transactions on Robotics, vol. 33, no. 1, pages 114–123, 2017. (Cited in page 24.)

[Wisama Khalil 1999] Etienne Dombre Wisama Khalil. Modélisation, identification et commande des robots. Hermès science, Paris, janvier 1999. (Cited in pages 45 and 47.)

[Xia 2008] Tian Xia, Clint Baird, George Jallo, Kathryn Hayes, Nobuyuki Nakajima, Nobuhiko Hata and Peter Kazanzides. *An integrated system for planning, navigation and robotic assistance for skull base surgery.* The International Journal of Medical Robotics and Computer Assisted Surgery, vol. 4, no. 4, pages 321–330, 2008. (Cited in page 34.)

[Xia 2013] T. Xia, S. Léonard, I. Kandaswamy, A. Blank, L. L. Whitcomb and P. Kazanzides. *Model-based telerobotic control with virtual fixtures for satellite servicing tasks.* In 2013 IEEE International Conference on Robotics and Automation, pages 1479–1484, May 2013. (Cited in pages 24 and 29.)

[Yamamoto 2012] Tomonori Yamamoto, Niki Abolhassani, Sung Jung, Allison M Okamura and Timothy N Judkins. *Augmented reality and haptic interfaces for robot-assisted surgery.* The International Journal of Medical Robotics and Computer Assisted Surgery, vol. 8, no. 1, pages 45–56, 2012. (Cited in page 29.)

[Yoon 2014] H.U. Yoon, R.F. Wang and S.A. Hutchinson. *Modeling user's driving-characteristics in a steering task to customize a virtual fixture based on task-performance.* In Robotics and Automation (ICRA), 2014 IEEE International Conference on, pages 625–630, May 2014. (Cited in page 31.)

[Zeestraten 2017] Martijn JA Zeestraten, Ioannis Havoutis, Joao Silvério, Sylvain Calinon and Darwin G Caldwell. *An approach for imitation learning on Riemannian manifolds.* IEEE Robotics and Automation Letters, vol. 2, no. 3, pages 1240–1247, 2017. (Cited in page 65.)

[Zhang 2012] Dongwen Zhang, Lei Wang, Jia Gu, Zhicheng Li and Ken Chen. *Realization of spatial compliant virtual fixture using eigenscrews.* In Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE, pages 1506–1509. IEEE, 2012. (Cited in pages 32 and 66.)

[Zhang 2014] Dongwen Zhang, Qingsong Zhu, Jing Xiong and Lei Wang. *Dynamic virtual fixture on the Euclidean group for admittance-type manipulator in deforming environments.* Biomedical engineering online, vol. 13, no. 1, page 51, 2014. (Cited in page 66.)

Title: Intuitive, Iterative and Assisted Virtual Guides Programming for Human-Robot Comanipulation

Keywords: Virtual guides, programming by demonstration, collaborative robotics, comanipulation

Abstract: For a very long time, automation was driven by the use of traditional industrial robots placed in cages, programmed to repeat more or less complex tasks at their highest speed and with maximum accuracy. This robot-oriented solution is heavily dependent on hard automation which requires pre-specified fixtures and time consuming programming, hindering robots from becoming flexible and versatile tools. These robots have evolved towards a new generation of small, inexpensive, inherently safe and flexible systems that work hand in hand with humans. In these new collaborative workspaces the human can be included in the loop as an active agent. As a teacher and as a co-worker he can influence the decision-making process of the robot. In this context, virtual guides are an important tool used to assist the human worker by reducing physical effort and cognitive overload during tasks accomplishment. However, the construction of virtual guides often requires expert knowledge and modeling of the task. These limitations restrict the usefulness of virtual guides to scenarios with unchanging constraints. To overcome these challenges and enhance the flexibility of virtual guides programming, this thesis presents a novel approach that allows the worker to create virtual guides by demonstration through an iterative method based on kinesthetic teaching and displacement splines. Thanks to this approach, the worker is able to iteratively modify the guides while being assisted by them, making the process more intuitive and natural while reducing its painfulness. Our approach allows local refinement of virtual guiding trajectories through physical interaction with the robots. We can modify a specific cartesian keypoint of the guide or re-demonstrate a portion. We also extended our approach to 6D virtual guides, where displacement splines are defined via Akima interpolation (for translation) and quadratic interpolation of quaternions (for orientation). The worker can initially define a virtual guiding trajectory and then use the assistance in translation to only concentrate on defining the orientation along the path. We demonstrated that these innovations provide a novel and intuitive solution to increase the human's comfort during human-robot comanipulation in two industrial scenarios with a collaborative robot (cobot).

Titre: Programation Intuitive, Itérative et Assistée de Guides Virtuels pour la Comanipulation Homme-Robot
Mots clés : Guides vrituels, programation par démonstration, robotique collaborative, comanipulation
Résumé : Pendant très longtemps, l'automatisation a été assujettie à l'usage de robots industriels traditionnels placés dans des cages et programmés pour répéter des tâches plus ou moins complexes au maximum de leur vitesse et de leur précision. Cette automatisation, dite rigide, possède deux inconvénients majeurs : elle est chronophage dû aux contraintes contextuelles applicatives et proscrit la présence humaine. Il existe désormais une nouvelle génération de robots avec des systèmes moins encombrants, peu coûteux et plus flexibles. De par leur structure et leurs modes de fonctionnement ils sont intrinsèquement sûrs ce qui leurs permettent de travailler main dans la main avec les humains. Dans ces nouveaux espaces de travail collaboratifs, l'homme peut être inclus dans la boucle comme un agent décisionnel actif. En tant qu'instructeur ou collaborateur il peut influencer le processus décisionnel du robot : on parle de robots collaboratifs (ou cobots). Dans ce nouveau contexte, nous faisons usage de guides virtuels. Ils permettent aux cobots de soulager les efforts physiques et la charge cognitive des opérateurs. Cependant, la définition d'un guide virtuel nécessite souvent une expertise et une modélisation précise de la tâche. Cela restreint leur utilité aux scénarios à contraintes fixes. Pour palier ce problème et améliorer la flexibilité de la programmation du guide virtuel, cette thèse présente une nouvelle approche par démonstration : nous faisons usage de l'apprentissage kinesthésique de façon itérative et construisons le guide virtuel avec une spline 6D. Grâce à cette approche, l'opérateur peut modifier itérativement les guides tout en gardant leur assistance. Cela permet de rendre le processus plus intuitif et naturel ainsi que de réduire la pénibilité. La modification locale d'un guide virtuel en trajectoire est possible par interaction physique avec le robot. L'utilisateur peut déplacer un point clé Cartésien ou modifier une portion entière du guide avec une nouvelle démonstration partielle. Nous avons également étendu notre approche aux guides virtuels 6D, où les splines en déplacement sont définies via une interpolation Akima (pour la translation) et une interpolation quadratique des quaternions (pour l'orientation). L'opérateur peut initialement définir un guide virtuel en trajectoire, puis utiliser l'assistance en translation pour ne se concentrer que sur la démonstration de l'orientation. Nous avons appliqué notre approche dans deux scénarios industriels utilisant un cobot. Nous avons ainsi démontré l'intérêt de notre méthode qui améliore le confort de l'opérateur lors de la comanipulation.